GDC | March 21-25, 2022
San Francisco, CA

Practical Automation: A Guide to Random Game Content Generation

Yiheng Zhou / Shuai Xu

#GDC22

When we generated contents and terrains for our games, we encountered multiple problems. These problems are likely to exist in many game design processes.

March 21-25, 2022
San Francisco, CA

e.g.

Players always complain that the **map is unfair**

Players always complain that the **spawn points are unreasonable**

Players always complain about **insufficient resources**

GDC

March 21-25, 2022
San Francisco, CA

#GDC22

Contents

- **Summarize** the **general problems**
- **Simulation story**: two teams mine golds under snow mountains
- **Unfair**: 2 maps comparison
- **Mathematical factor analysis**: fairness & stability & algorithms
- Takeaway: **Automatic Generation**

General problems to solve

- Given **two maps**, how to **compare** which is better?
- Is it **fair** to the two teams?
- Is it **still fair** when the two teams have **different numbers**?
- Is it easy to get props **at the beginning of the game**?
- To be fair, **how big** a map is needed?
- **Which** random generator **algorithm** to choose?
- How many prop rewards are needed?

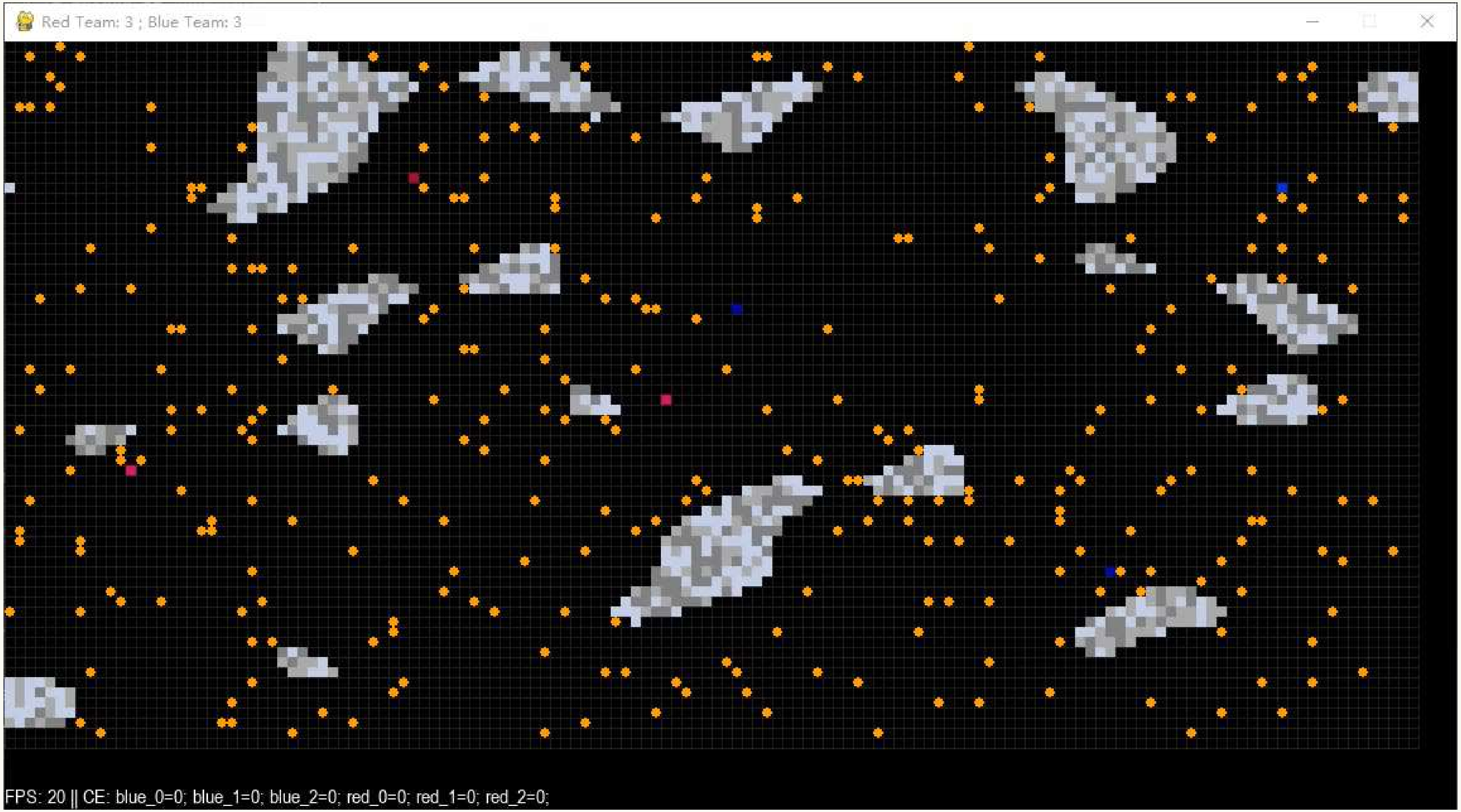
Dive into Practice

- To our knowledge, the degree of convenience from the **spawn points** to **props placement** greatly affects the **initial growth** of the game player.
- While the growth of game players **in the middle and late stages** of the game mainly depends on the player's own experience and skills.
- Therefore, in order to **reduce irrelevant factors**, the effectiveness and fairness of the map should be compared at **the early stage** of the game.
- Comprehensive **simulation** is an effective way to test hypotheses.
- The **controlled variable method** is the key to the simulation experiment.

Simulation

- **Two teams（red and blue）** are mining **gold mines** on the edge of the **snow-capped mountains**
- Gold mines and snow mountains are **generated by algorithms**
- The red team players are born from the **west** (randomly generated left half area). While the blue are the **east**.
- Each team member uses the **same pathfinding strategy**
- Players cannot enter the snow mountains
- **Settlement** when teams **first meet (the early stage)** or when the gold mine resources are exhausted

↑ mining **gold mines** under the **snow-capped mountains**


Red Team: 3 ; Blue Team: 3
FPS: 20 || CE: blue_0=0; blue_1=0; blue_2=0; red_0=0; red_1=0; red_2=0;

Why use the early stage（i.e. first meet）？

- Easy to **implement**
- Easy to **interpret**
- **Quick results:** Reduce the amount of computing power
- In line with **mathematical principles**

When two teams meet for the first time, they may ***fight or share the gold mine information****. This makes the game enter the middle and late stages.*

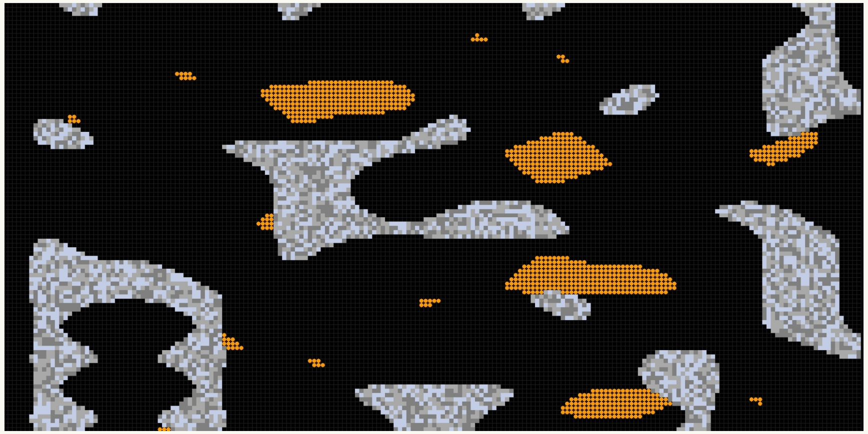
Then?

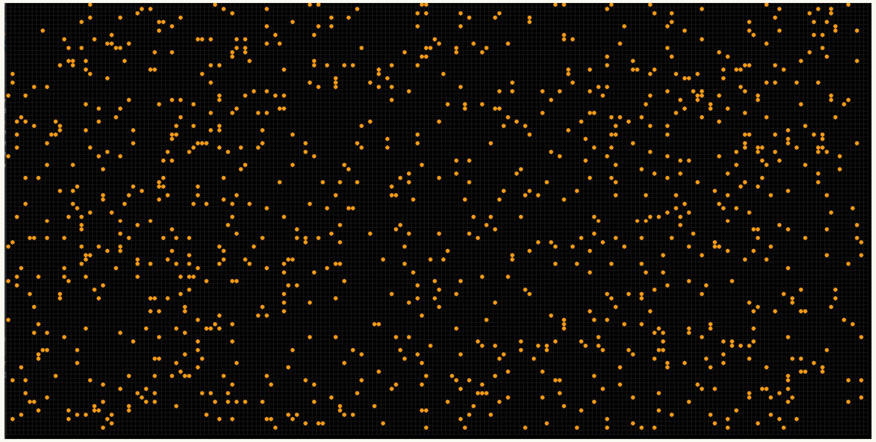
- The **difference between the gold gain** of the two teams in a single game can reflect the **difference between the spawn points and the location of the resources**, since they use the same pathfinding algorithm.
- The **variance of the difference** caused by thousands game times can reflect the **stability** of the map.
- The **expectation** of the score difference caused by thousands game times can reflect the **fairness** of the map.
- The **variances** of **different maps** can compare the **quality of the map.**
- The **variances** of **different** random generator **algorithms** can compare the **effectiveness of algorithm**.
- etc.

Model Definition

- $\mathbb{R}$ red team player set
- B blue team player set
- $\delta = \sum_{p \in \mathbb{R}} p - \sum_{p \in B} p$ diff golds in one game
- Δ diff golds set of games
- $\sigma^2 = Var(\Delta)$ variance of games under some settings
- $\mu\ (\Delta)$ mean value of Δ
- $\sigma\ (\Delta)$ standard deviation of Δ

Example1: map1 vs map2



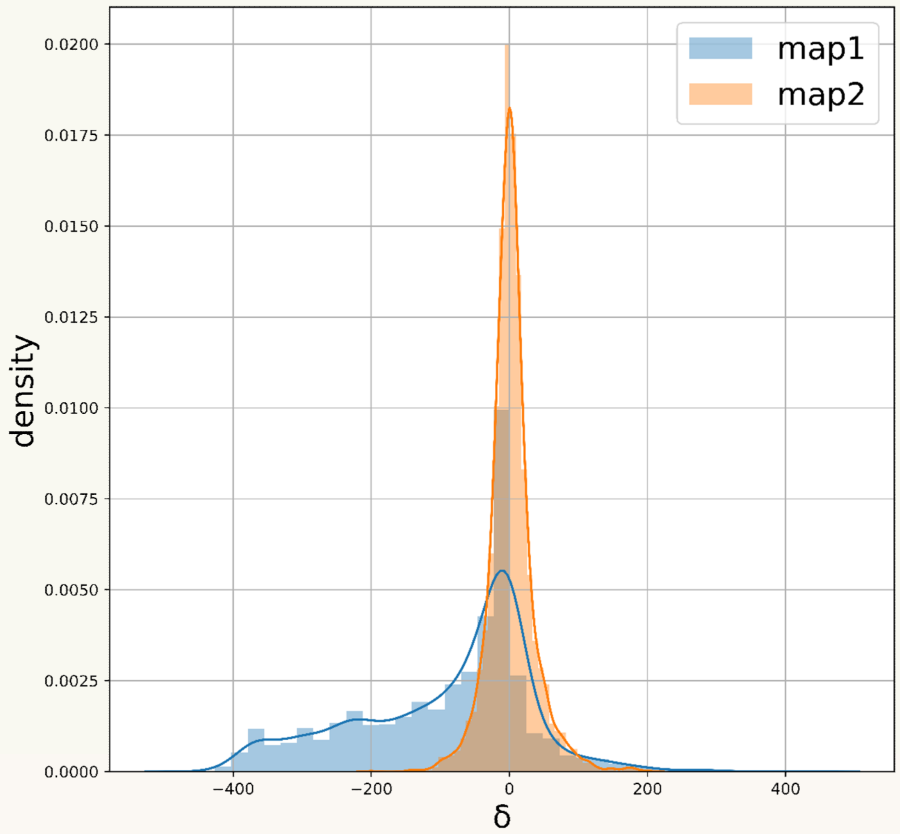


- *map1: snow mountain piles and gold piles*
- *map2: only scattered*

*Can you **directly tell** which map is **more gainful** for the red team?   (red left, blue right)*

*How to **verify** your conclusion?*

Example: map1 vs map2



← **density plot of δ** of 3000 simulation times each map

Confirmed:

1. map2 is **symmetric** about 0 point.
2. map2 is more **stable**, since the curve is narrow and tall
3. map1 is **asymmetry** about 0, though it peaks at 0.
4. map1 has **multiple peaks**.
5. Obviously, map1 is more **gainful** for the blue team.

How to test the hypothesis that "map1 is more gainful for the blue team"?

Example: map1 vs map2

Significance Test: map1 is more gainful for the blue team

(1) H0: blue team is not more gainful

(2) H1: blue team is more gainful

(3) u_r = 175.51 , $\overline{u_b}$ = 266.15

(4) SE = $\frac{\sigma_r}{\sqrt{n}}$ = 2.78

(5) z = $(\overline{u_b} - u_r)$ / SE = 32.60

Using **Z-test theory**, we can reject the H0 with a probability of **99.9+%** , then **accept H1**.

Example: map1 vs map2

- Conclusion from **Z-test**: **map1 is not fair**

- Is it possible to turn the map1 into **fair** by increasing **the player number** of the red team?
- Is there **any other way**?

Consider Influencing factors

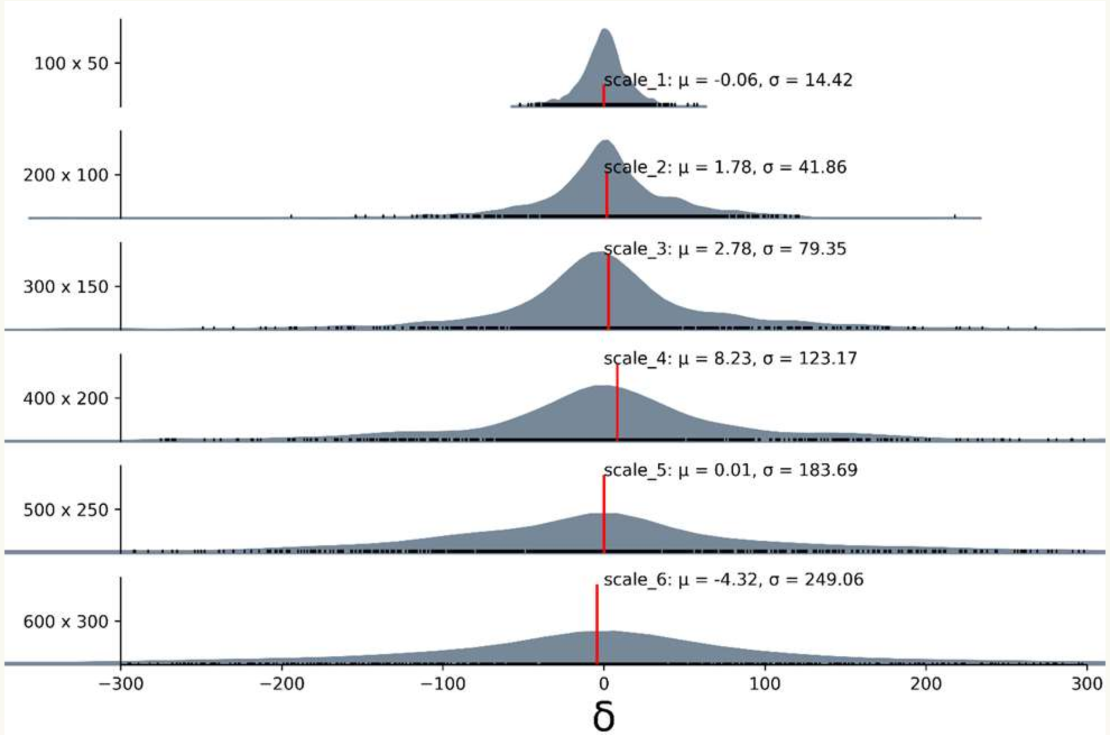
- Map scale
- Team size
- Gold quantity
- Pathfinding algorithm
- Random generator algorithm / seed
- etc.

Map scale: bigger is better?

*Does a **bigger** map means **more stability**?*
*Whether the map is **bigger** is less likely to be **unfair?***

- **constant gold proportion = 3%**
- constant snow proportion
- fixed random generator (with random seed)
- fixed team size
- fixed pathfinding algorithm
- **Map scales: 100 x 50, 200 x 100, 300 x 150, 400 x 200, 500 x 250, 600 x 300**

Map scale: bigger is better

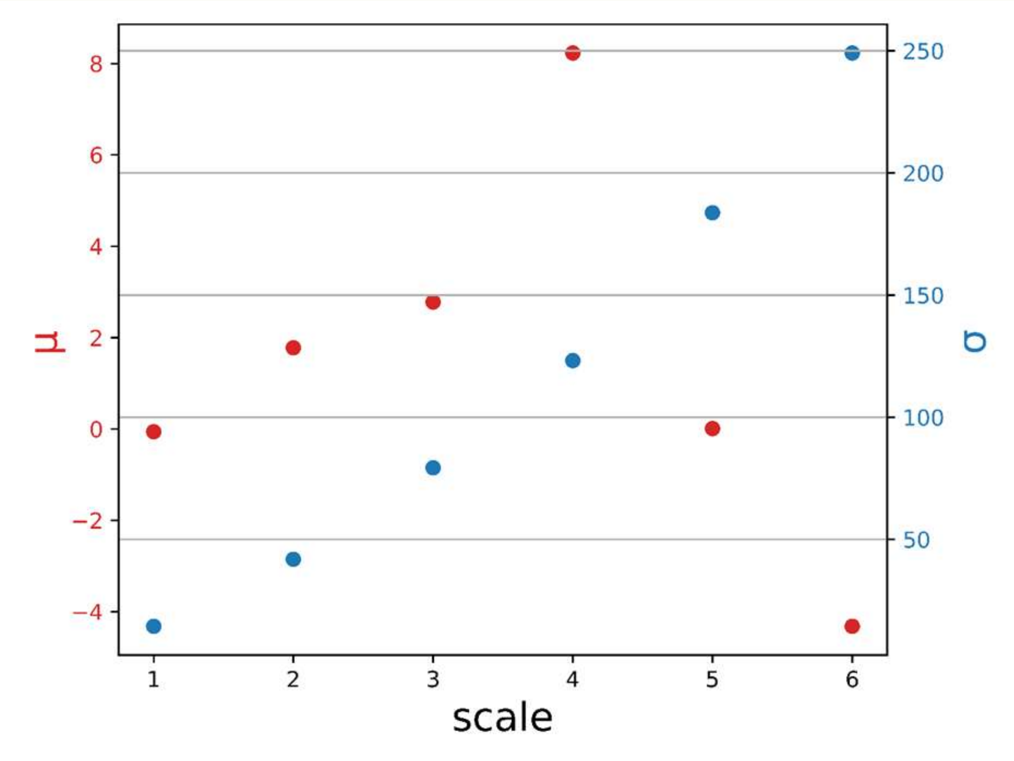


← half-violin plot of map scales

1. From the drawing, **map with bigger scale leads to bigger** σ （**curve tends to be flat**）, means more **instability**.
2. Since we use random seed for generation, We don't directly see apparent unfairness here. However, we know that big σ may lead to **big uncertainty of fairness**.

What we have overlooked so far?

Map scale: bigger is better?



←**Gaussian model params of map scales**

It seems that there is a **linear relationship** between the **map scale** and the standard deviation **σ**.

At the moment we have overlooked a factor,

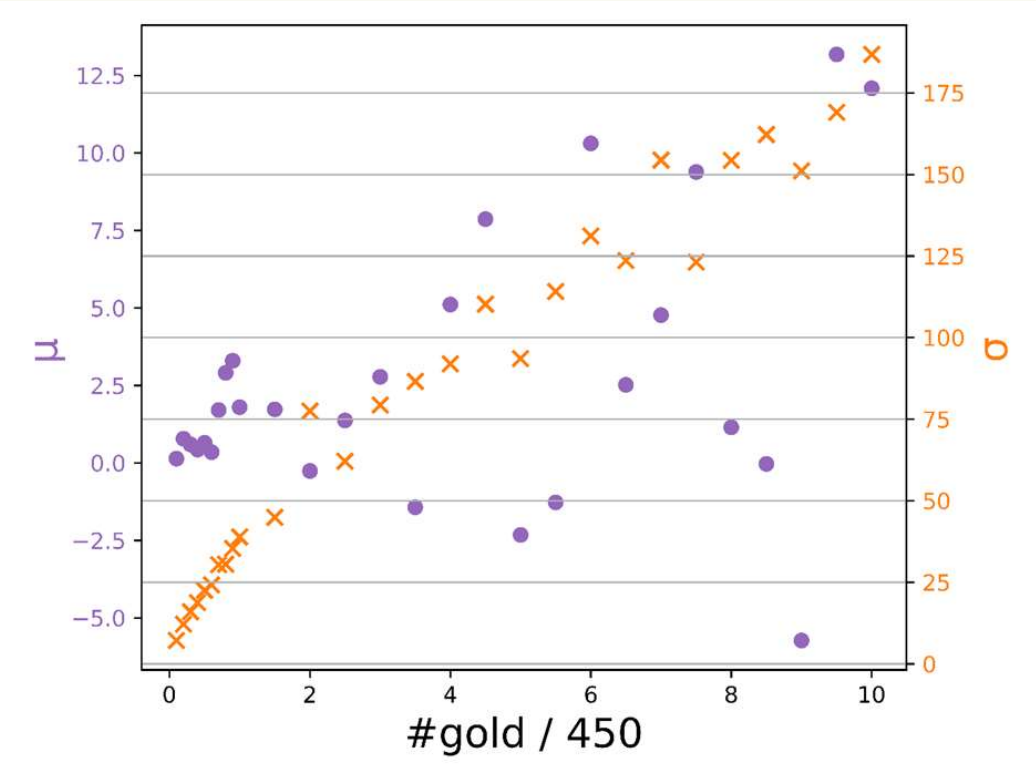
gold proportion ≠ gold amount 😲

e.g. 100x50x0.03 ≠ 600x300x0.03

$\sigma \propto$ map scale **?**

$\sigma \propto$ #gold **?**

Map scale: #gold



← **Gaussian model #gold** (with fixed map scale: **300x150**, fixed #snow, etc⋯)

$\sigma \propto$ #gold **?**

Compute **Pearson correlation coefficient**:

p (σ, #gold) = 0.9820

It seems linearity.

Map scale: conclusion

According to preliminary analysis,

1. **σ ∝ map scale , when constant gold proportion**
2. ***σ*** **∝ #gold**, when other settings are fixed
3. If conditions permit, **reducing the map scale** and **#gold** can **increase the stability** of the map.

Noting that these conclusions may be different in different types of games.

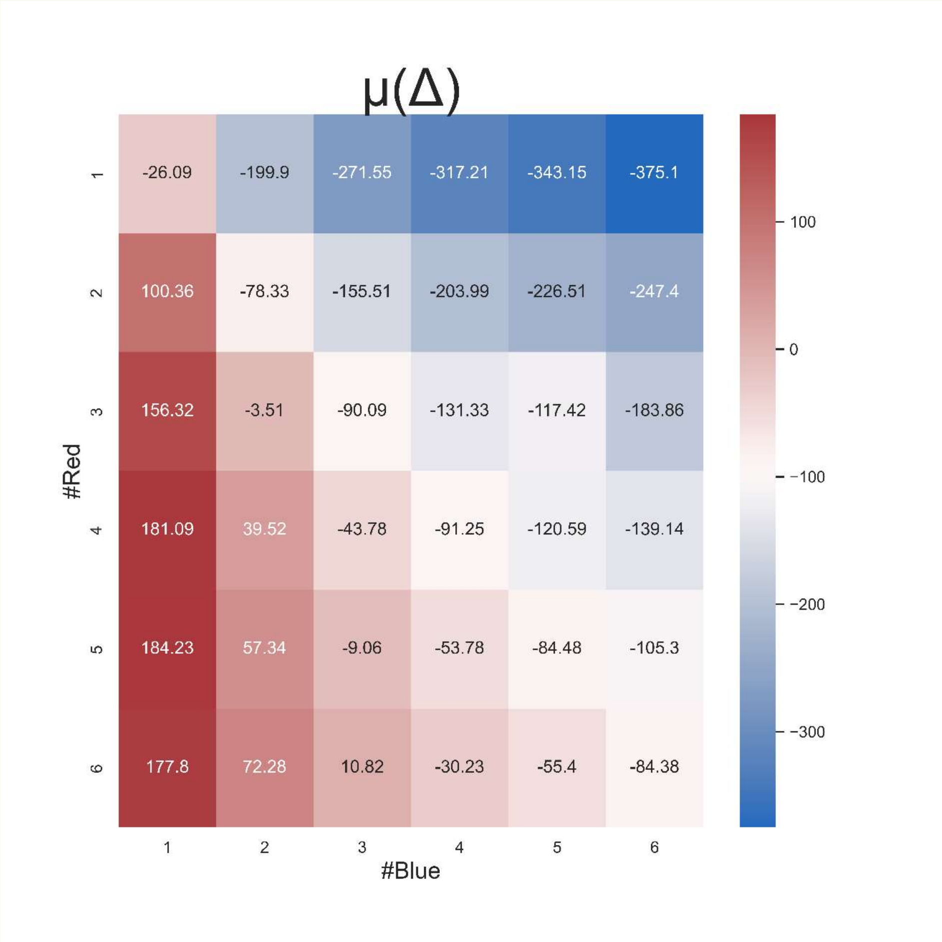
Team size : more people is better?

Back to the previous question: "Is it possible to ***make*** *the map1* ***fair by increasing the player number*** *of the red team?"*

- use the generated map 'map1' (by Perlin generator, will be introduced later)
- Try different team sizes (**red x blue**)： 1 x 1, 1 x 2 , 3 x 3, 6 x 2, etc…

To check whether **different team sizes** will change the **fairness** of the game

Team size : more people is better?



←Matrix of μ(Δ) with respect to team size

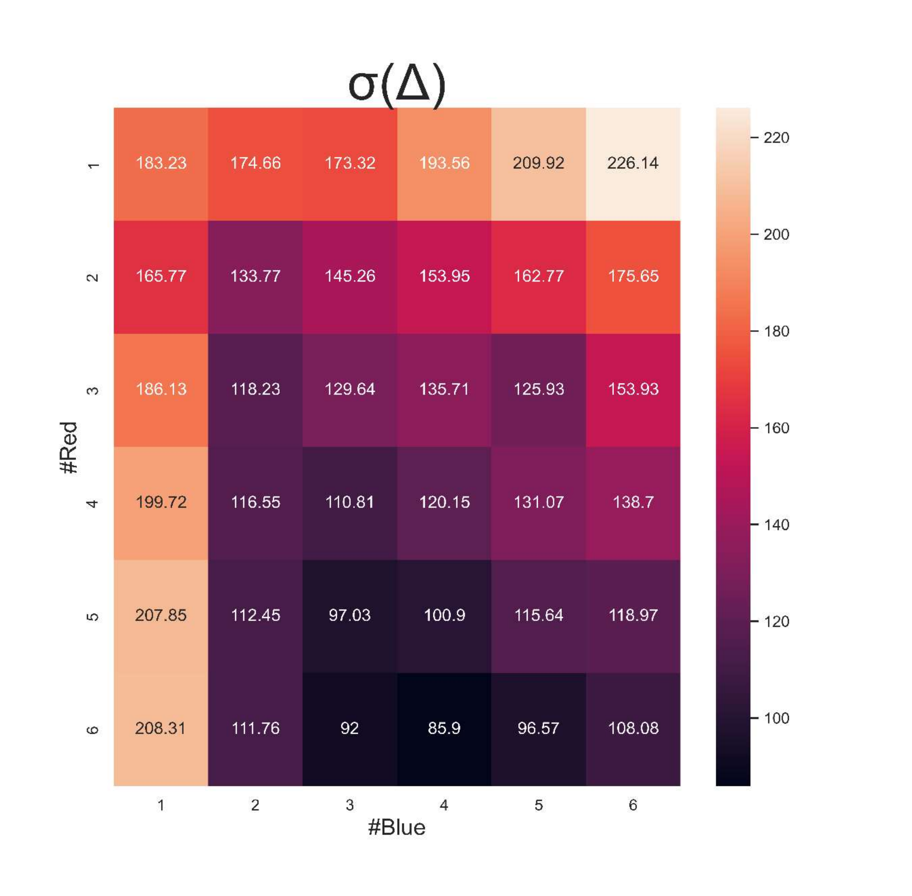
1.Row axis is for blue team members and column axis is for red team members.

2. From the drawing, **in most cases the blue team has the advantage** in this map.

3. Fortunately, **increasing the number of red teams may change fairness**, which means to increase the probability of the red team winning.

4. The team size 3 x 2 maybe a good choice for fairness.

Team size : more people is better?



←Matrix of σ (Δ) with respect to team size

1. The darker the color, the more stable.
2. The team size **6 x 4 reaches the max stability**, **though it is quite unfair**. It's interesting that **the blue team still has the upper hand** in this situation.

Pathfinding algorithm

The pathfinding algorithm will affect the simulation results:

e.g.

- **Walking to a certain gold location more slowly** may cause the gold to be mined by others first.
- If you tend to **go to the opposite side,** you may meet someone from the other team faster.
- If you always **walk with your teammates**, you might waste a person.

Pathfinding algorithm: principles

What to consider when designing a pathfinding algorithm for the simulation:

- AI is **greedy for gold**
- Does **not** always follow a **fixed route**, but there is a **limit** to **randomness**
- Every AI has the **same pathfinding algorithm/ algorithms**
- Have some intelligence and will **not fall into the same trap for a long time**
- Try to **imitate human** behaviors

A feasible greedy pathfinding algorithm

For Each Step, For Each AI:

1. Choose the one with **the closest European distance** from the valid golds, Denoted as **T**. Current location is **C**.

2. Initialize a **Candidate List CL**.

3. Get the next location by **Dijkstra Shortest Path algorithm**. Add the location as **a candidate** to CL.

4**. For each movable location L**:

    4.1 **IF distance(C, T) < distance(L,T),** then **add L to CL.**

    4.2 **ELSE**, then **add** L to CL **with probability p**. (to escape the traps)

5. **Randomly select** a **candidate** from CL **as the next location** for this AI.

Generators

*Generators are used to **generate resources** (gold mines) and **obstacles** (snow mountains):*

- **Uniform** Random Selection
- **Perlin** (Simplex) Noise
- **Fractal** Noise
- Fourier Series Noise
- etc.

Generator 1:Uniform Random Selection

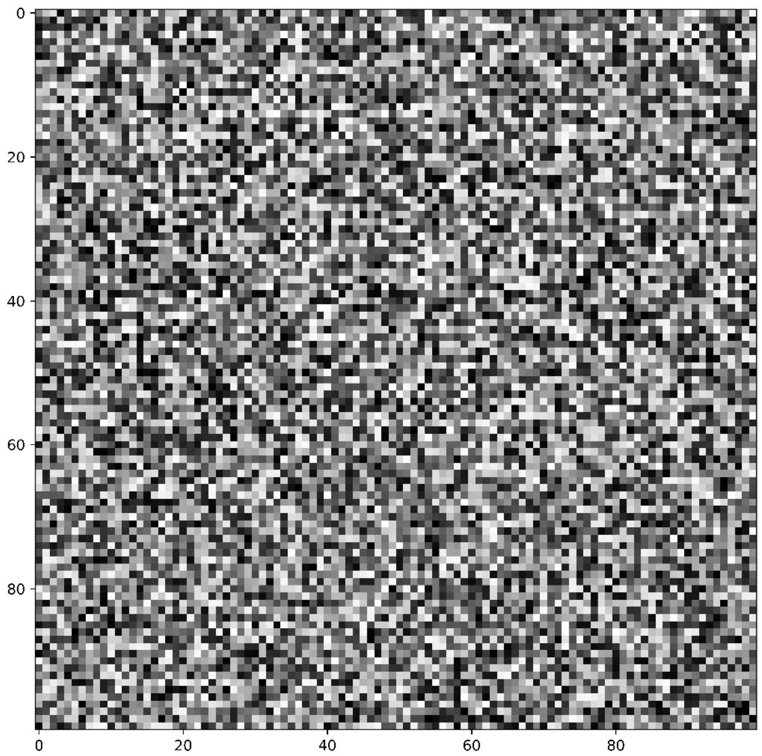
- Produce a random value between **0.0-1.0 from uniform distribution** for **each location** on the grid

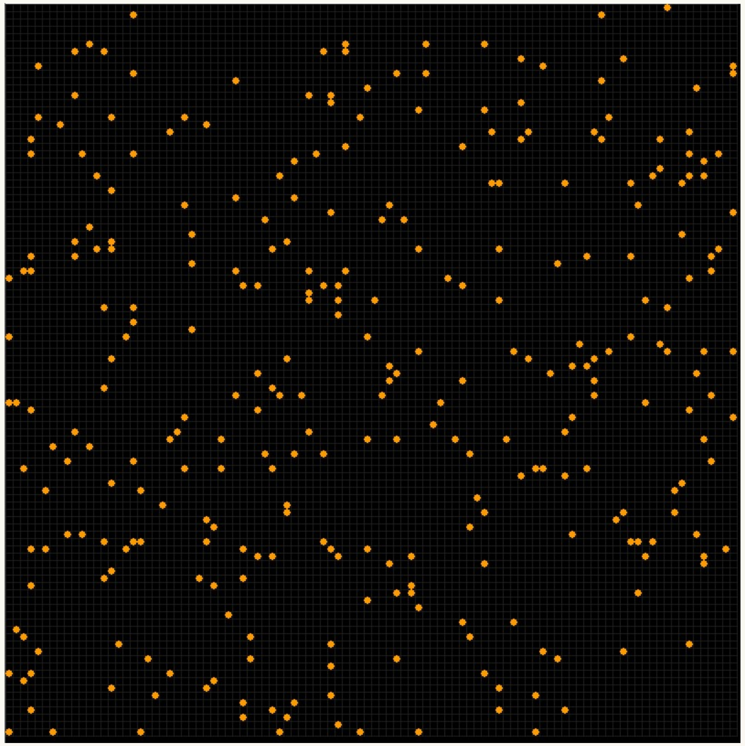
$$f(x) = \begin{cases} \frac{1}{b-a} \quad for\ a \leq x \leq b, \\ 0 \ \ for\ x < a\ or\ x > b \end{cases}$$

- Take the **locations** of the **largest n values** for generating items

- **Spawn items** in these **locations**

Generator 1: Uniform Random Selection





Generator 2: Perlin Noise

Ken Perlin developed Perlin Noise in 1983.

- It is statistically **invariant to rotation**.
- The **energy is concentrated** in a narrow band in the frequency spectrum, that is: the image is **continuous** and the **high frequency components are limited**.
- It is statistically **invariant to transformation**.

Generator 2: Perlin Noise

Steps （We will not discuss the details here）：

1. Defining a **grid** of random **gradient vectors**

noise2d(x, y) = z

2. Computing the **dot product** between the **gradient vectors** and their **offsets**

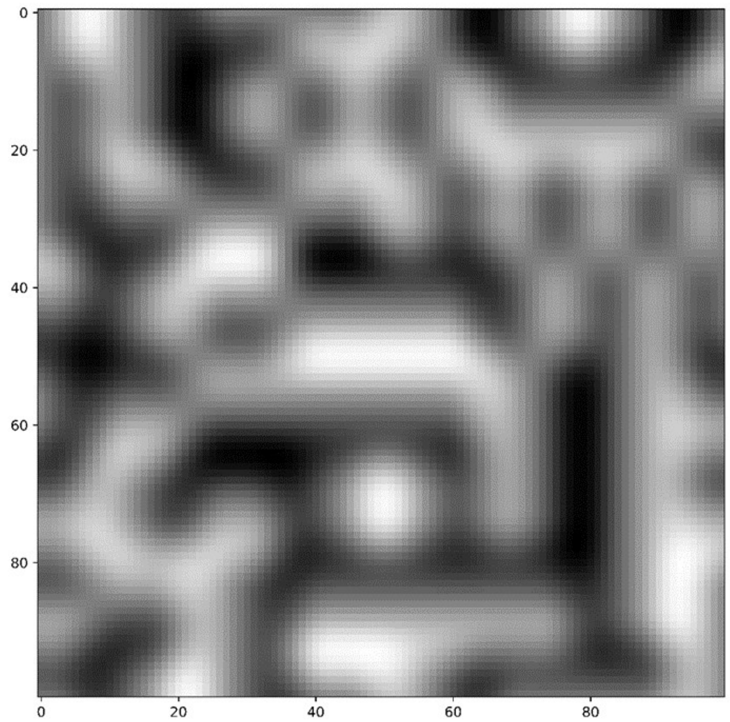
3. Do **interpolation** between these values

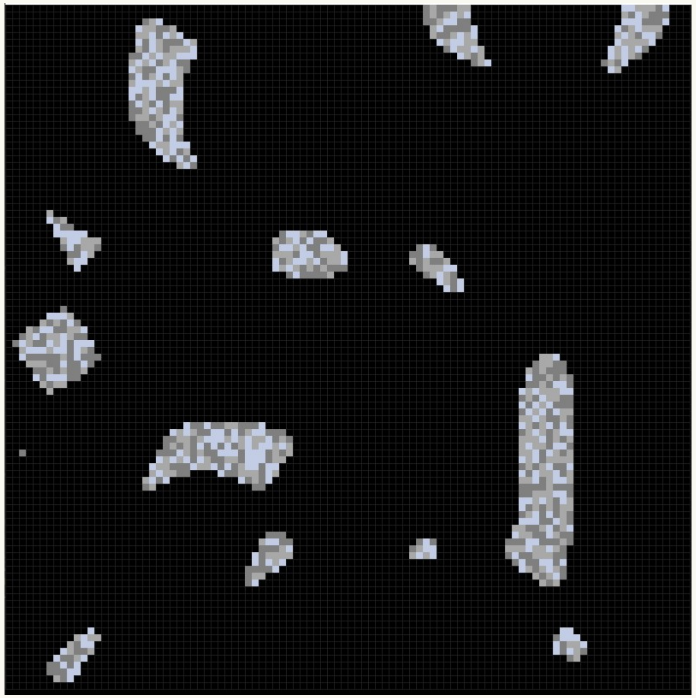
$s(t) = 6t^5 - 15t^4 + 10t^3$ // smooth function

$$f(x) = a_0 + s(x) \cdot (a_1 - a_0)$$

4. Use the **interpolated grid** to **generate terrain and items**

Generator 2: Perlin Noise



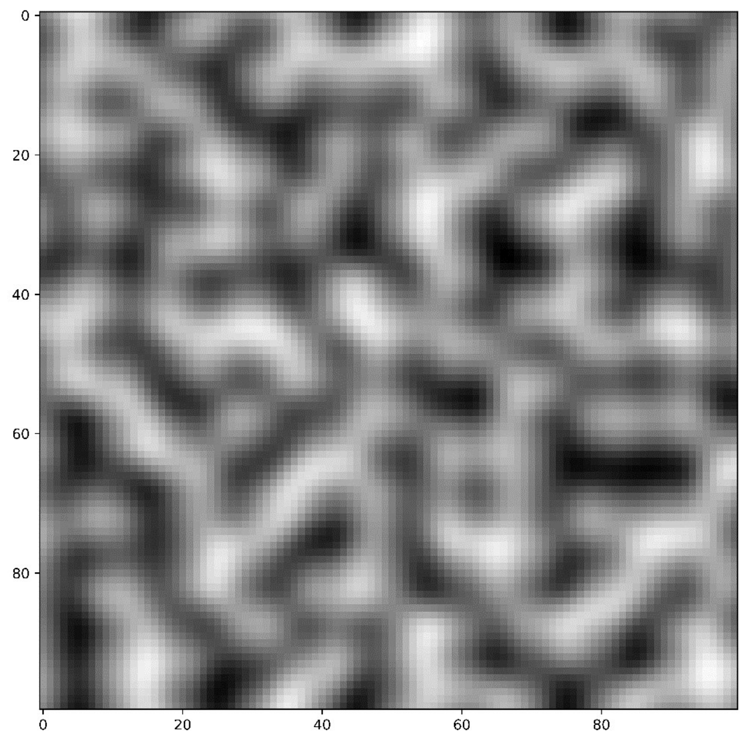


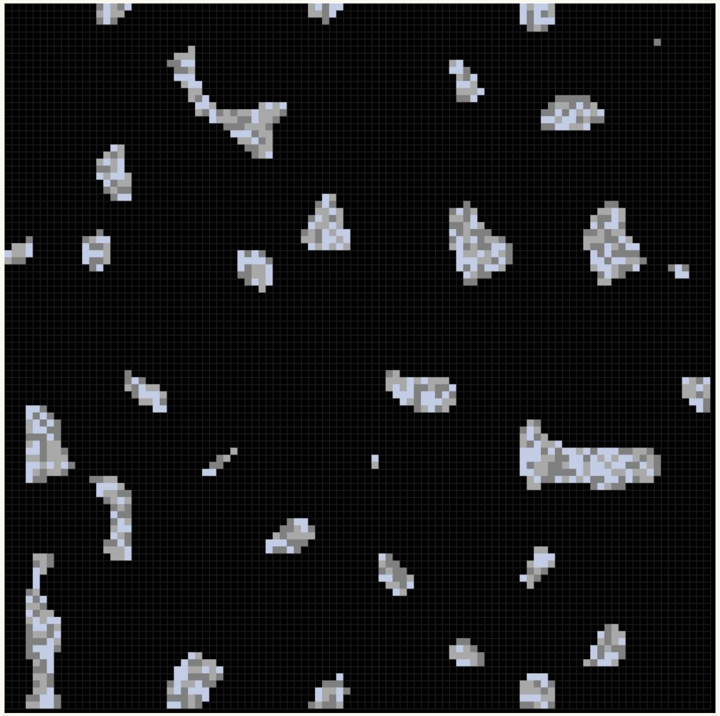
Generator 3: Fractal Noise

- **Based on Perlin Noise**
- Introduce **self similarity** and other effects neccesary for noise to be **fractal**
- **Closer** to the **natural** world

$$\sum_{t} \frac{|Noise(2^{i} point)|}{2^{i}} \quad \textit{// fractal function}$$

Generator 3: Fractal Noise

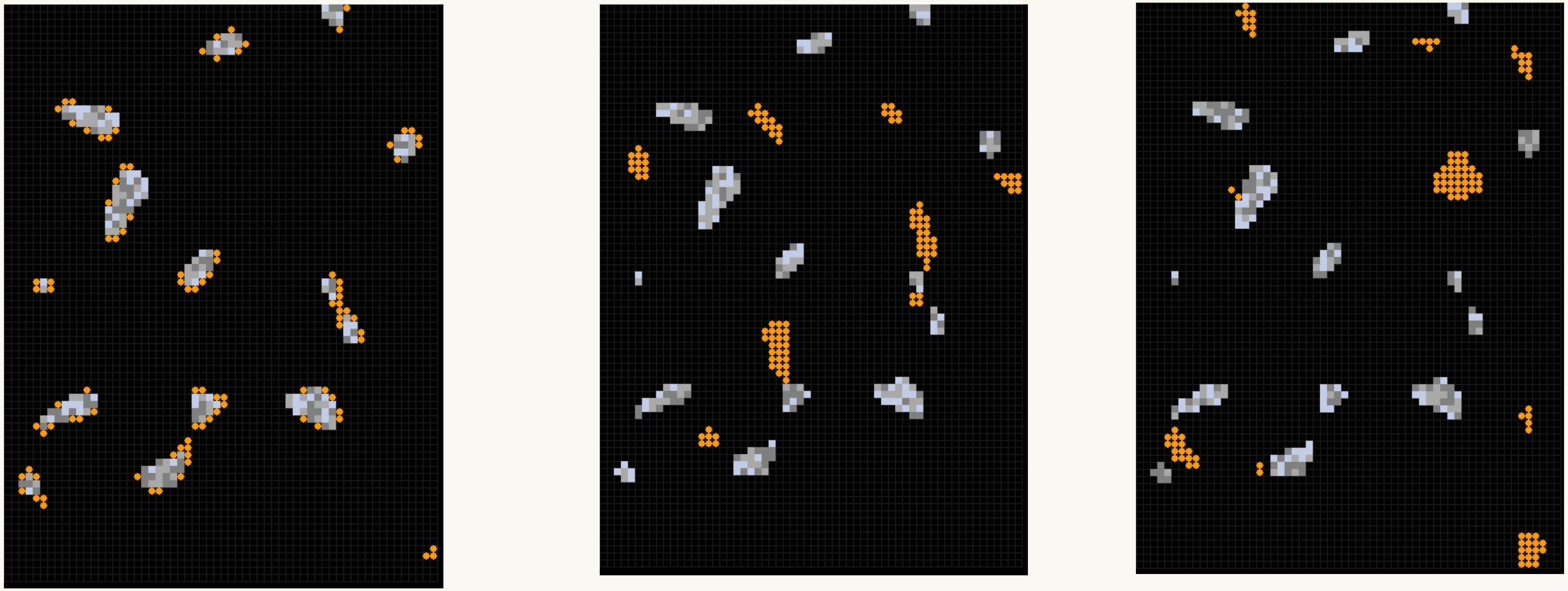




Generators: how to use?

- **Different** random **algorithms** can be used for **different resources and items**
- **Even the same algorithm** can **use different random seeds** for different type of items
- **Inappropriate settings of generators** may make the map **unfair** or have great **instability**

Equal seed vs Distinct seed



↑ Snows use **seed1**. Gold in the left grid uses equal seed. The other two use **seed2** and **seed3**.

The **equal seed** can bring gold mines **close** to the snow mountains.

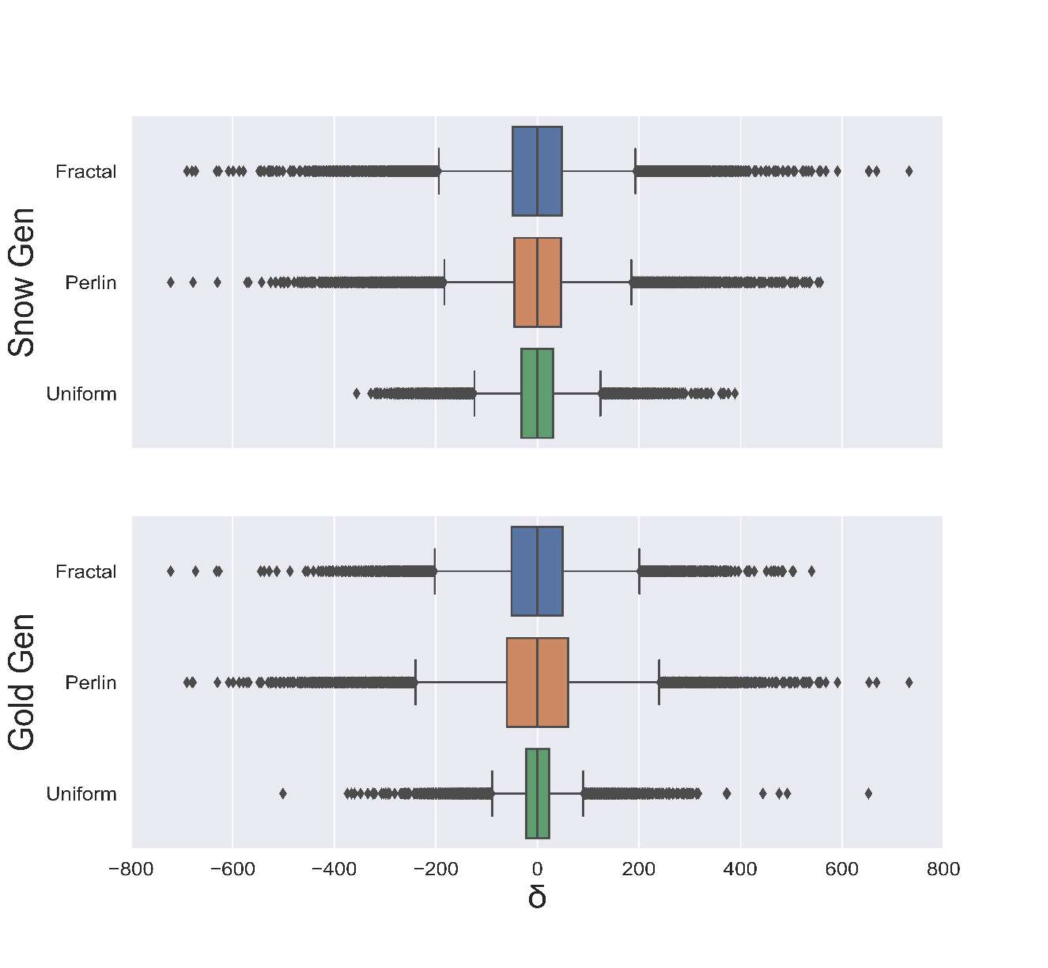
Generators

#	Equal Seed	Snow Gen	Gold Gen	μ	σ
1	N	Fractal	Fractal	-0.13	103.23
2	Y	Fractal	Fractal	-1.02	104.07
3	N	Fractal	Perlin	-0.53	148.73
4	Y	Fractal	Perlin	-3.63	153.59
5	N	Fractal	Uniform	-0.33	66.19
6	Y	Fractal	Uniform	-0.95	65.00
7	N	Perlin	Fractal	-0.29	102.72
8	Y	Perlin	Fractal	-0.59	103.34
9	N	Perlin	Perlin	-2.76	153.88
10	Y	Perlin	Perlin	6.04	152.70
11	N	Perlin	Uniform	1.72	55.27
12	Y	Perlin	Uniform	-0.00	57.13
13	N	Uniform	Fractal	1.81	89.19
14	Y	Uniform	Fractal	0.42	88.17
15	N	Uniform	Perlin	-0.55	69.99
16	Y	Uniform	Perlin	-1.43	67.61
17	N	Uniform	Uniform	-0.27	39.65
18	Y	Uniform	Uniform	-0.36	40.16

← Gaussian

1. In general **Perlin/Fractal has bigger σ** than Uniform Random Selection, since the **value exceeds 100**.
2. If use arbitrary seeds, Perlin/Fractal/Uniform can reach **Expectation close to 0**.

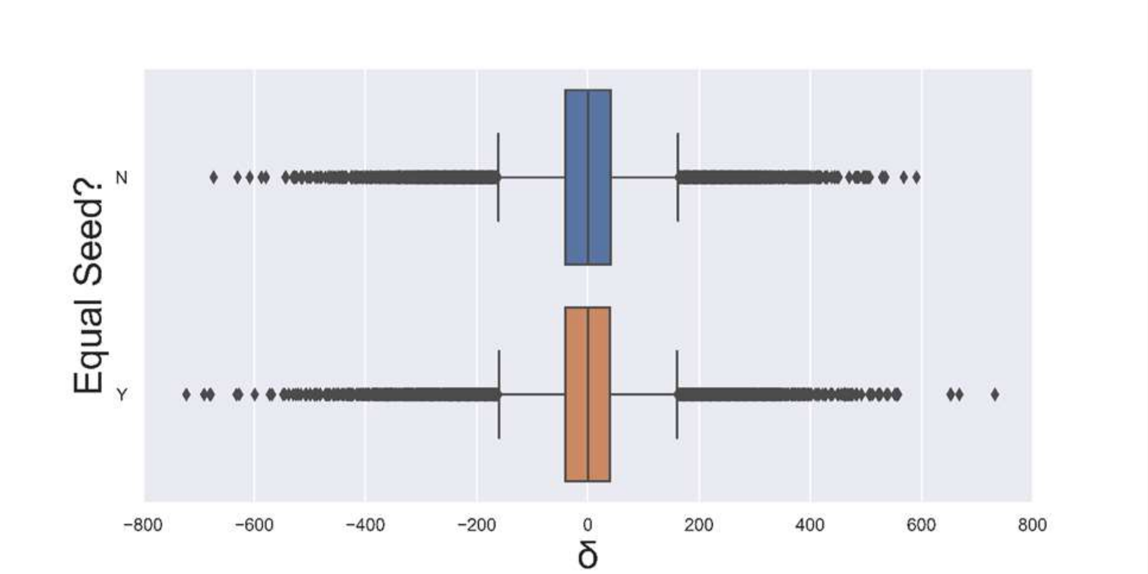
Generators



← Boxplot of δ with respect to generators

1. **Uniform** is the most **stable**. (small data range ignoring the outliers)
2. **Fractal** is the most **unstable** for generating **snow mountains**.
3. **Perlin** is the most **unstable** for generating **gold mines**.

Generators



← Boxplot of δ with respect to equal/distinct seed

We do **not** see **much difference** between using **equal seed** and **distinct seed** for generating snows and golds. **Though** the **maps** look very **different visually**.

Math analysis: Conclusion

*What **indicators** need to be considered?*

μ (Δ) Fairness； σ (Δ) Stability

*What **factors** need to be considered?*

Map scale; Gold quantity; Random generator algorithm / seed; etc.

***How to analyze**?*

Violin plot; Linear fit; Matrix heatmap; Visualization; Pivot table; Boxplot; etc.

Takeaway: Automatic Generation

1. Based on expert knowledge, **design the map scale, the quantity of rewards, the quantity of obstacles and the number of players**, etc.
2. **Set up indicators and acceptable limits**: such as **expectation $|\mu| < 10$** and **variance $\sigma < 30$**
3. Define the "**early stage**" of the game
4. Choose a **suitable** and **easy-to-implement generator**
5. Generate **several simplified maps** (remove irrelevant items)
6. Randomly generate AIs, and **perform several simulation experiments** on the map through appropriate pathfinding algorithms
7. **Calculate** the **indicators** of each map based on the results
8. **Accept** the **maps** with indicators that meet the limits

Thank You