

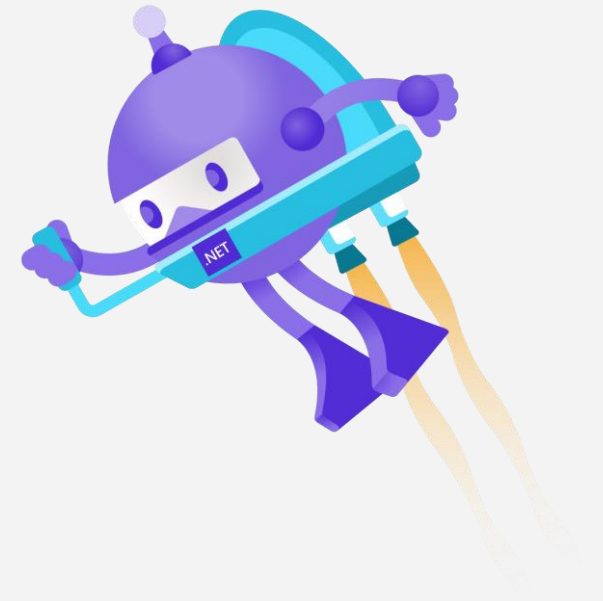
Agenda

- 01 | **How did I solve it?** 康威生命遊戲發想與解題方式
- 02 | **What did I learn from this homework?** 刻意練習之下學到什麼？
- 03 | **Conclusion** 總結

01

How did I solve it?

發想與解題方式



Milestone 1

Implement Basic Rules

Implement Basic Rules

1. 每個細胞有兩種狀態 - 存活或死亡，每個細胞與以自身為中心的周圍八格細胞產生互動。
2. 當前細胞為存活狀態時，當周圍的存活細胞低於2個時（不包含2個），該細胞變成死亡狀態。
3. 當前細胞為存活狀態時，當周圍有2個或3個存活細胞時，該細胞保持原樣。
4. 當前細胞為存活狀態時，當周圍有超過3個存活細胞時，該細胞變成死亡狀態。
5. 當前細胞為死亡狀態時，當周圍有3個存活細胞時，該細胞變成存活狀態。

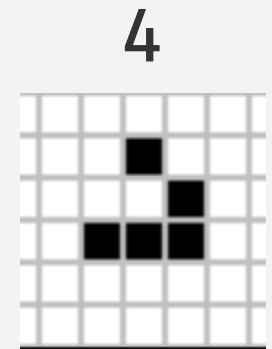
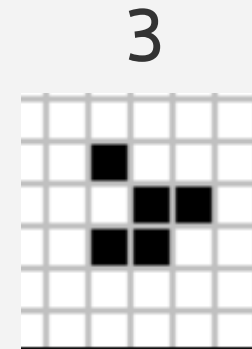
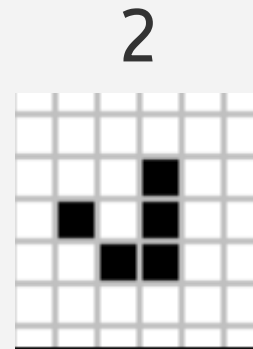
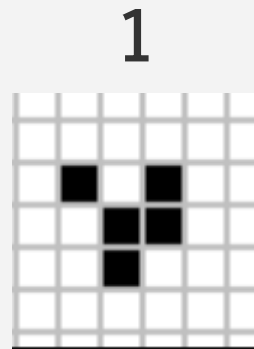
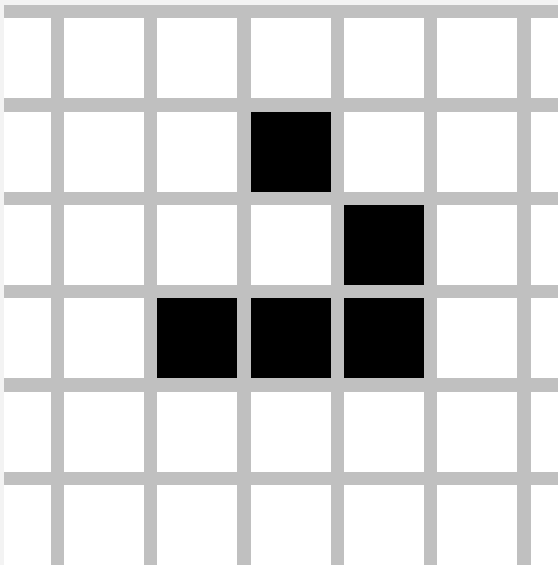
```
private bool CheckIsAlive()
{
    var isAlive = this.IsAlive;
    int aliveCount = 0;

    for (int i = 0; i < Partners.GetLength(0); i++)
        for (int k = 0; k < Partners.GetLength(1); k++)
            if (Partners[i, k].IsAlive) aliveCount++;

    if (this.IsAlive)
    {
        aliveCount--;
        if (aliveCount < 2 || aliveCount > 3)
            isAlive = false;
    }
    else
    {
        if (aliveCount == 3)
            isAlive = true;
    }
    return isAlive;
}
```

Glider 滑翔機

會移動的振盪狀態



回合制：同時間刷新整個畫面

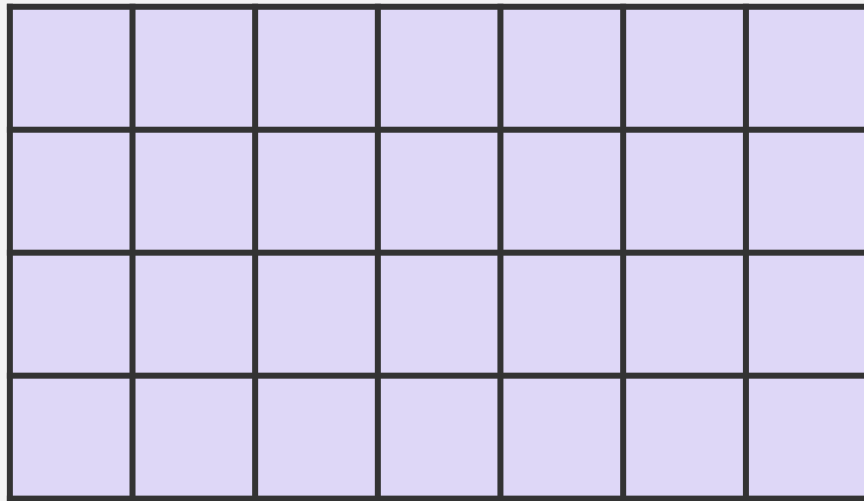
Milestone 2

OOP

哪些 **Class** 參與了這場遊戲？

Class

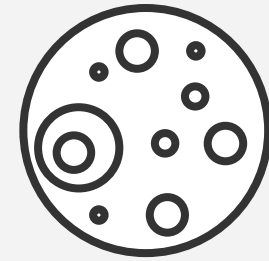
Map



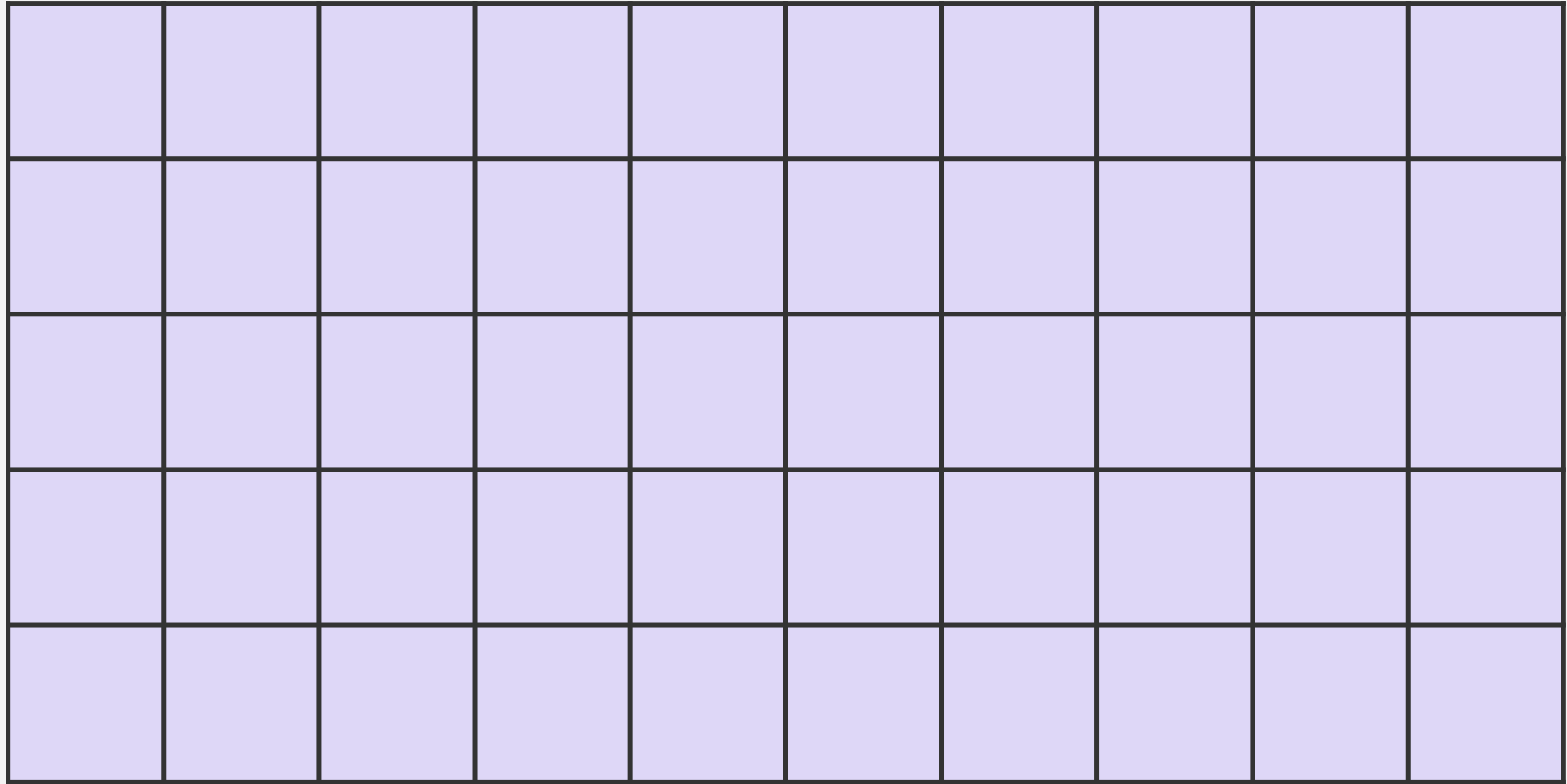
Cell

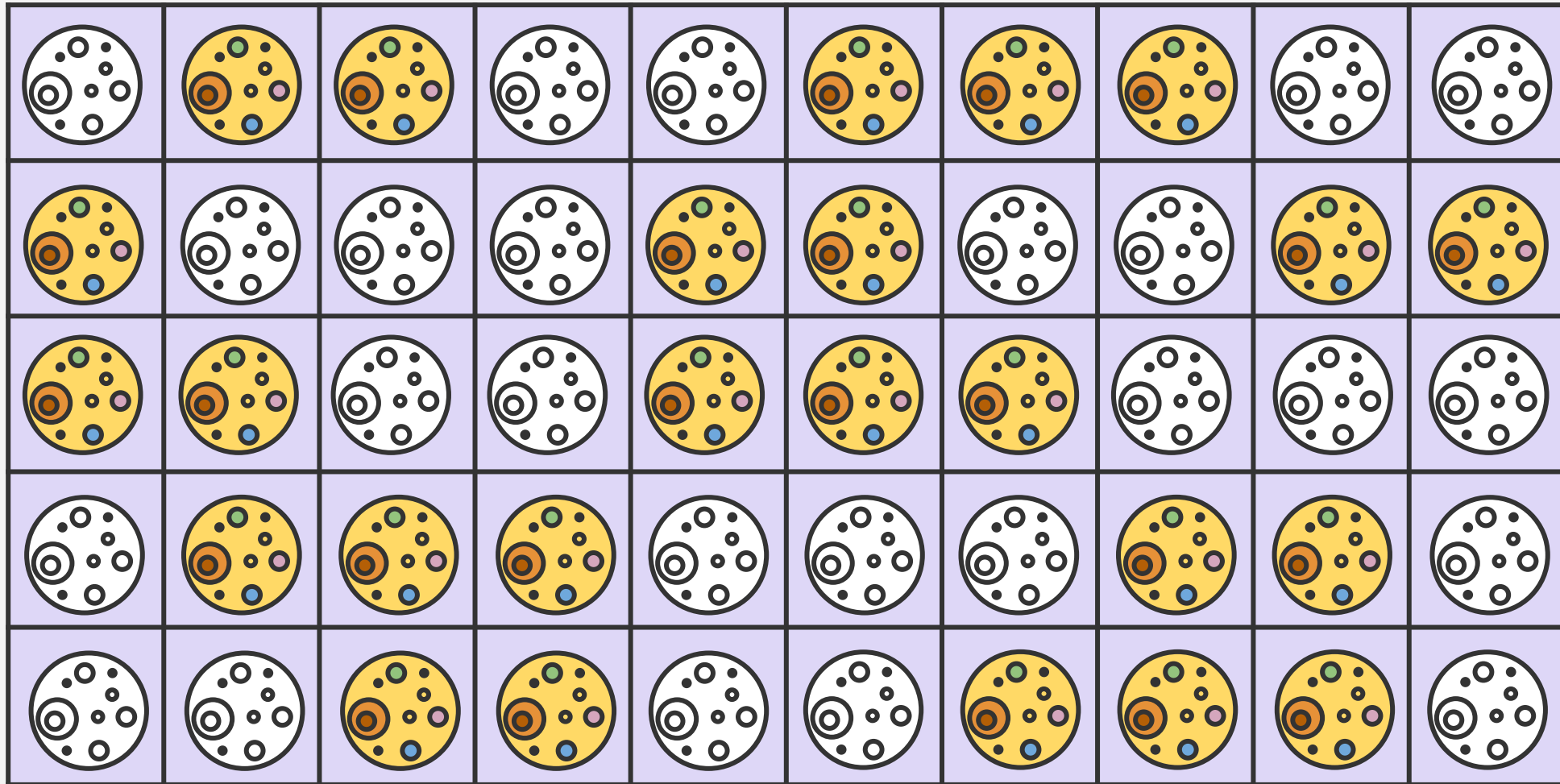


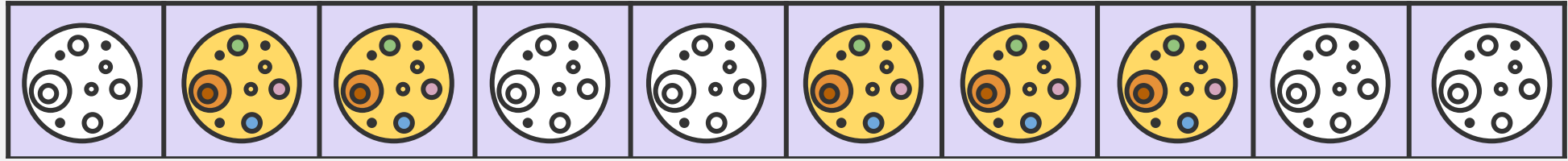
存活



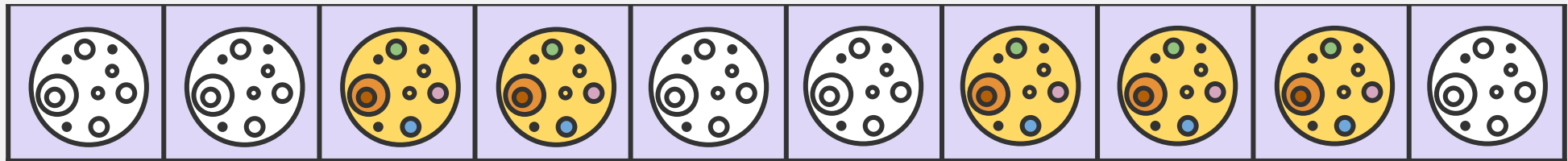
死亡







x, y 座標 到底要放在 **Map** 還是放在 **Cell** 上 ?





How did I set partners?

Map

3 references

```
public void Init(bool[,] matrix)
{
    this.Width = matrix.GetLength(0);
    this.Height = matrix.GetLength(1);

    Matrix = new Cell[Width, Height];
    for (int y = 0; y < Height; y++)
    {
        for (int x = 0; x < Width; x++)
        {
            Matrix[x, y] = new Cell
            {
                Status = matrix[x, y]
            };
        }
    }

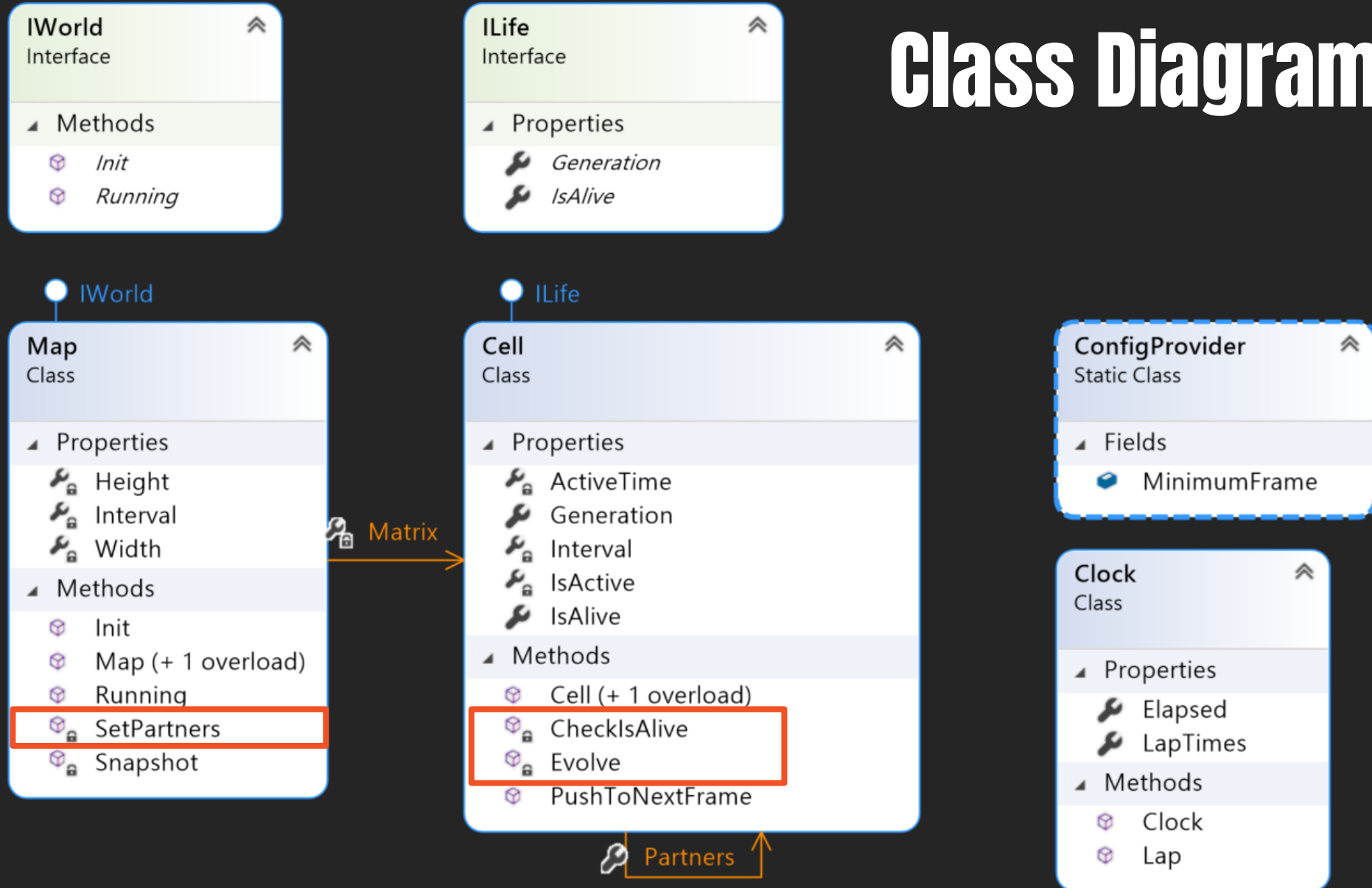
    SetPartners();
}
```

```
private void SetPartners()
{
    for (int y = 0; y < Height; y++)
    {
        for (int x = 0; x < Width; x++)
        {
            var partners = new Cell[3, 3];
            for (int ay = 0; ay < 3; ay++)
            {
                for (int ax = 0; ax < 3; ax++)
                {
                    int cx = x - 1 + ax;
                    int cy = y - 1 + ay;

                    partners[ax, ay] = new Cell();

                    if (cx < 0) partners[ax, ay].Status = false;
                    else if (cy < 0) partners[ax, ay].Status = false;
                    else if (cx >= Matrix.GetLength(0)) partners[ax, ay].Status = false;
                    else if (cy >= Matrix.GetLength(1)) partners[ax, ay].Status = false;
                    else partners[ax, ay] = Matrix[cx, cy];
                }
            }
            Matrix[x, y].Partners = partners;
        }
    }
}
```

Class Diagram

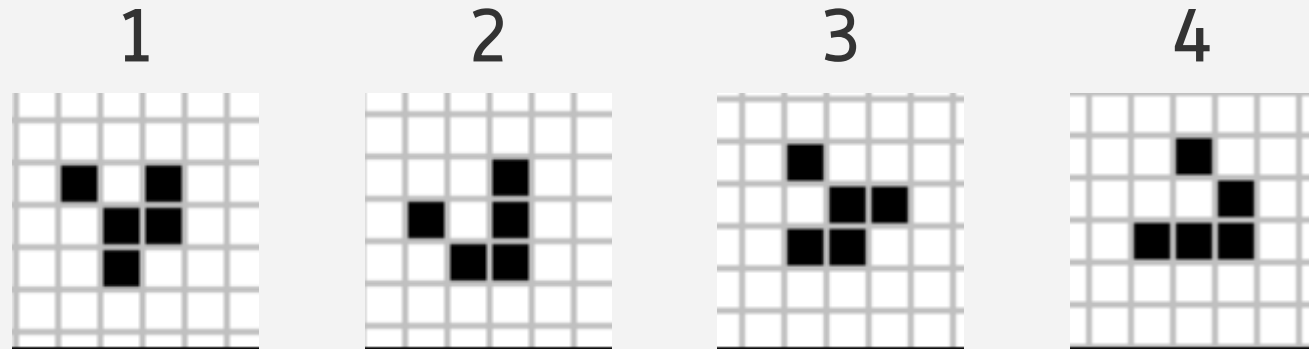
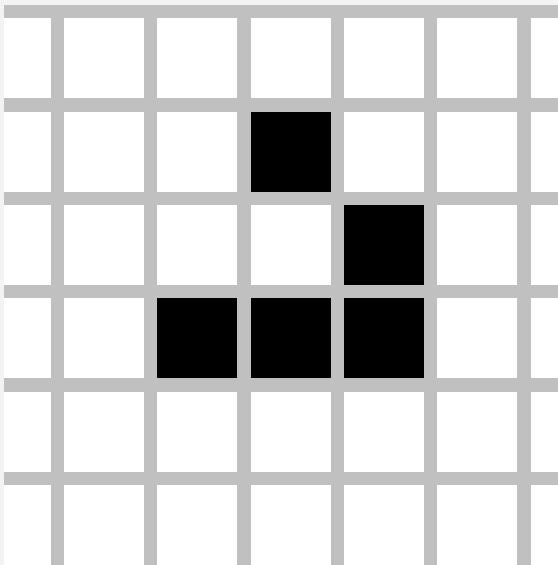


Milestone 3

Different cycles

Glider 滑翔機

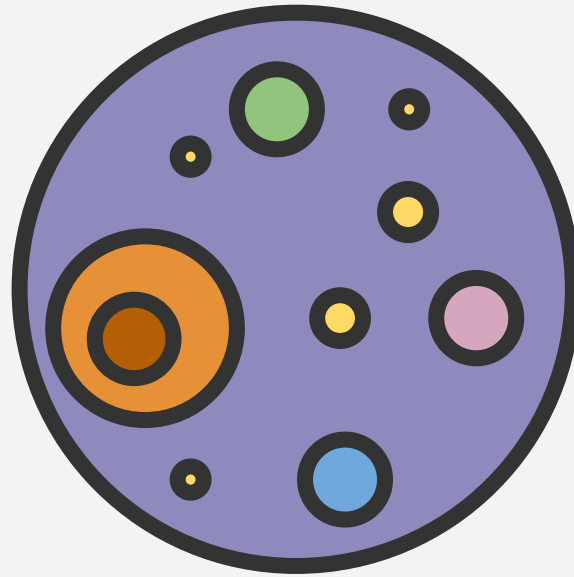
會移動的振盪狀態



~~回合制：同時間刷新整個畫面~~

1. 每個細胞有自己演化的週期時間
2. 世界有自己刷新畫面的時間

Conceiving # 1 - thread & timer

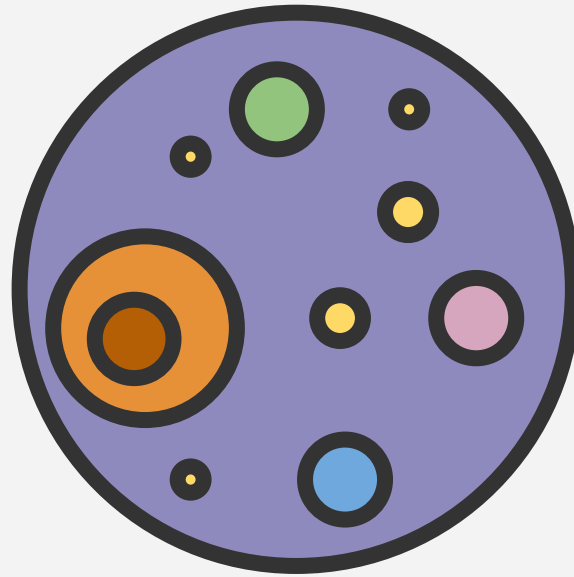


Interval : 30ms

AutoReset: true



Conceiving # 1 - thread & timer



Interval : 30ms

AutoReset: true



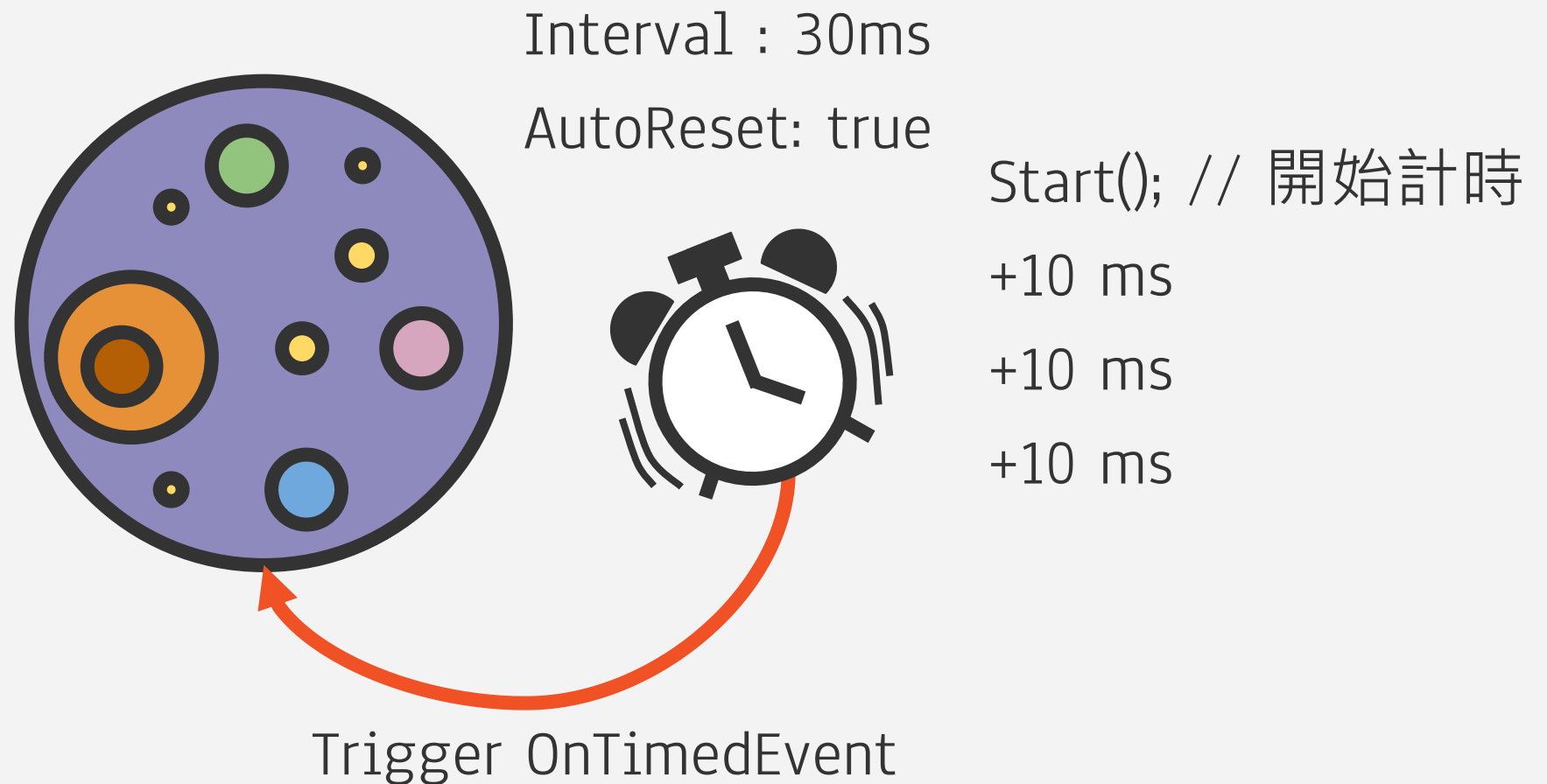
Start(); // 開始計時

+10 ms

+10 ms

+10 ms

Conceiving # 1 - thread & timer



Conceiving # 1 - thread & timer

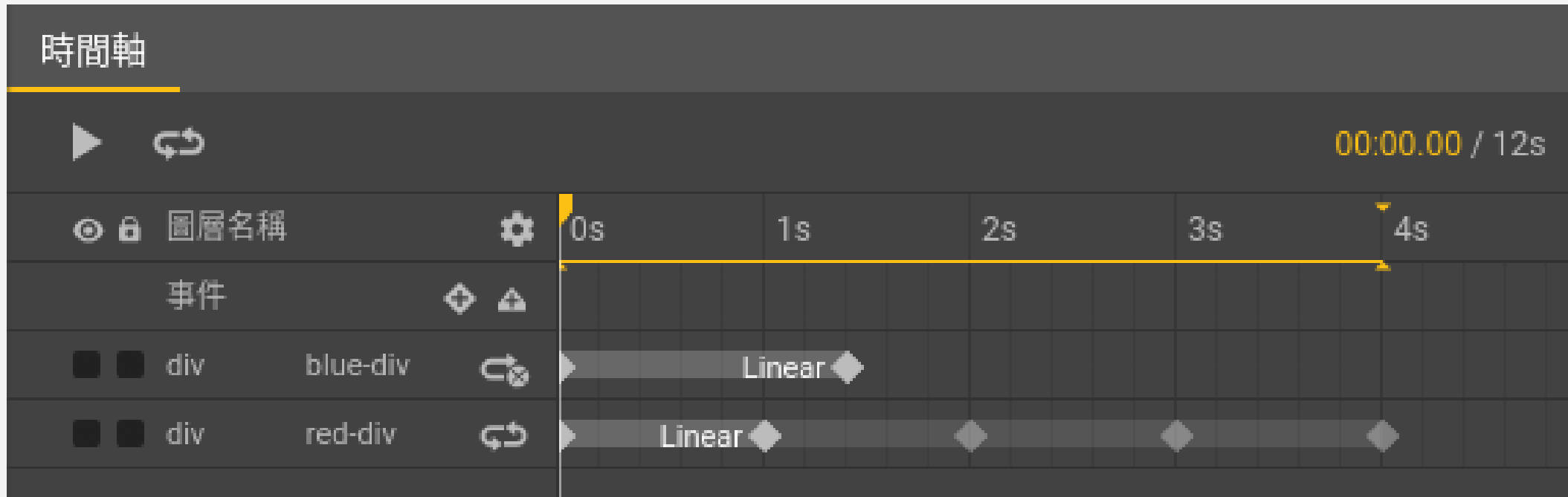
System.Timers.Timer

1. 支援 ThreadPool
2. 事件會依 Interval 屬性定義的間隔定期引發

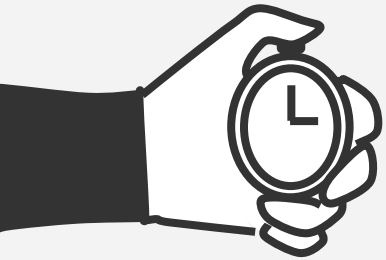
缺點

1. 無法絕對精準地控制 thread 的時間
2. 也就是每過 1 分鐘就是現實世界的 1 分鐘，所以無法實作忽略真實時間的 realtime mode

Conceiving # 2 - producing animation



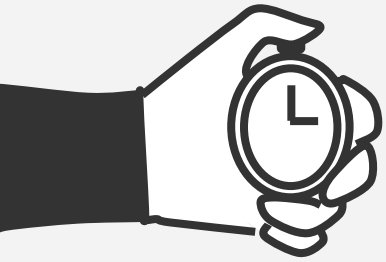
[illegible]



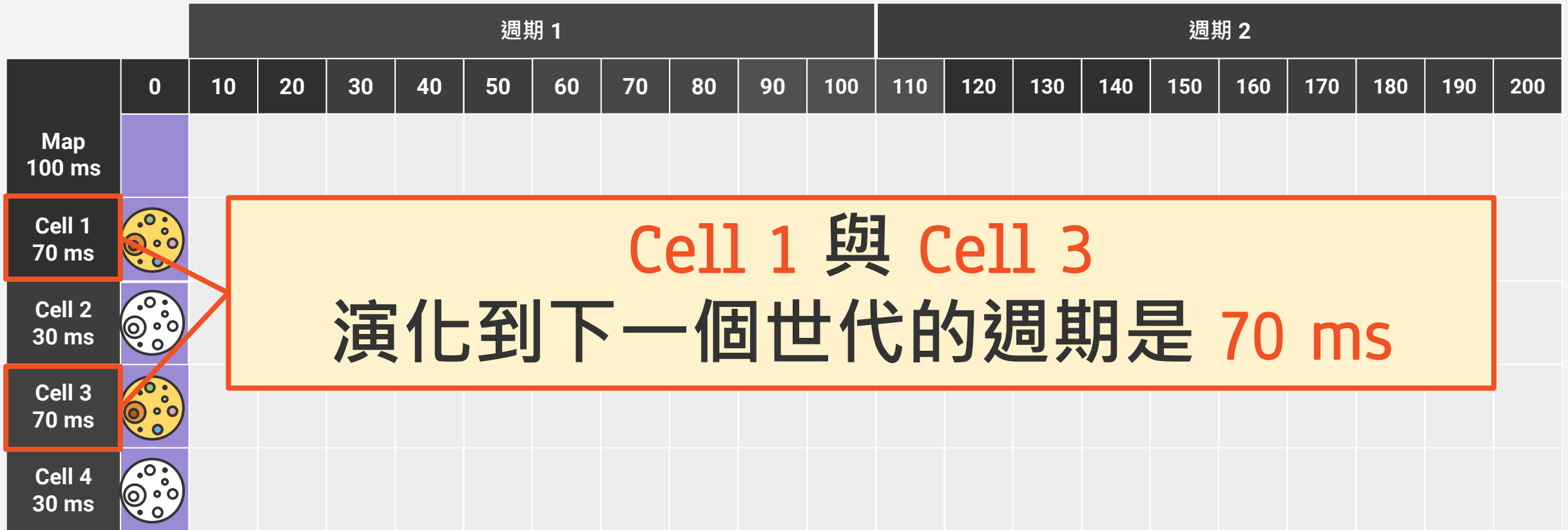
0 time / 0 ms

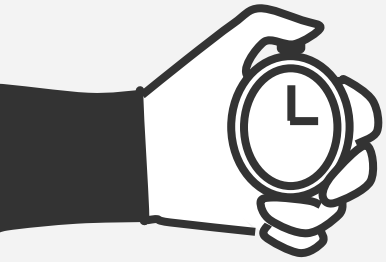
		週期 1										週期 2									
	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
Map 100 ms																					
Cell 1 70 ms																					
Cell 2 30 ms																					
Cell 3 70 ms																					
Cell 4 30 ms																					

整個 Map 被 snapshot 的週期是 100 ms



0 time / 0 ms



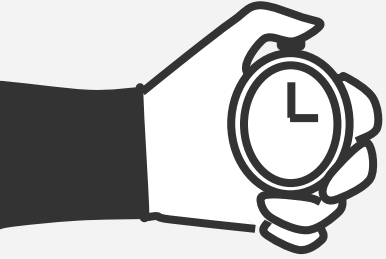


0 time / 0 ms

		週期 1										週期 2										
		0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
Map	100 ms																					
Cell 1	70 ms																					
Cell 2	30 ms																					
Cell 3	70 ms																					
Cell 4	30 ms																					

Cell 2 與 Cell 4
演化到下一個世代的週期是 30 ms

Cell 2 與 Cell 4
演化到下一個世代的週期是 30 ms

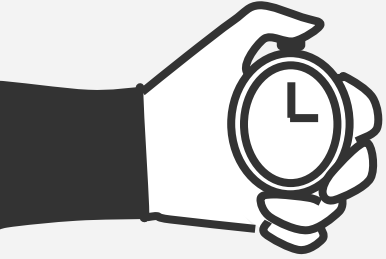


0 time / 0 ms





		週期 1										週期 2									
	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
Map 100 ms																					
Cell 1 70 ms																					
Cell 2 30 ms																					
Cell 3 70 ms																					
Cell 4 30 ms																					

Clock 會紀錄按了多少下 (times)
代表多少時間 (ms) 被推移過去了

Clock 會紀錄按了多少下 (times)
代表多少時間 (ms) 被推移過去了







0 time / 0 ms

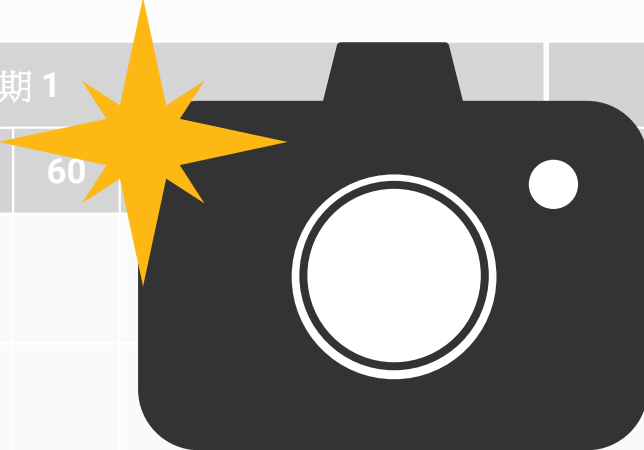
		週期 1										週期 2									
	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
Map 100 ms																					
Cell 1 70 ms																					
Cell 2 30 ms																					
Cell 3 70 ms																					
Cell 4 30 ms																					

Map 初始化並產生第一代細胞圖

Map 初始化並產生第一代細胞圖

[illegible]

		週期 1								週期 2							
	0	10	20	30	40	50	60		120	130	140	150	160	170	180	190	200
Map 100 ms																	
Cell 1 70 ms																	
Cell 2 30 ms																	
Cell 3 70 ms																	
Cell 4 30 ms																	



Snapshot!

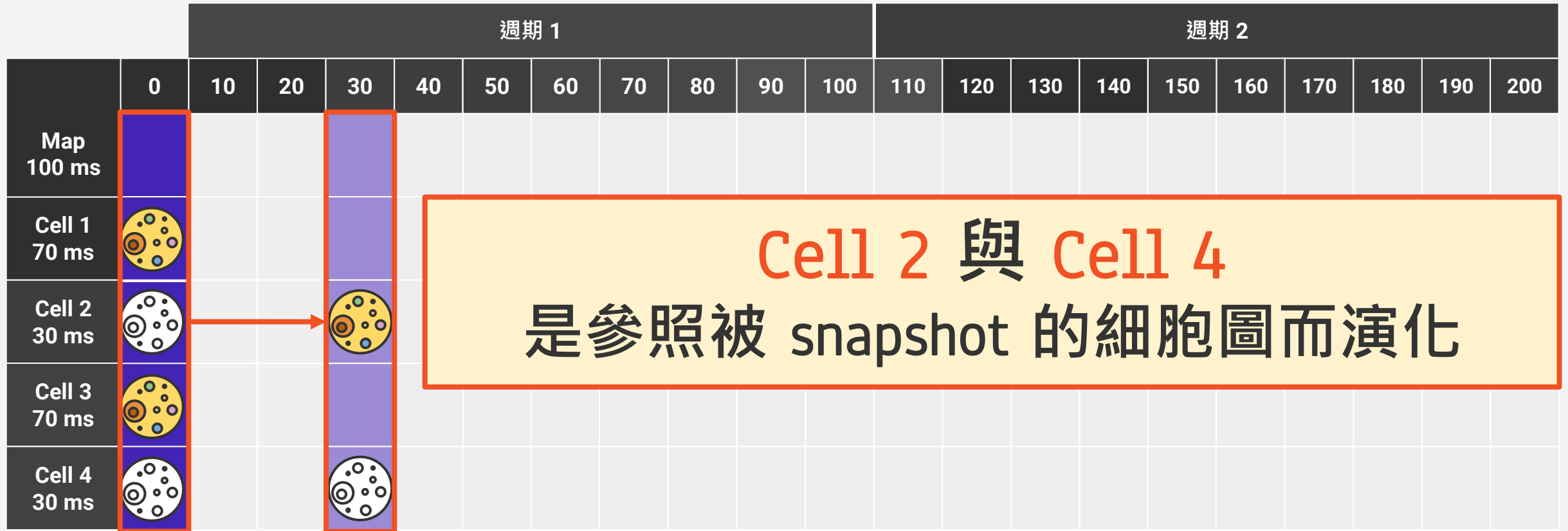
[illegible]

[illegible]

[illegible]



[illegible]

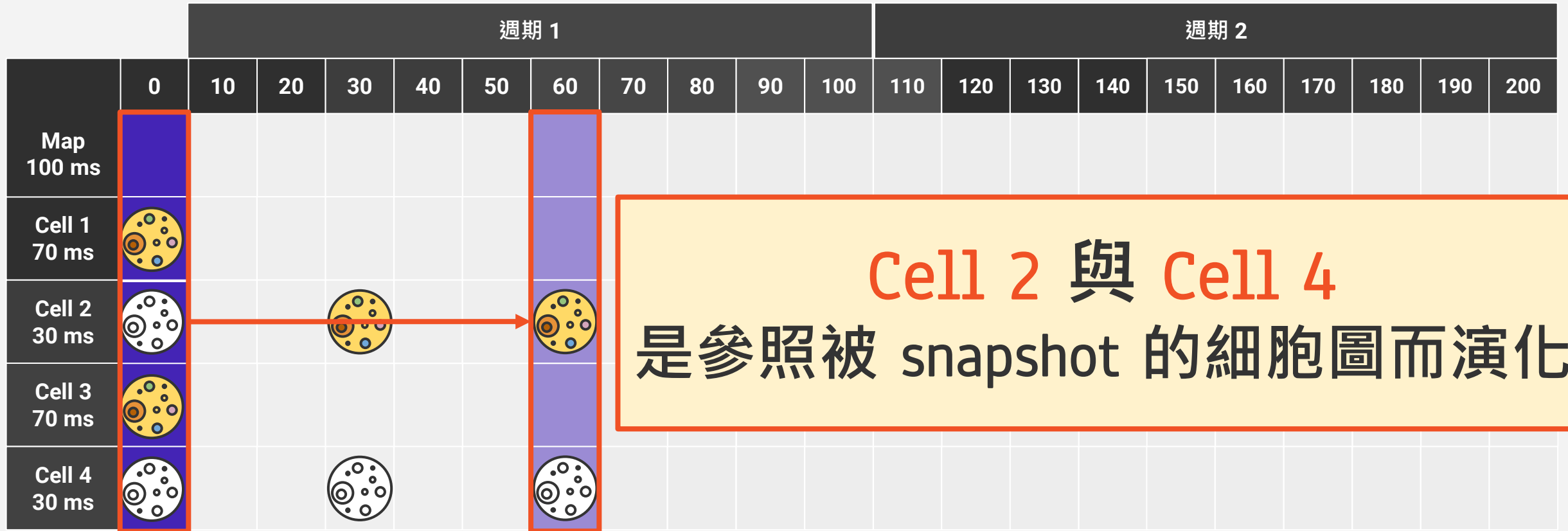


[illegible]

[illegible]

[illegible]

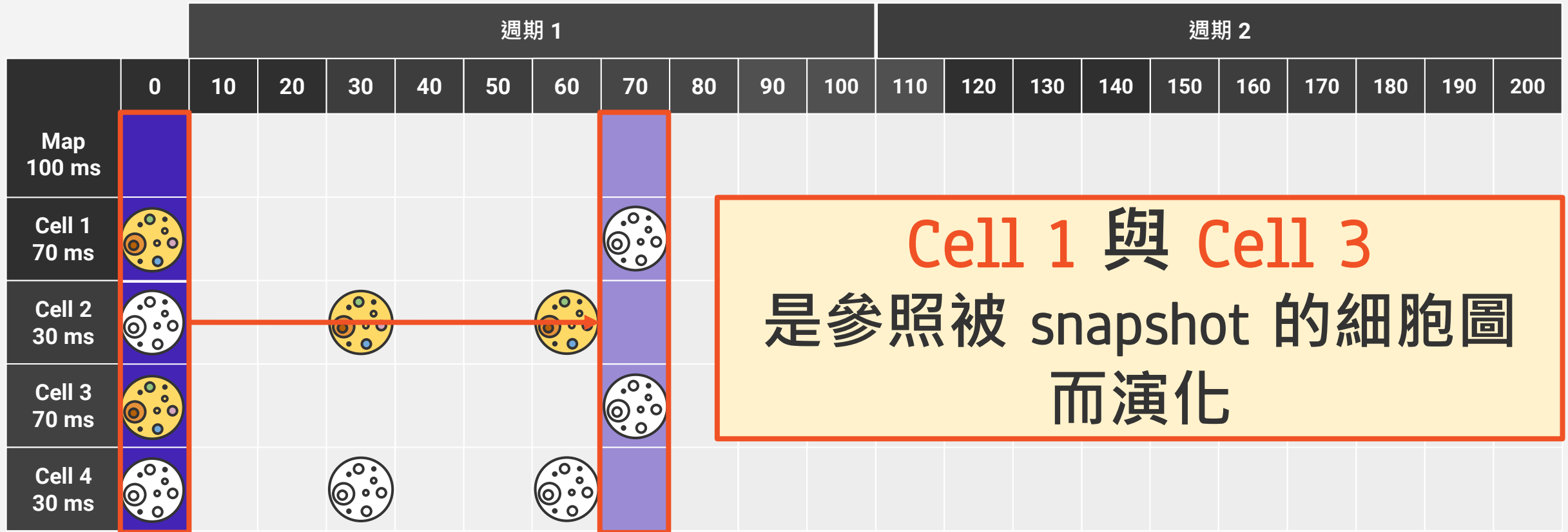
[illegible]





[illegible]

[illegible]

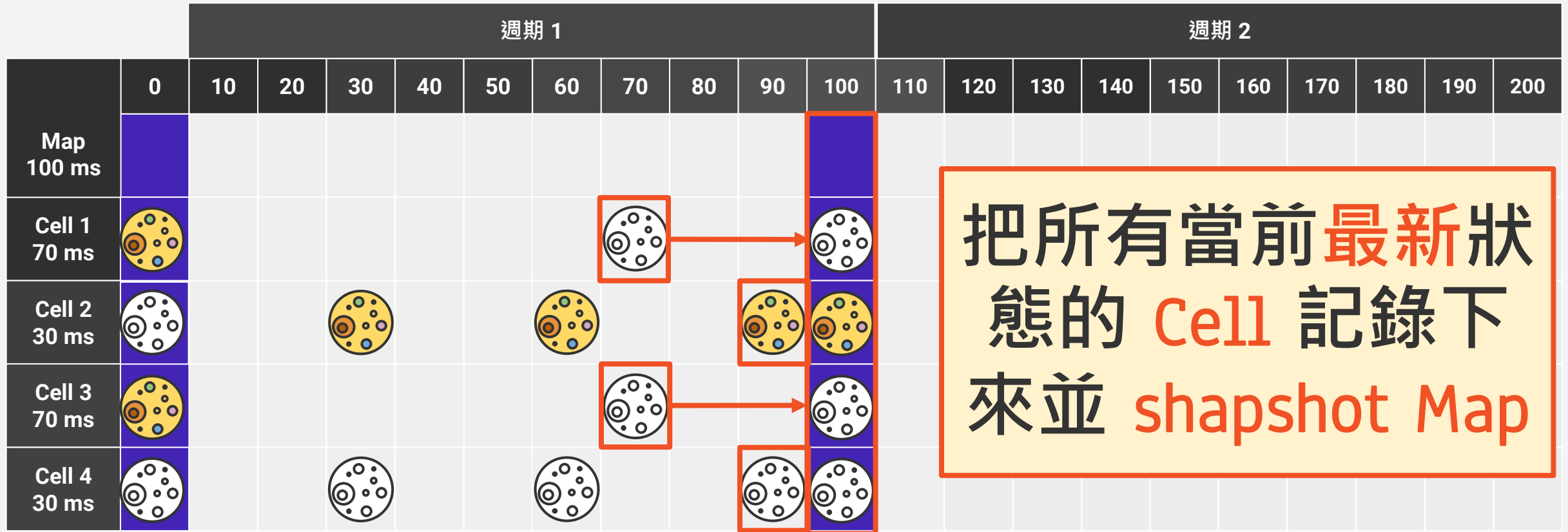


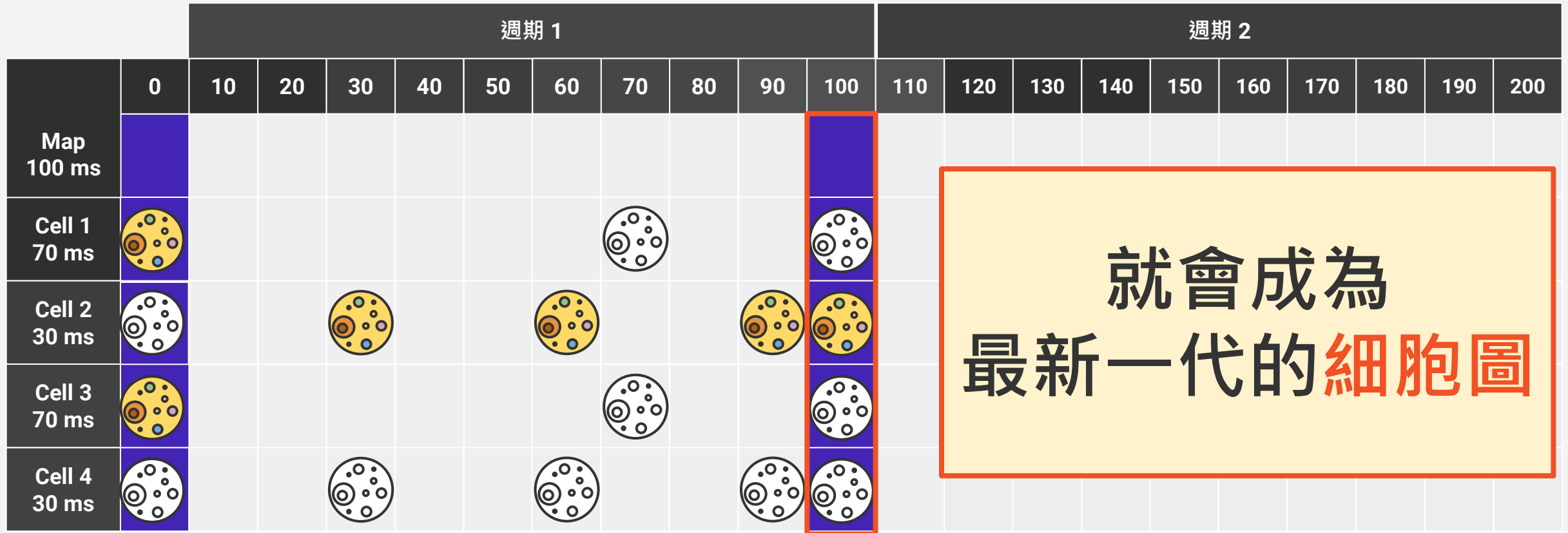
[illegible]

[illegible]























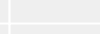





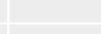

[illegible]





就會成為
最新一代的細胞圖



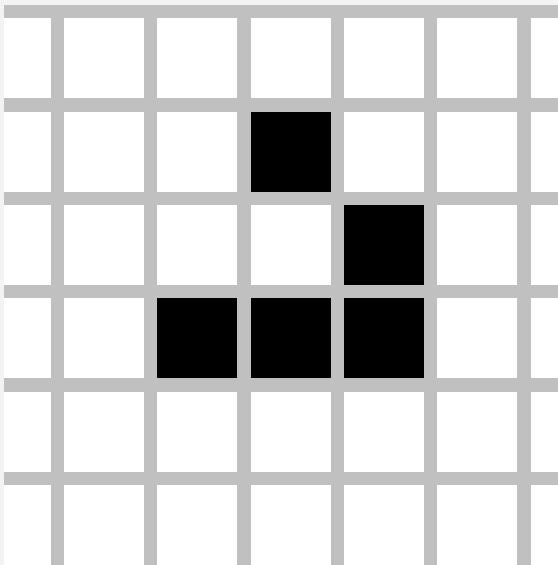
		週期 1										週期 2									
	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
Map 100 ms																					
Cell 1 70 ms																					
Cell 2 30 ms																					
Cell 3 70 ms																					
Cell 4 30 ms																					

Class Diagram

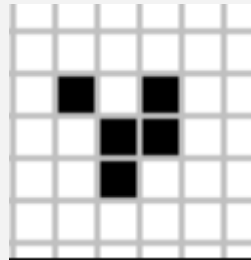


Glider 滑翔機

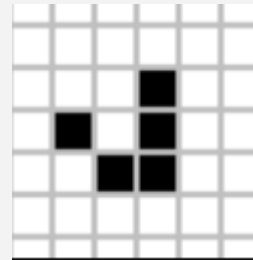
會移動的振盪狀態



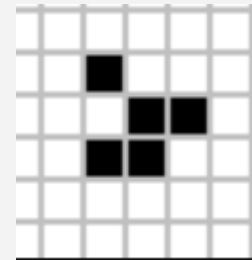
1



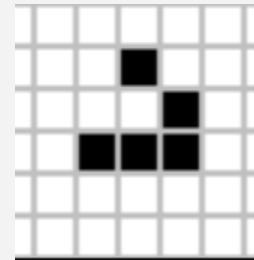
2



3

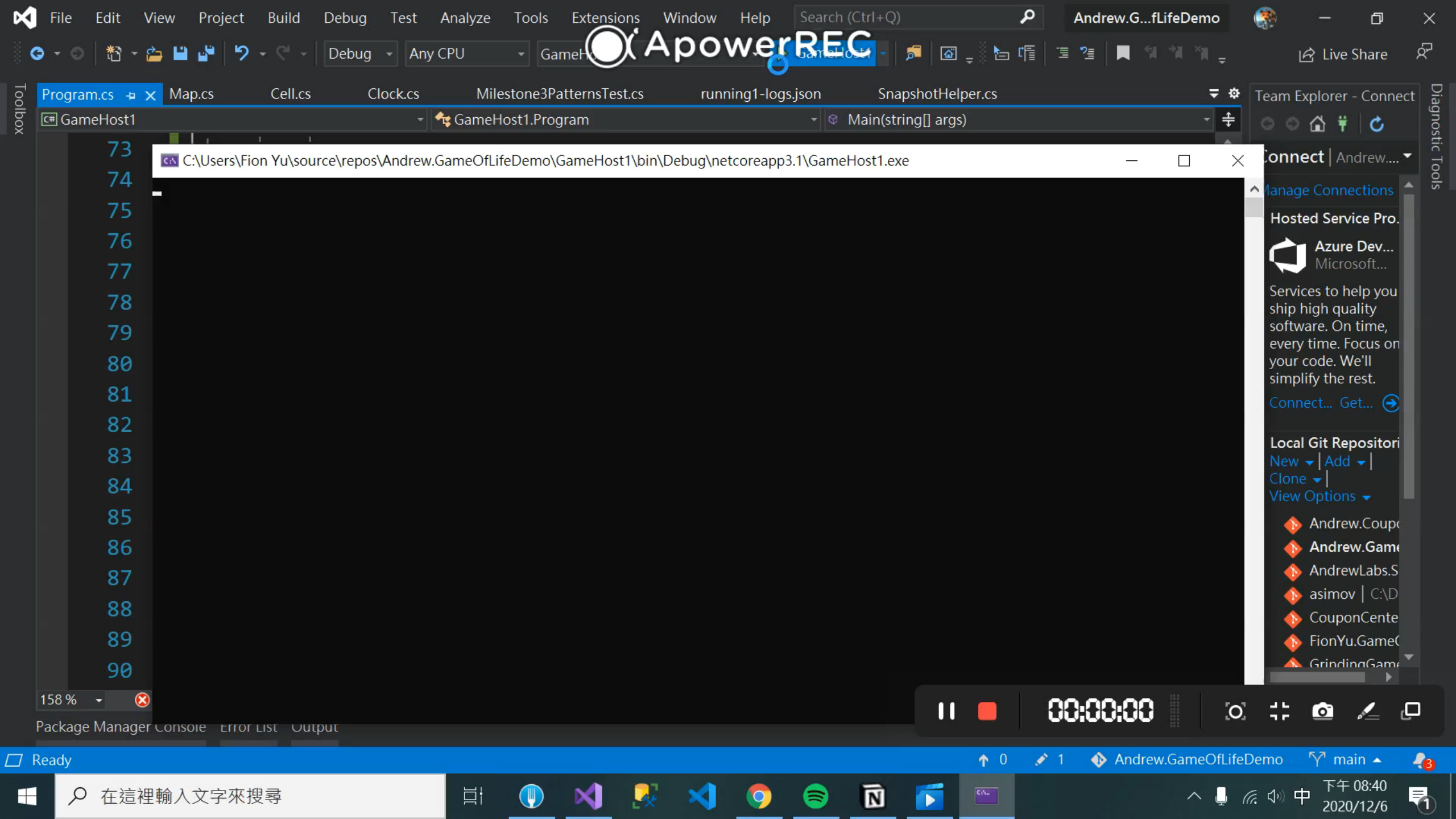


4



最終會成為

- a. 穩定狀態
- b. 振盪狀態
- c. 會移動的振盪狀態



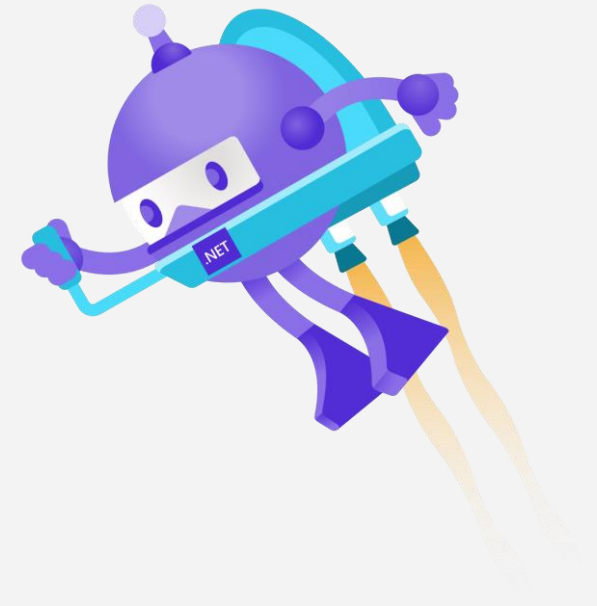
Demo Video

<https://youtu.be/8vikKrVRkvU>

02

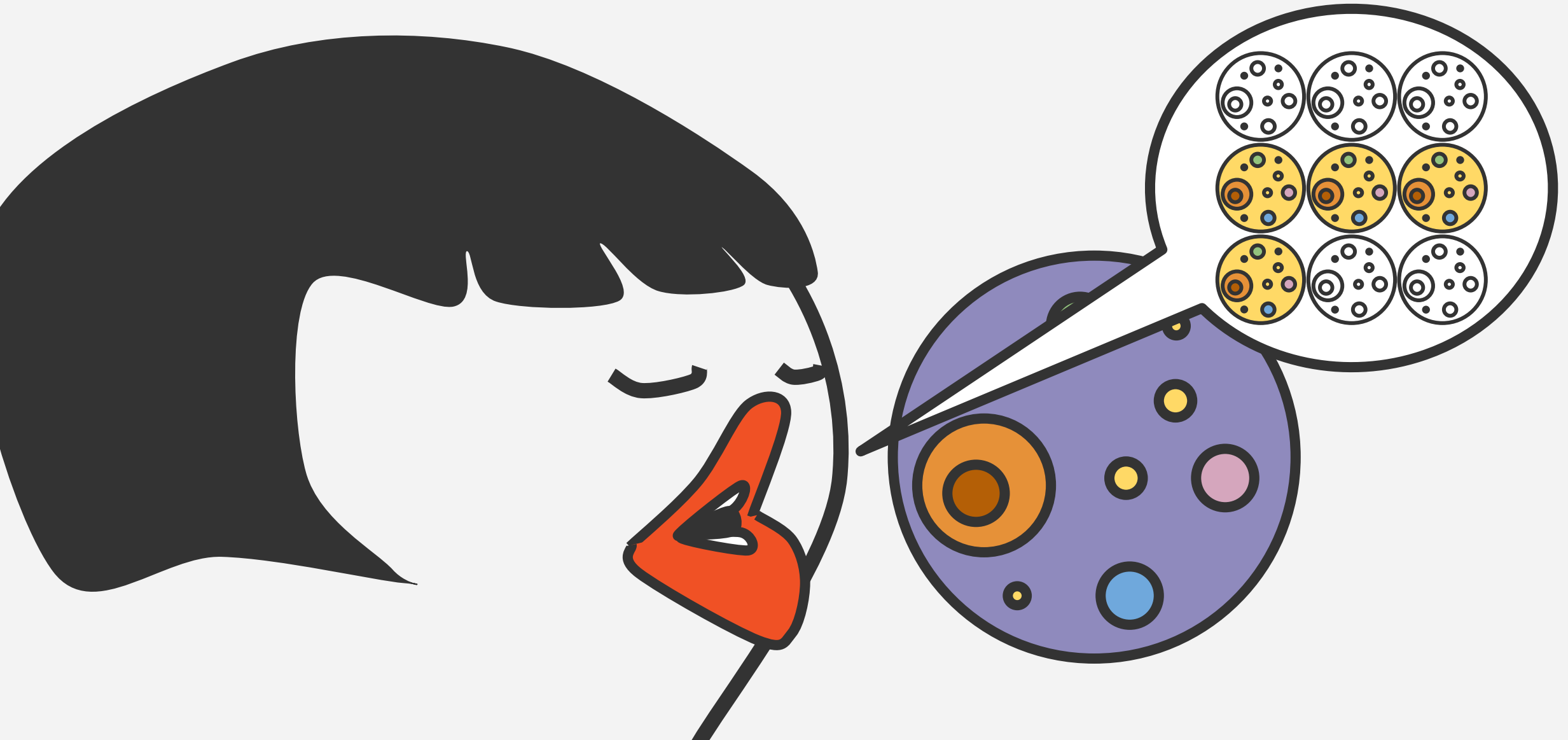
What did I learn from this homework?

刻意練習之下學到什麼？



How to distinct cells' vision

#1 Tell cells who are your **partners**



#1 Tell cells who are your **partners**

Advantages

1. 可以不用透過反查 Map 得知所有 partners 的細節
2. 限制 cells 的視野

Disadvantages

當 Map 太大時會佔用太多記憶體

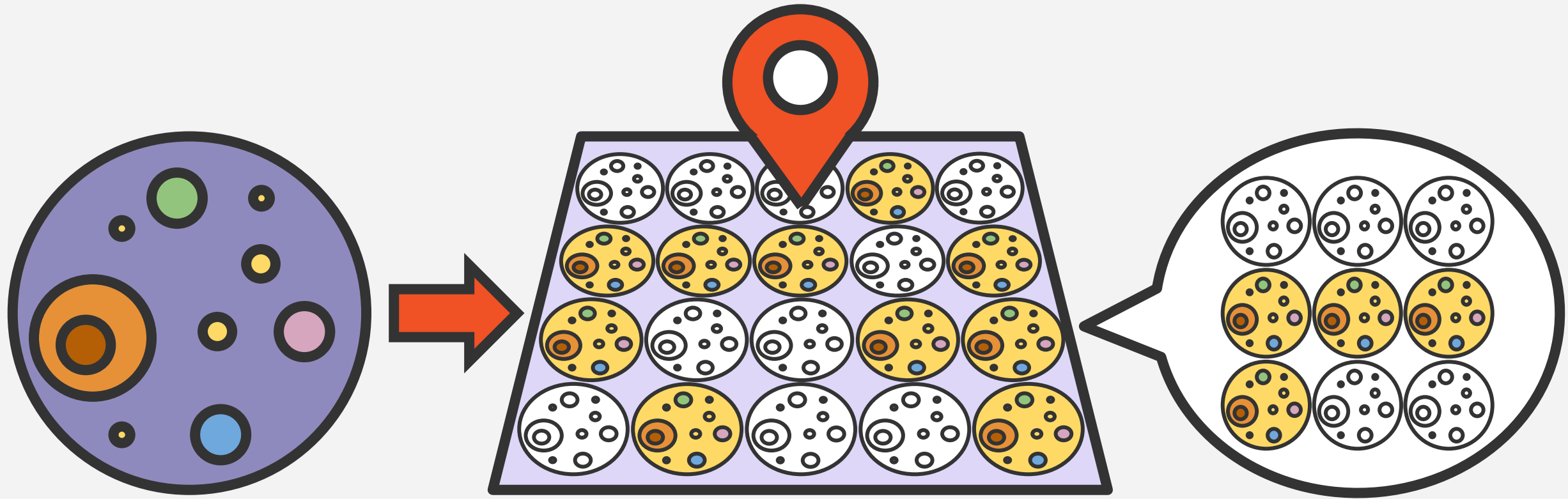
Ex. 地圖是 $500 * 20$

細胞總共有 10,000 個

每個細胞需要記住 8 個鄰居

= 80,000 個細胞

#2 Cells' Google Map



#2 Cells' Google Map

Advantages

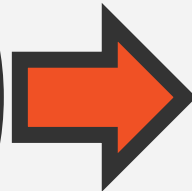
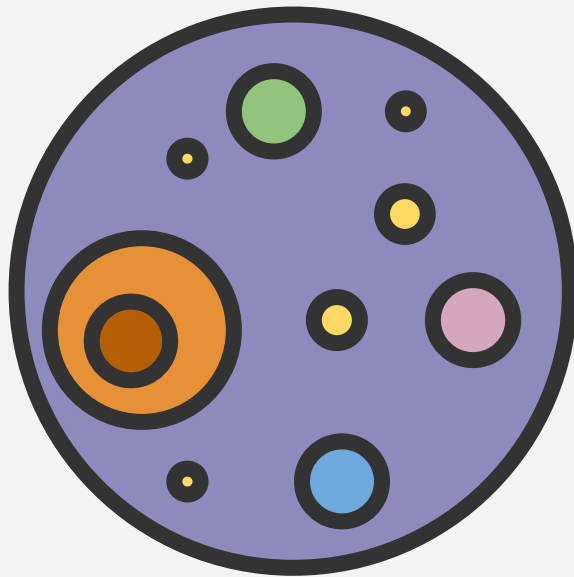
不用另外花記憶體去記 cells

Disadvantages

沒有限制 cell 的視野，有一定的機會可以看到其他 cell 的狀態

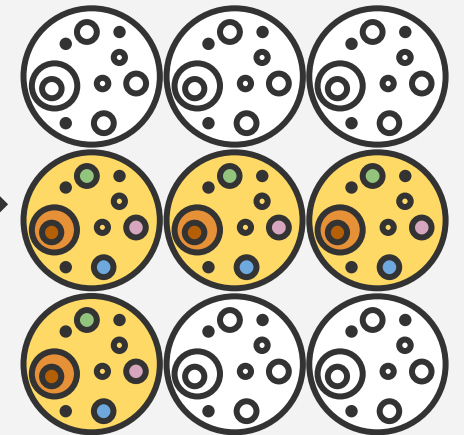
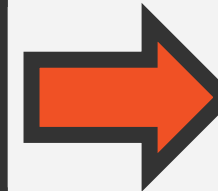
#3 Cells' position coordinate

$\{ 1, 1 \}$



Loop

$\{ 0, 0 \}$	$\{ 0, 1 \}$	$\{ 0, 2 \}$
$\{ 1, 0 \}$	$\{ 1, 1 \}$	$\{ 1, 2 \}$
$\{ 2, 0 \}$	$\{ 2, 1 \}$	$\{ 2, 2 \}$



#3 Cells' position coordinate

Advantages

不用另外花記憶體去記 cells

Disadvantages

沒有限制 cell 的視野，有一定的機會可以看到其他 cell 的狀態

**God's Perspective is
not a good practice**

God's **Perspective**

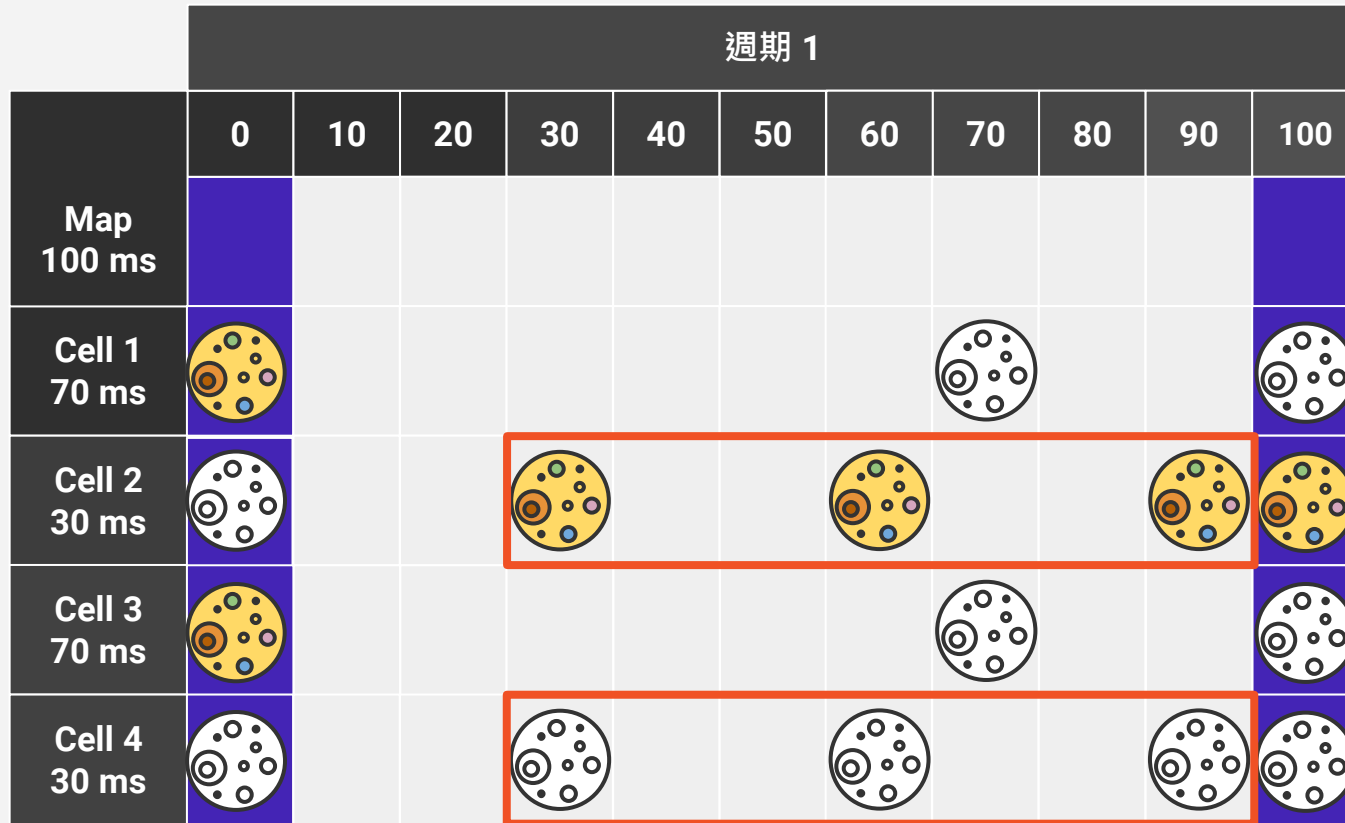
我就愛

上帝

視角

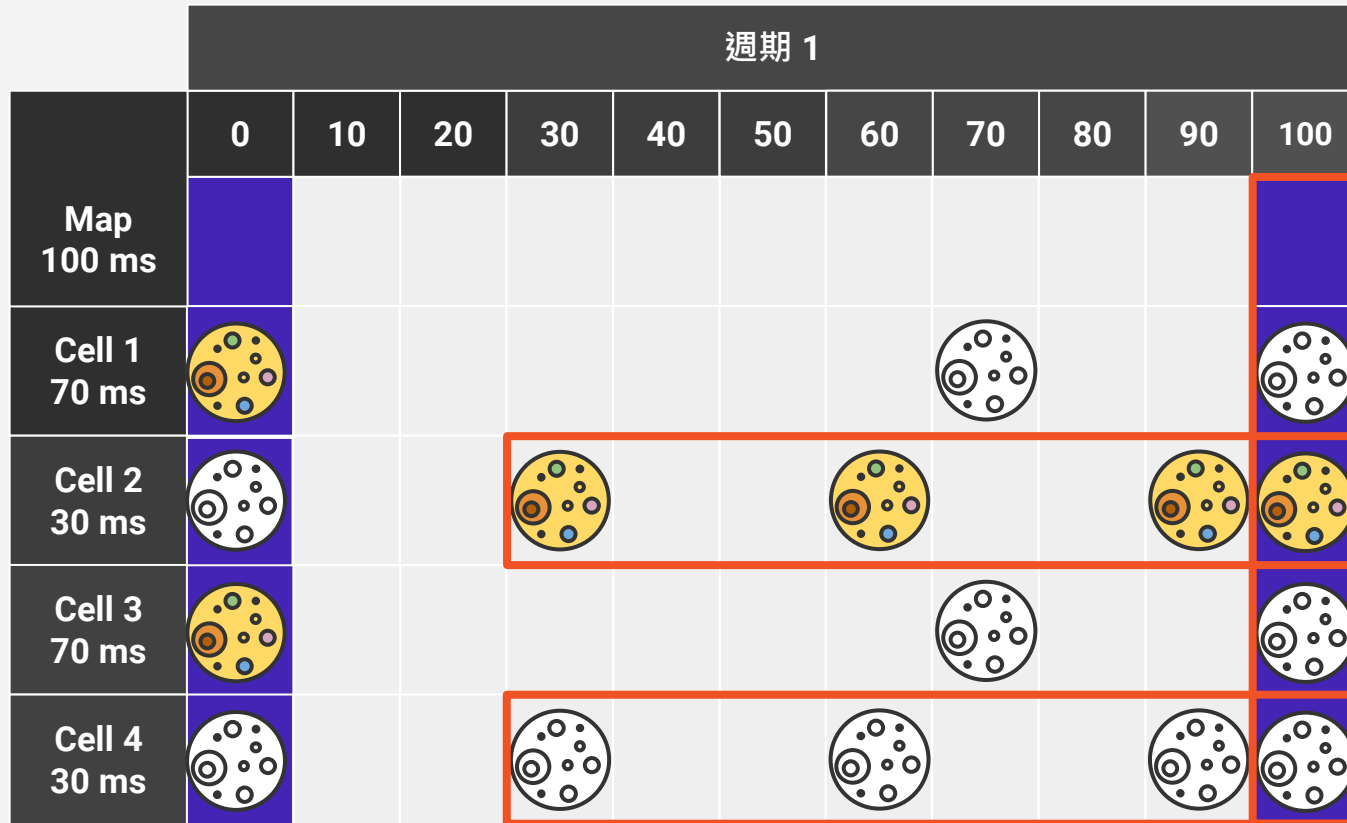


God's Perspective



Cell 的生命週期短於 Map snapshot 時，在每個細胞圖之間演化結果都會是一樣的

God's Perspective



Map 在 snapshot 時 Cell 才去演化，在這之前 Cell 都不會有任何行動。

Cell 知道 Map snapshot 的資訊(ex. snapshot 時間)

God's Perspective

Cell 不可以
知道 Map
snapshot 的
時間資訊！



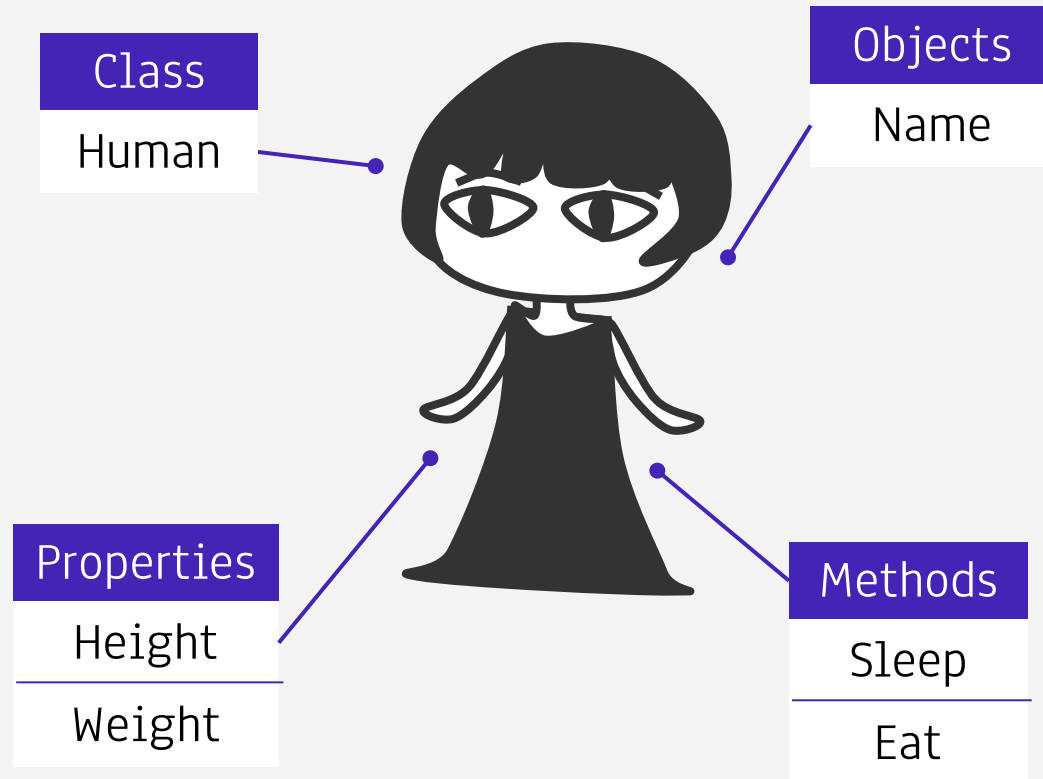


God's Perspective



寫 code 知道太多不該知道的細節，哪天人家改版就會有相容性的問題。

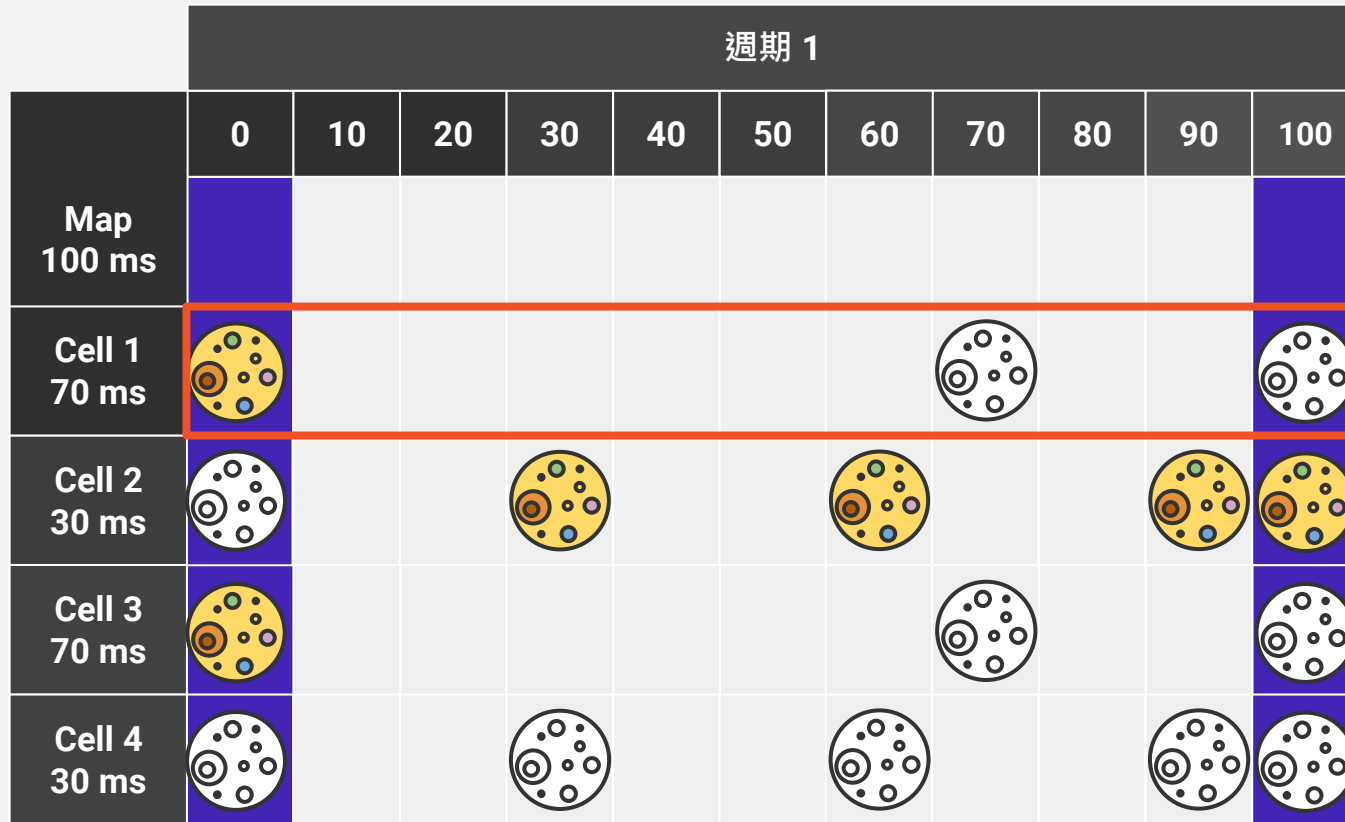
OOP



- 正確的抽象化
- 正確的對應現實世界的需求變化
- 正確的封裝並避免錯誤的存取

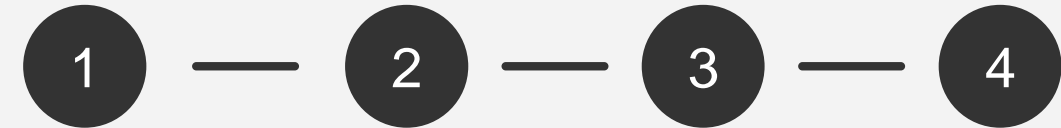
都是 **OOP** 化考量的重點

Thinking with **timeline**

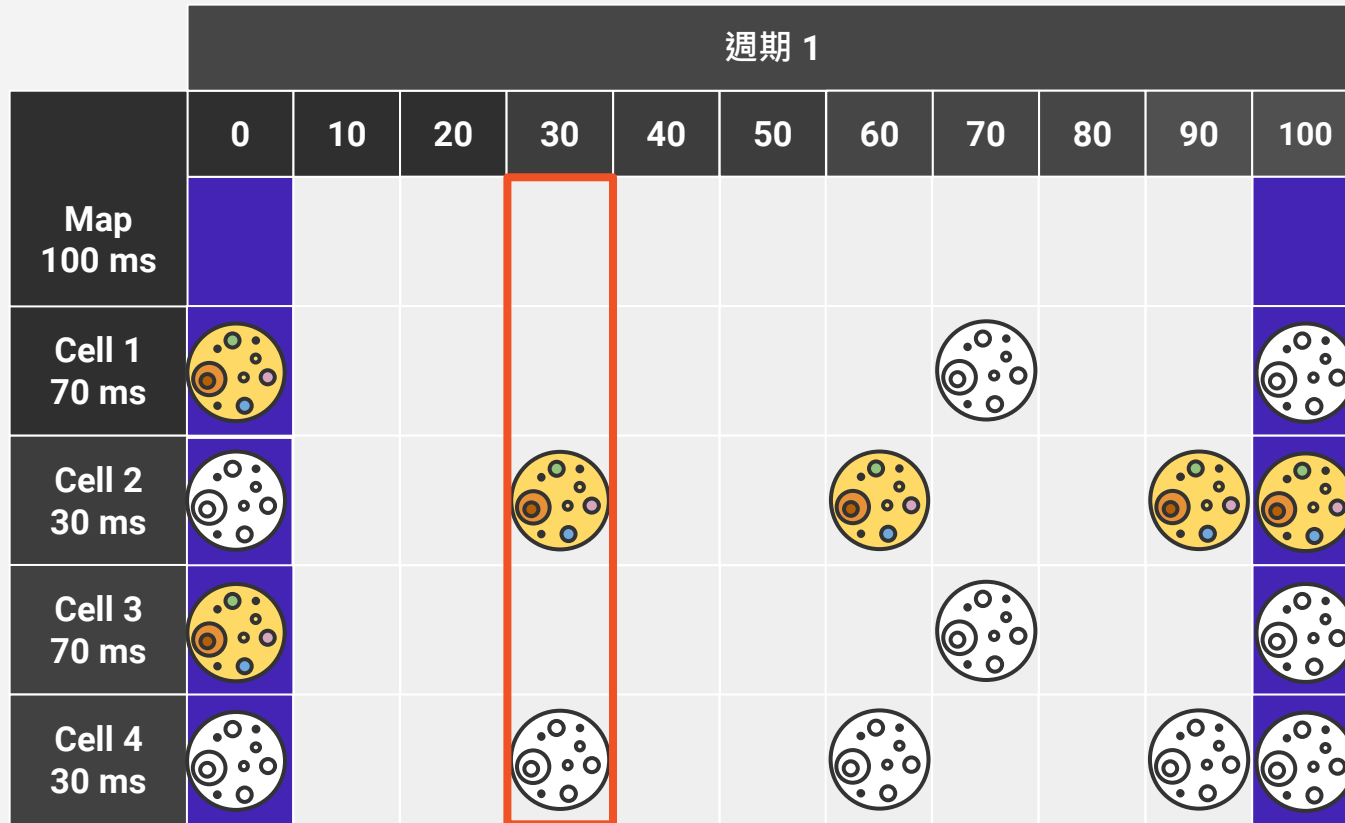


直線思考方式

順序性，只專注一個細胞的演化狀態

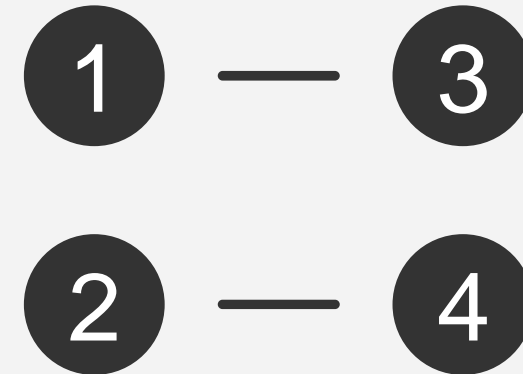


Thinking with timeline



跳躍性思考方式

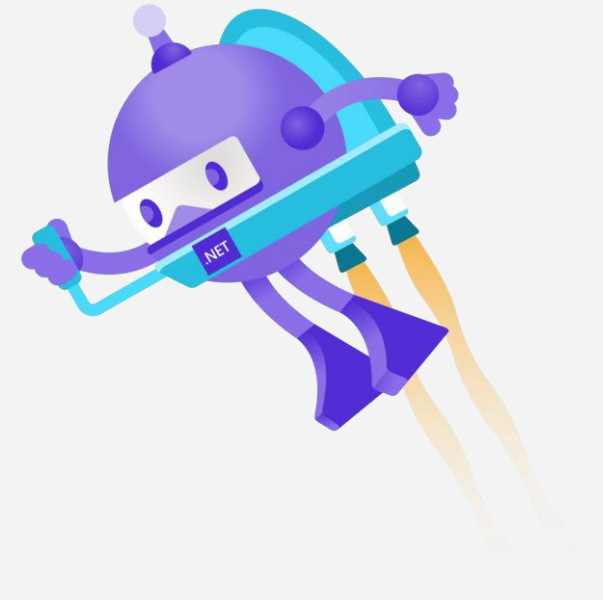
同時處理不同細胞的狀態



03

Conclusion

結論



我就爛!



Conclusion

- 01** | **OOP** 封裝：合理的抽象化與職責分離
- 02** | 設計時間序的思考方式：從直線到跳躍
- 03** | 刻意練習：面對同一個問題與挑戰，有更多不同的解法

All Images Maps News Videos More Settings Tools

About 2,780,000 results (0.40 seconds)

zh.wikipedia.org › zh-tw › 康威生命... Translate this page
康威生命遊戲- 維基百科，自由的百科全書 - Wikipedia




康威生命遊戲（英語：Conway's Game of Life），又稱康威生命棋，是英國數學家約翰·何頓·康威在1970年發明的細胞自動機。它最初於1970年10月在《科學美國》...
規則 · 概述 · 穩定狀態 · 振盪狀態

pansci.asia › archives Translate this page
電腦裡的生命遊戲，等你挑戰讓生命無限延續！ - PanSci 泛科學
Mar 23, 2016 — 康威生命遊戲中的泛科學跑馬燈！遊戲規則。生命遊戲是由英國數學家約翰·何頓·康威（John Horton Conway）發明，刊登在 ...


ithelp.ithome.com.tw › articles Translate this page
康威生命遊戲- iT 邦幫忙::一起幫忙解決難題，拯救IT 人的一天
Oct 14, 2019 — 老實說，我很早就知道生命遊戲了，這再Emacs裡面就有，只是我一直記不住全名：我會突然想分享在這個主題，是因為下面這個影片：影片裡頭 ...

kknews.cc › 遊戲
用Python模擬生物繁衍進化（康威的生命遊戲） - 每日頭條
Oct 27, 2018 — 康威的生命遊戲Conways的生命遊戲是由JohnConway創建的蜂窩自動化方法。這個遊戲是以生物學為基礎創建的，但已應用於各種領域，如圖形 ...

Videos

- 【分形与混沌8】生命的源代码？康威生命游戏与复杂性探索
YouTube · 妈咪说MommyTalk
Jan 13, 2020
- 生命游戏：另一种计算机| Life game：Another type of computer | ChaosMuseum
YouTube · 混乱博物馆 ChaosMuseum
Oct 9, 2019
- 数字版生命游戏Alternative Game of Life with Numbers 演示视频






Conway's Game of Life (康威生命遊戲)



The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. It is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. [Wikipedia](#)

Genre: Zero-player game

People also search for [View 15+ more](#)

 Mineswe... Breakout Pac-Man Tic-tac-toe The Game of Life

Feedback

Easter Egg from Google

<https://youtu.be/5yaNK1L0BYE>