Graph-based wavefunction collapse

Specifikace zápočtového programu pro letní semestr 2023/24

Cíle

Cílem je implementovat WFC jako knihovnu, která dokáže obarvit libovolný orientovaný graf podle pravidel a distribucí jednotlivých barev. Tuto knihovnu lze použít například pro generování map pro roguelike hry (třeba v Unity), barvení grafů (algoritmus v nejhorším případě vyzkouší všechny možnosti, lze jím dokázat, že nějaký graf je k-obarvitelný), řešení sudoku.

Formáty vstupů

- graf zadaný jako seznam hran `A->B`, s možností interpretovat graf jako neorientovaný:

```
0 1
1 2
3 2
...
```

- pravidla zadaná ve formátu JSON, příklad pro obarvení čtyřmi barvami:

```
{
  "0": [ "1", "2", "3" ],
  "1": [ "0", "2", "3" ],
  "2": [ "0", "1", "3" ],
  "3": [ "0", "1", "2" ]
}
```

- formát výstupu, tady si ještě nejsem úplně jistý, určitě by šlo o dvojice `vrchol, barva`

Použití a ukázky funkčnosti

- uživatel zadá
- načtení grafu a pravidel ze souboru
- sudoku
- generování ve čtvercové síti
- barvení grafu

Možná vylepšení

- podpora populárních knihoven pro práci s grafy v C#

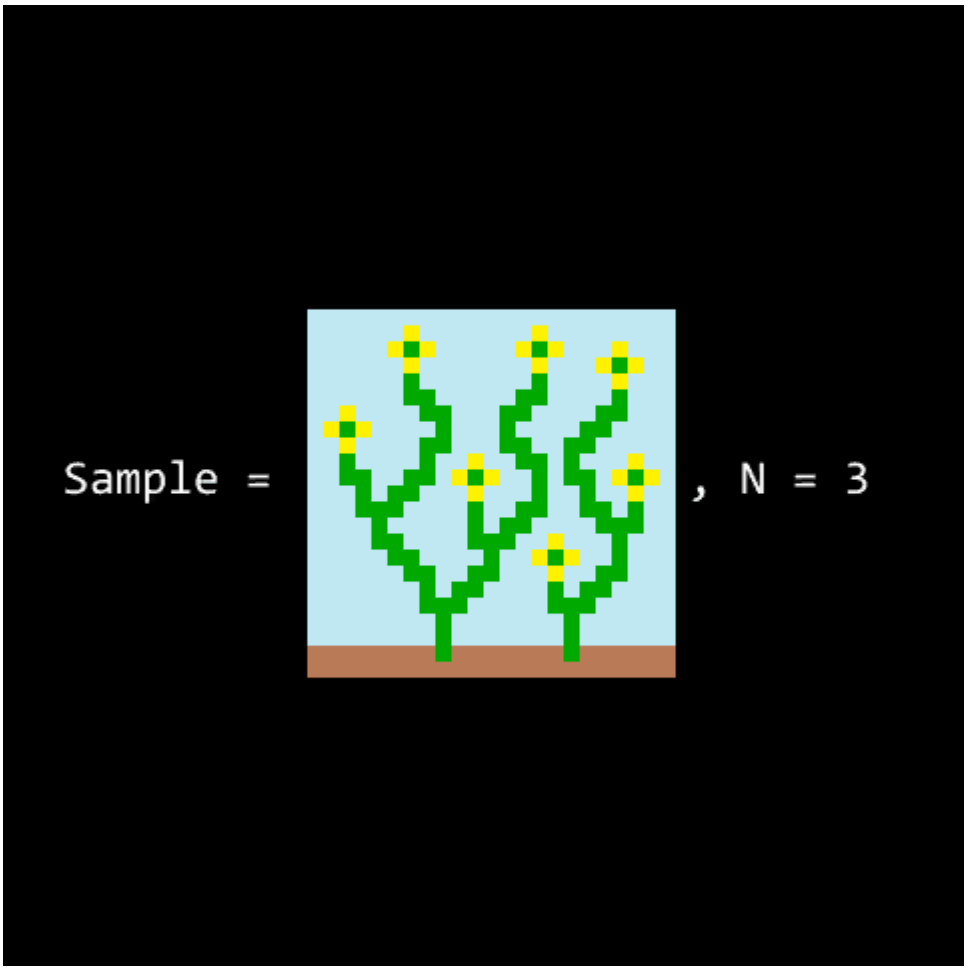
Co je to WFC?

WFC se nejčastěi používá v herním průmyslu pro procedurální generování obsahu, jako jsou mapy, terény nebo levely ve hrách. Inspiraci si bere z kvantové mechaniky - pracuje se superpozicemi buněk a kolapsem vlnových funkcí.

WFC začíná s několika buňkami, kde každá buňka může být v libovolném z několika možných stavů. Algoritmus postupně vybírá buňky a omezuje jejich stavy na základě pravidel sousedství, dokud nezůstane pro každou buňku jediný možný stav.

Pseudo algoritmus:

1. **Inicializace**: Každá buňka je nastavena na superpozici všech možných stavů.
2. **Kolaps**: Vybere se buňka s nejmenší entropií a její stav se zafixuje.
3. **Propagace**: Omezení se šíří do sousedních buněk, aktualizují se jejich možné stavy na základě pravidel sousedství.
4. **Opakování**: Proces kolapsu a propagace se opakuje, dokud nejsou všechny buňky v právě jendom stavu.



Znaky typické implementace WFC

Implementace WFC dostupné online se povětšinou omezují na čtvercové sítě a pravidla, která jsou symetrická. S trochou nadsázky bychom mohli říct, že standardní implementace umí řešit obarvování rovinných grafů. (Pro mě je nejlepší přemýšlet o WFC jako o obarvování grafů s

pravidly navíc.)

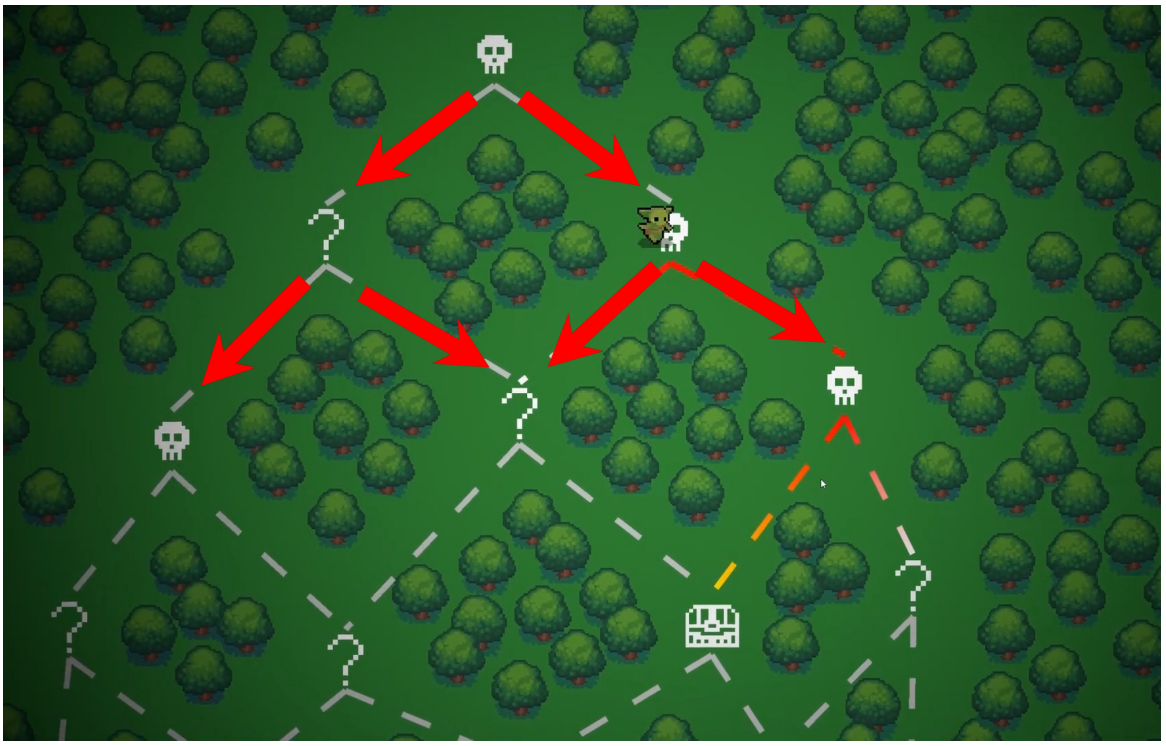
Pravidla jsou zadána jako `A-B`, to znamená, že `A` může sousedit s `B`.

Proč to nestačí?

Za dva roky studia na matfyzu jsem přišel na to, že ne všechny problémy se dají redukovat na čtvercovou síť.

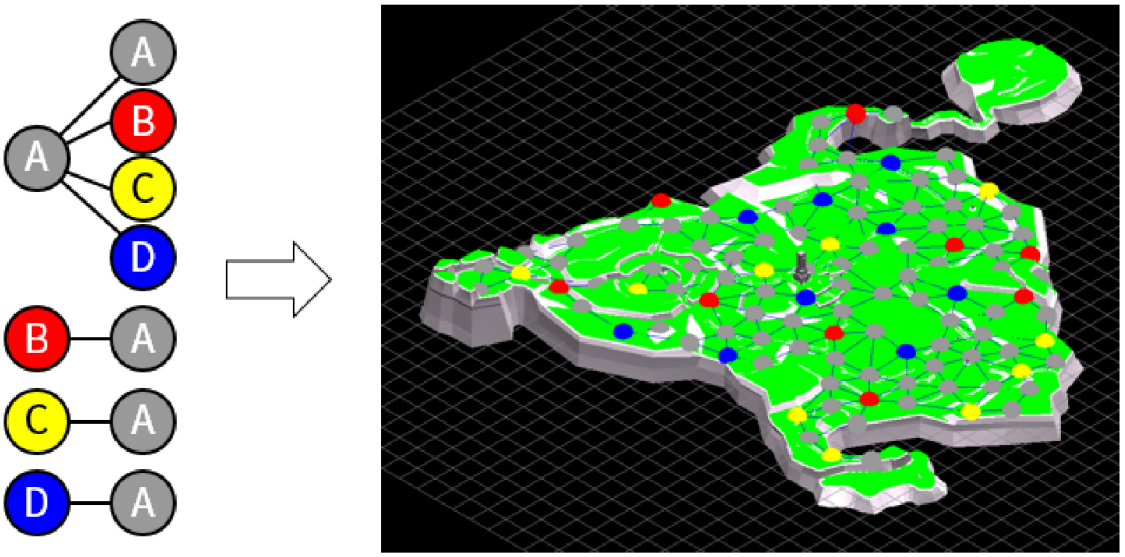
Problémy symetrie pravidel

Ve hře Peglin má hráč po každé bitvě na výběr mezi dalšími dvěma možnostmi. A protože se hráč nemůže vracet, jedná se o DAG - **orientovaný** graf.



Dejme tomu, že cheme, aby hráč po každým bossovi (💀) měl možnost doplnit zásoby (💰). Určitě ale nechceme, aby si hráč doplnil zásoby před bossem - pravidlo není symetrické. vyjádříme ho jako 💀 -> 💰, typická WFC s tímto nepočítá, nemáme možnost zadávat nesymetrická pravidla. (Některé implementace WFC vykoukají pravidla z příkladového obrázku, taková pravidla jsou inherentně symetrická.)

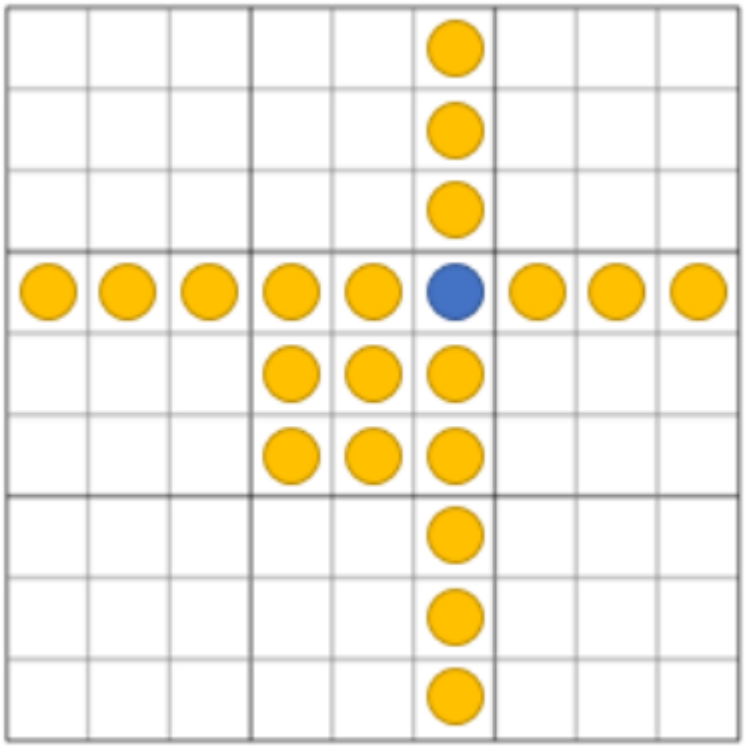
Složitejší konstrukce herních levelů mohou vypadat třeba takto:

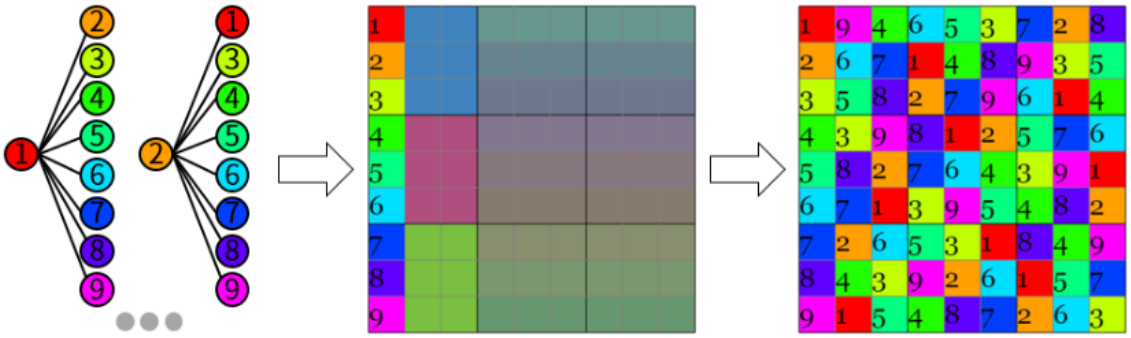


Problémy sousedství

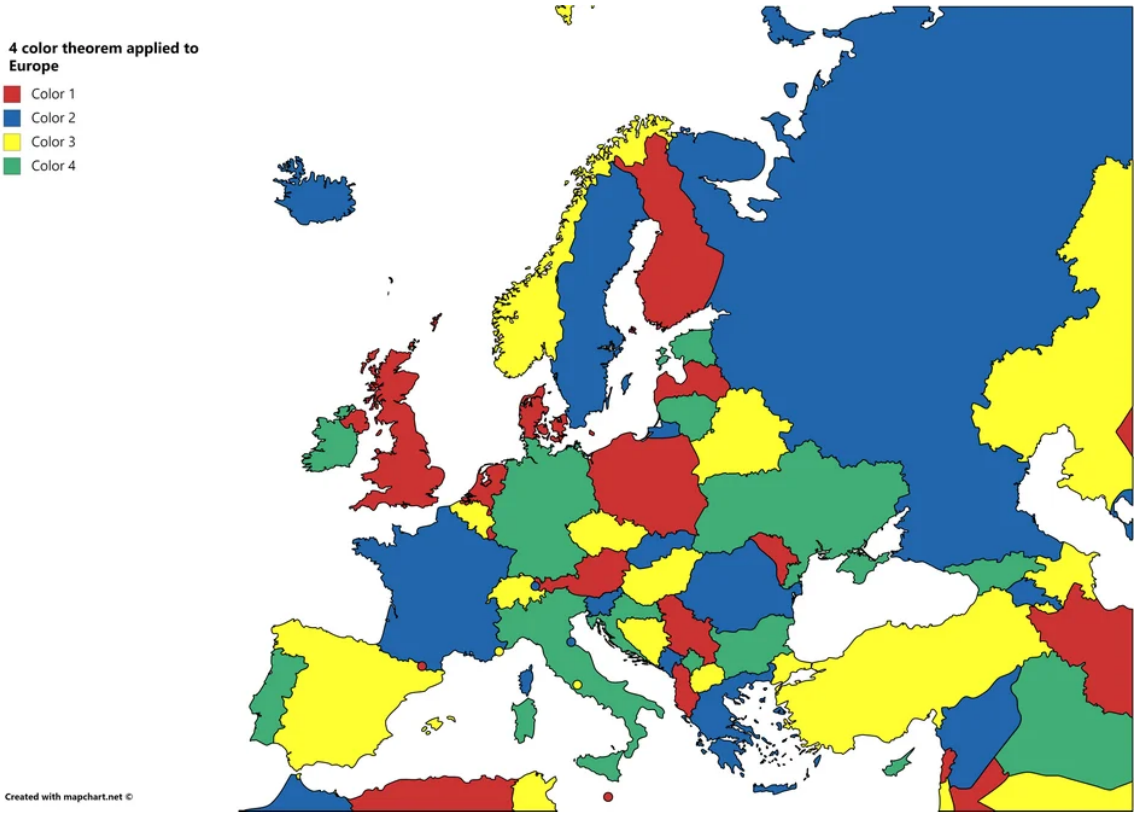
Grafy samozřejmě mohou být výrazně složitější, třeba s cykly. Občas je potřeba vyjádřit závislosti mezi buňkami, které spolu v síti neousedí. GBWFC lze použít na řešení sudoku, kde jsou sice pravidla symetrická, ale ne všechny na sobě závislé buňky spolu sousedí.

Políčko vlevo nahoře má jenom tři sousedy v síti, ovšem podle pravidel sudoku je závislé na levém sloupci, vrchím řádku a 3x3 čtverečku vlevo nahoře. To bychom jen sotva obsáhli ve čtvercové síti.





Dalším příkladem může být obarvování mapy, super, mapa světa je přece rovinný graf, tak to je jednoduché a dokonce to zvládneme jen čtyřmi barvami! Opak je pravdou, co takové exklávy a enklávy? Třeba Kaliningrad a pevninské Rusko by měly mít na mapě stejnou barvu, přestože to jsou dvě buňky oddělené jinými buňkami. Pokud se takových problémů sejde víc, nemusíme být schopni je ohlídat.



> podle komentáře na Redditu tato mapa není úplně správně, pro ilustraci postačí

Z těchto úvah jasně vidíme, že nejlepší by bylo mít algoritmus, který pracuje na libovolném orientovaném grafu - jakýkoliv jiný problém na něj dokážeme převést.

Proč nepoužít klasický algoritmus na barvení grafu?

Jednou z výhod WFC je, že si předem můžeme říct, jak často se mají jednotlivé typy buněk (barvy) vyskytovat. Například místnosti s bossy by neměly být časté, to jenom z pravidel vykoukat nejde, proto WCF na začátku předáme pravděpodobnosti/četnosti (například `normalni : 80, boss : 15, poklad : 5`). To standardní implementace obarvování neumí.

Zdroje

- Automatic Generation of Game Content using a Graph-based Wave Function Collapse Algorithm
- The Wavefunction Collapse Algorithm explained very clearly