# 1.cím dia

Témám egy catan térkép generátor amely kiegyensúlyozza a játék térképet.

# 2. dia **Probléma**

A cél tehát hogy a kezdő játékos ne tudjon egyértelműen egy túl jó kezdőhelyet választan

Ehhez a következő szempontokat kell figyelembe venni:  
Azonos erőforrások ne érintkezzenek egymással

Azonos számok ne érintkezzenek kifejezetten figyelve a két dobokócka összegeként kapott számok nem egyeneltes eloszlásának.

# 3.dia **Erőforrások**

A kezdő játékos választ egy olyan helyet a településnek ahol több azonos erőforrás van ahogyan ez a képen piros pöttyel van jelölve.

Ezzel a többi játékosnál hiány fog jelentkezni az adott erőforrásból

Egyértelműen látszik, hogyha ilyen erőforrás tömbök alakulnak ki, akkor azon játékosoknak akik ilyenre tesznek szert a teljes játék során előnyük lesz.

# 4.dia **számok**

Minden körben a játékosok dobnak a 2 kockával és ezek összege alapján termelnek az erőforrásmezők, ha a dobott szám megegyezik a mezőn található számmal.

A 2 kockával való dobás összegének eredménye nem egyenletes eloszlású, ezt szemlélteti a diagramm. Emiatt a vannak gyakrabban előforduló jó számok, (6,8) és nagyon ritkán előforduló rossz számok (2,12).

Nyilvánvaló, hogy olyan helyre akarunk települést rakni ahol jobb számok érintkeznek, de emiatt a kezdő játékos szintén előnyt tud szerezni ha a jó számok egymás szomszédai.

Az alsó ábrán látható, egy jó és egy rossz kezdő hely, a bal oldali statisztikailag jóval többet fog termelni.

# 5.dia **Megoldás**

A térképet értelmezzük gráfként, amelyben a csúcsok a mezők és akkor megy él két csúcs között, ha érintkeznek.

Ezután a megoldáshoz 2 külön gráf színezési problémát kell megoldani, ezeket 1-1 a különböző beállításoknak megfelelően paraméterezhető rekurzív backtracking algoritmussal oldottam meg.

# 6.dia Tervezés, UML

Ahogyan látható A logikai és a Nézet teljesen elkülönül, és a App komponens csak a lényeges információt látja a logikából.

A térkép elemeit a MapTile osztály reprezentálja, tárolja a típusát és a számot, valamint mivel gráfként kezelem a térképet, tárolja a szomszédok indexeit (MapViewStateHandler mapTiles attribútum). A gráf egy tömbben tárolódik az egyszerű megjelenítéskori feldolgozás miatt)

Az erőforrásokat a generateMapWithBacktracking függvény helyezi el a számokat pedig a PlaceNumbers.

# 7.dia Logika elkészítése és tesztelés

A következő lépésben elkészítettem a logikát majd teszteltem, hogy megfelelően működik e.

Minden függvény különböző bementtel tesztelve van, többnyire a 2 fő backtrackingen keresztül.

A térkép generálást 1000szer futtatja és ellenőrzi minden tesztelésnél, csak akkor sikeres ha minden eset az elvártnak megfelel.

A képen látszik, hogy a placeNumbers 47.sora nem fut le, ennek az oka, hogy egy hiba dobás van ebben a sorban ha elfogynának a számok, de ez mióta helyen az algoritmus nem fordul elő.

# 8.dia UI

Ezután következett a UI tervezés és megvalósítás.

# 9.dia Wireframe

A wireframe figmában készült, később a clear funkció kikerült mert feleslegesnek bizonyult-

# 10.dia Nappal

Látható a reactben készített térkép generátor és a végleges opciók:

* Random térkép
* „Okos térkép”
  + - 6 és a 8 érintkezhe
    - 2 és 12 érintkezhet
    - Ugyan azok a számok érintkezhetnek
    - Ugyan azok az erőforrások érintkezhetnek
* Amit más generátor nem csinál: Csak a számok vagy csak az erőforrások újra generálása

+ konzulensem kérésére más erőforrás képekhez készül night map kapcsoló amit ha megnyomunk (dia váltás) az esti térképet láthatjuk.

# 11.dia Este

Az esti térképhez saját gyártású többnyire kézzel rajzolt erőforrás képek készültek.

# 12.dia Vége