Symulator rozwoju cywilizacji

Projekt na zaliczenie przedmiotu
Symulacje i sterowanie procesów dyskretnych
pod kierunkiem
mgra inż. Jakuba Porzyckiego

Autorzy:
Jan Bieroń
Robert Przystasz
Przemysław Szczepaniak

Spis treści

1 Abstrakt

Celem projektu była próba graficznego przedstawienia procesu rozwoju i upadku cywilizacji przy uwzględnieniu wszystkich istotnych czynników mogących mieć wpływ na ten proces, a więc ukształtowania terenu, interakcji z sąsiednimi cywilizacjami, łatwości zdobywania i obrony terenów jak również ich strategiczna przydatność. Model symulacji oparty został o dyskretne automaty komórkowe.

Cywilizacja, w kontekście tego projektu, powinna być rozumiana jako państwo lub twór państwowy, posiadające władcę i jasno określone strefy wpływów (nazywane tak wymiennie z granicami). Nie chodzi nam o modelowanie wpływów cywilizacji w ich pierwotnym, kulturowym sensie, są to bowiem wpływy bardzo złożone i równocześnie subtelne. Zasadniczo terminem, jakim powinniśmy się posługiwać jest „państwo", niestety zdaliśmy sobie z tego sprawę w bardzo późnym etapie tworzenia naszego projektu, kiedy stosowna zmiana byłaby bardzo kosztowna. Dlatego też, w celu zachowania jednolitości kodu z opisującą go dokumentacją, zostaniemy przy terminie „cywilizacja".

Należy także zauważyć, że model budowaliśmy z myślą o cywilizacji starożytnej lub ewentualnie średniowiecznej. Późniejsze okresy, ze względu na skomplikowane relacje polityczno-gospodarcze, i rozwój technologiczny nie stanowią przedmiotu tej symulacji. Nie jest wykluczone, że dla epoki kolonialnej osiągnęlibyśmy wyniki nieodbiegające od tych, które otrzymaliśmy dla przyjętego przez nas okresu historycznego, ale z całą pewnością próba modelowania rozwoju państw począwszy w okresie rewolucji industrialnej lub dowolnej późniejszej po prostu mija się z celem.

2 Przyjęte założenia

Przyjęty przez nas model automatu komórkowych oparty jest o dwuwymiarową siatkę kwadratowych komórek o sąsiedztwie w sensie Moore'a. Ze względu na bardzo dużą rozdzielczość stosowanych plansz zrezygnowaliśmy z geometrycznego oddalenia sąsiadów leżących po skosie (koszt $\sqrt{2}$ zamiast 1), co nie poskutkowało pojawieniem się żadnych artefaktów. Automat w przypadku naszego projektu możemy określić jako homogeniczny. Aktualizacja czasu odbywa się synchronicznie. Funkcja przejścia do kolejnego stanu jest niedeterministyczna, oparta o czynnik losowy, a więc automat nie jest odwracalny, nie było to jednak przydatne dla naszego problemu. Ze względu na to, że naraz badamy tylko pewien wycinek kuli ziemskiej, nasze warunki brzegowe są zamknięte oraz pochłaniające. Dodatkowo, ze względu na sposób działania algorytmu parsującego mapę (patrz!) brzegi mapy posiadają zniekształcone dane (dla przyjętej w dokumentacji rozdzielczości jest to skrajny pas o szerokości 6px), co generuje nietypowe zachowanie przy rozwoju cywilizacji, więc należy mieć to na uwadze przy analizie symulacji.

Każda komórka w siatce scharakteryzowana jest, oprócz swoich współrzędnych, przez dwa parametry: desirability (popyt) i defensibility (obronność). Te dwie wartości stanowią sedno algorytmu zarządzającego symulacją. Desirability odzwierciedla w jakim stopniu dany region (komórka) jest pożądany przez wszystkie cywilizacje. Wpływ na ten parametr ma bliskość wody oraz obecność flory. Uznaliśmy oba te czynniki za istotne przy podejmowaniu decyzji, jaki obszar dana cywilizacja powinna w następnej kolejności dołączyć do strefy swoich wpływów. Im wyższe desirability, tym większe prawdopodobieństwo, że w następnym kroku symulacji cywilizacja sąsiadująca z tą komórką będzie starała się o nią walczyć. Jeżeli komórka nie posiada jeszcze właściciela (nie jest zajęta przez jakąś cywilizację), aneksja odbywa się automatycznie, w przeciwnym wypadku brany jest pod uwagę drugi parametr.

Defensibility oznacza jak łatwo jest dany region obronić przed próbą przejęcia ją przez inną

cywilizację. Można to rozumieć jako wartość strategiczną. Defensibility zależy od ilości górzystych regionów w pewnym otoczeniu komórki (od 0 do 25) oraz graniczenie z wodą. W tej definicji najbezpieczniejsze i najtrudniejsze do zdobycia będą górzyste wyspy, zaś najłatwiejsze – puste równiny. Wydaje się nam to podejściem zdroworozsądkowym.

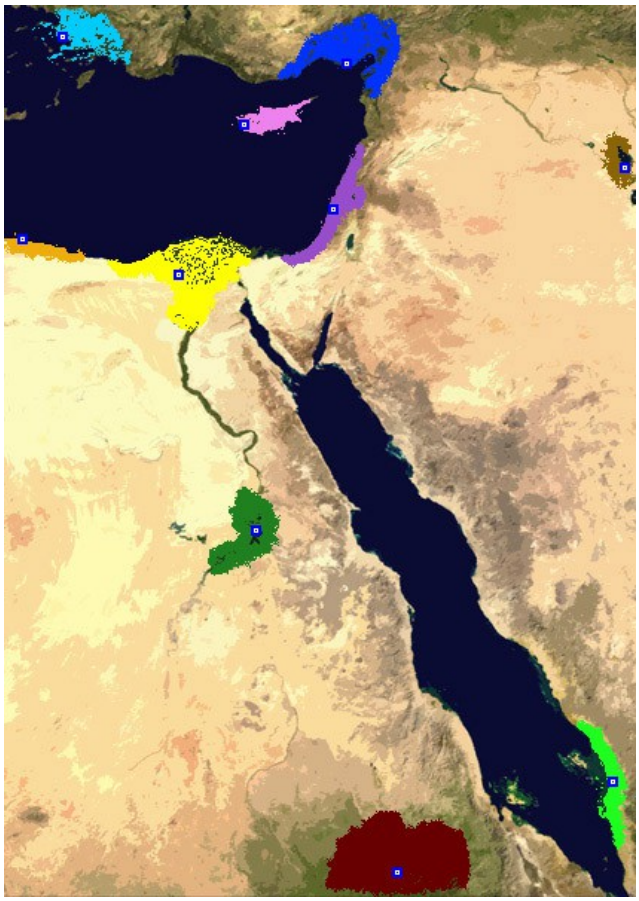
Każda cywilizacja posiada stolicę i jest to również odzwierciedlone w naszym modelu. Stolica jest pierwszą komórką, z jakiej składa się cywilizacja kiedy powstaje. Kiedy region w którym znajdowała się stolica zostanie utracony, inny zostanie wybrany na podstawie desirability wszystkich dostępnych komórek, ale z dużą losowością. Położenie stolicy jest istotne z punktu widzenia rozwoju cywilizacji, gdyż możliwość ofensywna dla danej komórki (wykorzystywana między innymi jako wartość przeciwstawiana defensibility wrogiej komórki) jest funkcją odległości stolicy i jest to naturalnie funkcja malejąca wraz ze wzrostem odległości.

Oprócz parametrów komórek, zdefiniowaliśmy także dwa parametry (w zasadzie trzy) dla cywilizacji. Pierwszą parę już omówiliśmy, gdyż są to współrzędne stolicy. Drugim parametrem jest siła cywilizacji. Określa ona zdolność do przejmowania nowych terenów. W każdej iteracji symulacji każda aktywna cywilizacja może przejąć ograniczoną ilość z puli wszystkich sąsiadujących z nią komórek. To ograniczenie jest wprost proporcjonalne właśnie do siły cywilizacji, zaś ta wynika z ilości zajętych terenów – dokładnie jest sumą po czasie z pewną wagą zmniejszającą rozwój tej cechy. W przypadku próby przejęcia komórki posiadającej już właściciela,

Wraz z rozwojem cywilizacji coraz bardziej powszechne będzie zjawisko rozpadu cywilizacji – pewna komórka, zwykle znajdująca się na peryferiach, postanowi wyłamać się spod rządów stolicy i sama stanie się stolicą nowej cywilizacji. Taki bunt jest bardzo kosztowny dla macierzystej cywilizacji i czyni ją podatną na agresje sąsiadów; jeśli jest ona jednak dostatecznie silna, co jest standardem przy buntach, szybko jest w stanie odrobić stratę siły. Natomiast buntownicza komórka dostaje „kredyt rozruchowy” na rozwój, co ma symulować przyłączanie się (dobrowolne lub nie) okolicznych komórek do buntu. Więcej na ten temat (patrz!).

3 Przyjęte dane do symulacji

Chociaż początkowo planowaliśmy modelować rozwój Imperium Rzymskiego od jego powstania aż do rozpadu, ostatecznie zdecydowaliśmy się na Egipt ze względu na wyraźny podział między rejonami żyznymi i tymi nieatrakcyjnymi do przejęcia. Obszar symulacji to południowo-wschodni wycinek basenu Morza Śródziemnego, Morze Czerwone i okalające je: Półwysep Arabski oraz północno-wschodni skrawek Afryki. Na potrzeby modelu wybraliśmy 10 par współrzędnych określających położenie pierwszych dziesięciu stolic cywilizacji. W trakcie trwania symulacji miejsca te mogą ulec zmianie, tak jak mogą powstawać nowe cywilizacje z własnymi stolicami.

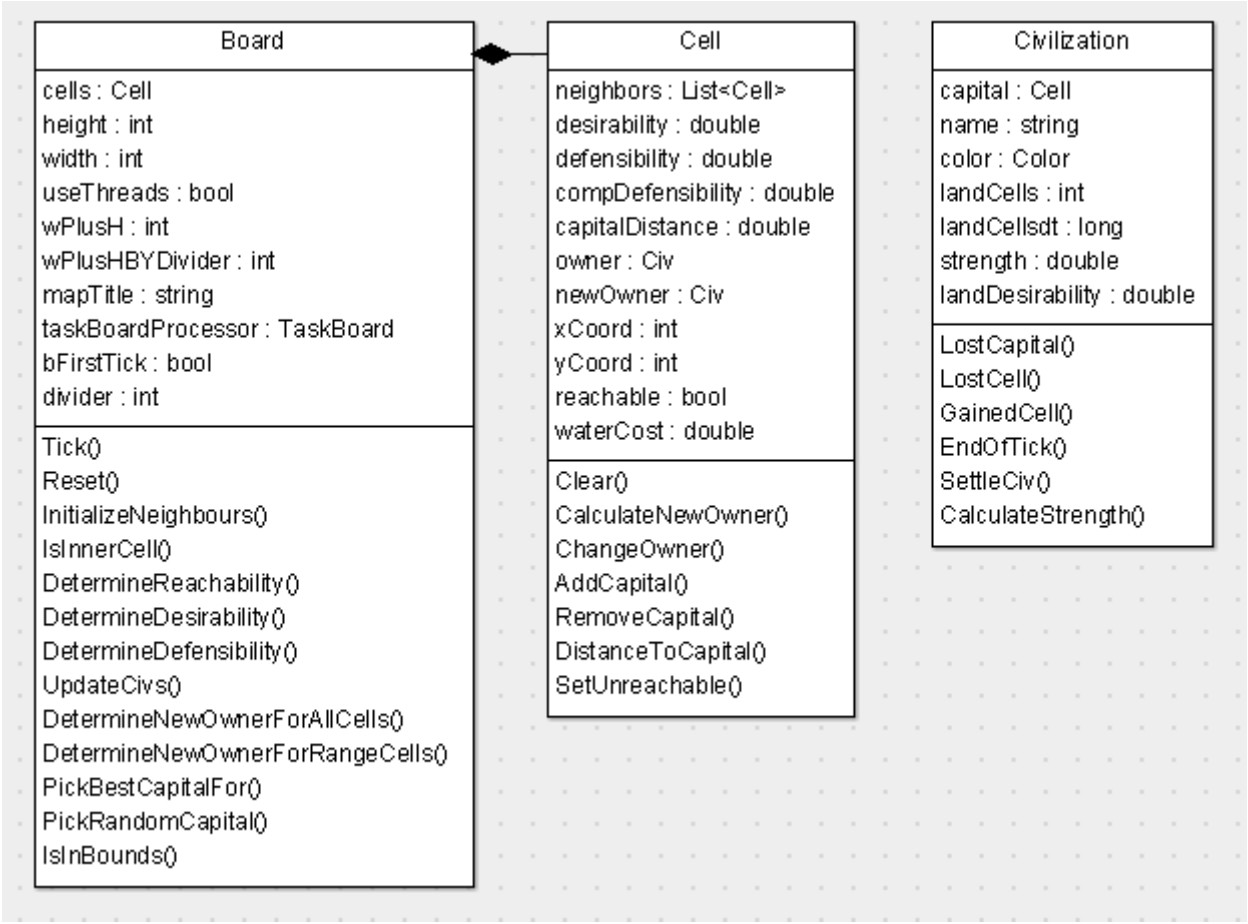


Rys 1.1 i 1.2. Czysta mapa symulacji oraz jej kopia z początkowymi 10 stolicami, obie zatrzymane w 50. iteracji

4 Szczegóły implementacyjne

4.1 Diagram klas modelu

Poniżej przedstawiamy diagram klas tworzących nasz model, aby pokazać, w dużym uogólnieniu, jak został zbudowany. Staraliśmy się zachować przejrzystość w kodzie i stosować intuicyjne nazwy, aby kod sam dla siebie stanowił dokumentację i czytający go wiedzieli, jakie procesy zachodzą w ramach danej funkcji.



4.2 Algorytm parsera map

Na potrzeby projektu stworzyliśmy skrypt w Matlabie, parsujący plik graficzny z mapą i liczący na jego podstawie trzy mapy tej samej rozdzielczości i rozmiaru:

- land.bmp, będący zerojedynkową maską oznaczającą układ woda-ląd
- desirability.bmp, będący zbiorem punktów z przedziału 0-255, przy czym im wyższa wartość (jaśniejszy piksel w sensie graficznym), tym większe desirability dla tej komórki
- defensibility.bmp, analogicznie jak wyżej

Naturalnie, jeden piksel odpowiada jednej komórce automatu. Źródłowa mapa powinna w sposób graficzny, choć niekoniecznie zauważalny dla ludzkiego oka, odróżniać wodę od lądu, rejony bardziej obfite w roślinność od tych ubogich oraz uwzględniać wysokość terenu. Teoretycznie nic

nie stoi na przeszkodzie, by wspomnianego kodu używać do wczytania dowolnej mapy spełniającej te wymagania, praktyka stawia jednak dodatkowe warunki. Ponieważ każda mapa w inny sposób przedstawia powyższe cechy jakościowe, dla każdej mapy należy ręcznie skalibrować parametry działania skryptu, lecz ze względu na to, że jest on zasadniczo prosty, nie powinno stanowić to dużego problemu.

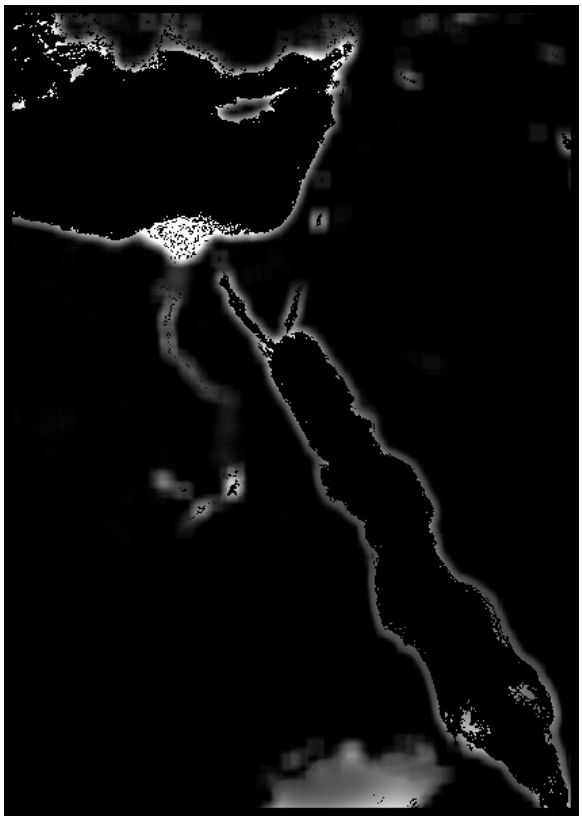
Poniżej przedstawimy i opiszemy wybrane elementy kodu, a następnie pokażemy 3 obliczone mapy uzyskane za jego pomocą.

```
r = img(:,:,1); %podział obrazu źródłowego na kanały
g = img(:,:,2);
b = img(:,:,3);

desert = r>210 & g>210 & b<200; %pustynia bedzie miala zerowe desirability (nie uwzględniając wody)
land = (b<60 & b>35 & r+g<130) | (b>10 & b<31 & g>30 & g<80); %podział na wodę i ląd
land = imerode(land, strel('disk',1)); %zgaszenie samotnych pixeli
highlands = r-g<30 & b+20<r & b<160 & b>60; %wzgórza są istotne przy liczeniu defensibility

for x=1:X
    for y=1:Y
        %sumowanie górzystości otoczenia każdego piksela, za wyjątkiem skrajnych
        c = 0;
        for i=x-1:x+1
            if i<1 || i>X, continue, end
            for j=y-1:y+1
                if j<1 || j>Y, continue, end
                c = c + highlands(i,j);
            end
        end
        high(x,y) = c;

        %poszukiwanie roślinności (odcienie zielone) - lasy, laki, zyzne brzegi
        if g(x,y)-5 > r(x,y) && g(x,y) > b(x,y)
            foliage(x,y) = g(x,y)-b(x,y);
        end
    end
end

def = (high - desert - land + 1)*10; %defensibility = obronność

d = 10; %iterowanie po otoczeniu o promieniu 10 każdego piksela (za wyjątkiem skrajnych 10)
for x=1+d:X-d

    for y=1+d:Y-d
        if(land(x,y) == 1)
            continue;
        end
        wsum = 0;
        gsum = 0;
        for i=x-d:x+d
            for j=y-d:y+d
                wsum = wsum + land(i,j);% 0 lub 1
                gsum = gsum + (foliage(i,j)>0);
            end
        end
        waterProx(x,y) = wsum; %bliskość do wody - wpływa na desirability
        foliageProx(x,y) = gsum * 2; %bliskość do flory - wpływa na desirability

        if waterProx(x,y) > 180 %rejony nadbrzeżne łatwiej się bronią
            def(x,y) = def(x,y) + waterProx(x,y)/4;
        end
   end
end

des = uint8(~desert-land)*25;
des = des + waterProx + max(foliageProx, uint8(foliage)); % desirability = popyt
```

Otrzymane za wyżej przytoczonego kodu mapy wartości:

Rys 2. Macierz woda/ląd (woda na czarno :)

Do pierwszej i trzeciej mapy dołączyliśmy ramki wyłącznie dla zwiększenia czytelności. Nie stanowi ona części mapy. Należy zwrócić uwagę, że mapa defensibility w rzeczywistości nie jest taka jednolita, jej wygląd w tym dokumencie jest zniekształcony.



Rys 3. Od lewej: desirability (popyt) i defensibility (obronność).
Wyższa wartość odzwierciedlona jest przez jaśniejszy odcień.

Na mapie desirability daje się zobaczyć zniekształcenie brzegowe będące skutkiem działania algorytmu parsującego (przy dolnej krawędzi w okolicy jasnej plamy). Jest to jedna z rzeczy, które mają potencjał do poprawy.

4.3 Algorytmy wyliczania wartości modelu

4.3.1 Odległość

Zaimplementowaliśmy trzy metody obliczania odległości pomiędzy komórkami. Różnią się między sobą prostotą obliczeń i „inteligencją", przydatnością wyników. Dany algorytm należy wybrać w konfiguracji przed uruchomieniem programu.

CT_PYTHAGORIAN to prymitywny, ale bardzo szybki algorytm. Liczy on po prostu odległość w linii prostej, przy czym odległość wyrażona jest w komórkach, i nie bierze on pod uwagę, że przechodząc komórki diagonalnie geometrycznie pokonuje odległość 1.41 raza. Ze względu na swoją prostotę i prędkość jest on domyślnym sposobem obliczania odległości, pomimo niezbyt wartościowych wyników.

CT_PYTHAGORIAN2 to usprawnienie powyższego, uwzględniający wspomnianą zależność, ale wciąż nie biorący pod uwagę przeszkód terenowych, które byłyby istotne dla ludzi.

CT_ASTAR ma największą złożoność obliczeniową; korzystanie z niego zauważalnie zmniejsza prędkość symulacji, równocześnie znacząco poprawiając jej jakość. Bierze on pod uwagę przeszkody leżące na drodze pomiędzy dwiema komórkami.

4.3.2 Desirability i Defensibility

Obie te wartości zostały już omówione przy okazji opisywania działania parsera map (patrz 4.1)

4.3.3 Siła cywilizacji

Jest obliczana za pomocą wzoru:

```
strength = landDesirability + 0.1 * landCellsdt;
```

landDesirability to suma popytu po wszystkich komórkach należących do cywilizacji (czyli po prostu ilość * jakość).
landCellsdt to suma powierzchni cywilizacji po czasie (który jest dyskretny, bo iterowany).

Tłumaczy to omawiane już przez nas zjawisko mające miejsce podczas wybuchu buntu, kiedy macierzysta cywilizacja zostaje chwilowo osłabiona i staje się bardziej podatna na ataki, ale jeżeli przetrwa to dzięki „elementowi całkującemu" odzyska siłę.

4.3.4 Inne

Komórki wodne również są zdobywane przez cywilizację, nie podnoszą jednak jej landDesirability (oczywiście) ani nie wpływają na siłę. Służy to symulowaniu zdobywania nowych terenów dzięki ekspedycjom morskim.

Przyjęliśmy, że gdy na skutek zmagań z sąsiadem fragment cywilizacji zostanie odcięty od swojej stolicy (nie istnieje już ścieżka zawierająca się w terytorium cywilizacji, po której da się dotrzeć do niej), zostaje on utracony – usuwana jest informacja o jego przynależności, a jego dotychczasowy właściciel obserwuje utratę powierzchni, landDesirability, a więc również siły. Proszę zwrócić uwagę, że ze względu na rozwiązanie omówione w akapicie wyżej, nie dotyczy to wysp i kolonii zamorskich – chyba że sąsiednia cywilizacja (lub, wspólnymi siłami, ich większa liczba) „zastosuje embargo".

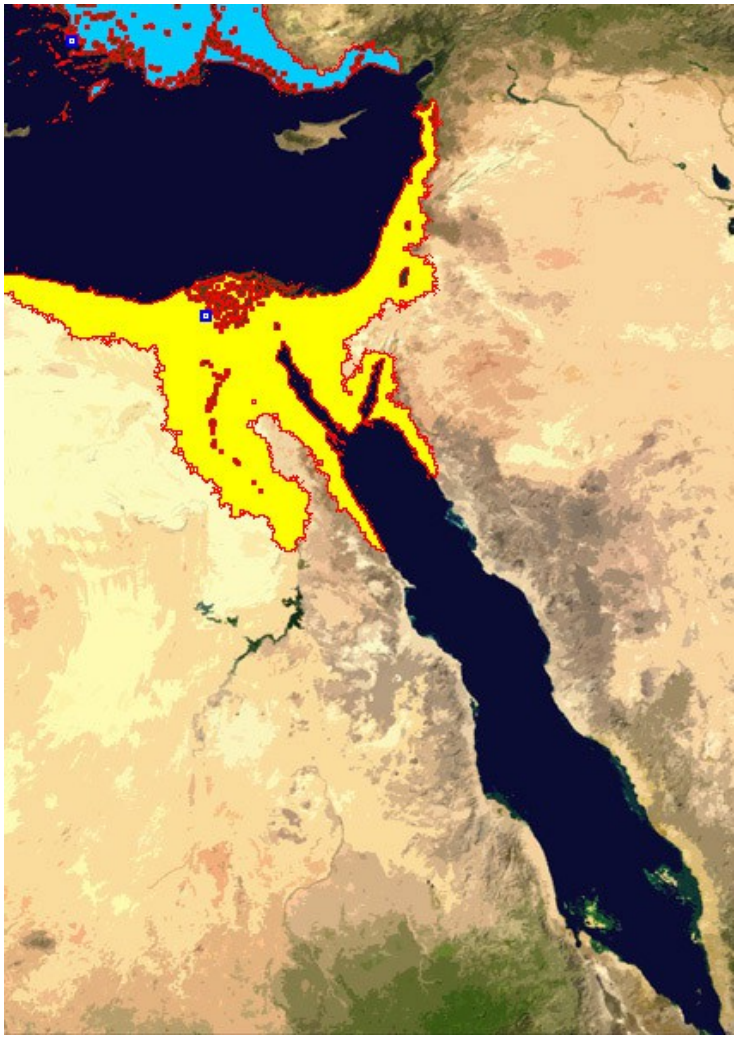
5 Walidacja

Walidację w naszym rozumieniu należało przeprowadzić porównując historyczne dane o granicach wpływów starożytnego Egiptu z wynikami naszej symulacji. Można uważać za błąd nie upewnienie się przed wybraniem Egiptu jako przedmiotu naszego modelu, czy mamy dostęp do informacji na temat jego granic, aby walidacja była możliwa. Niemniej znaleźliśmy granice z kilku okresów, dzięki którym w uogólnieniu wiemy, jak przebiegał rozwój granic tego państwa.

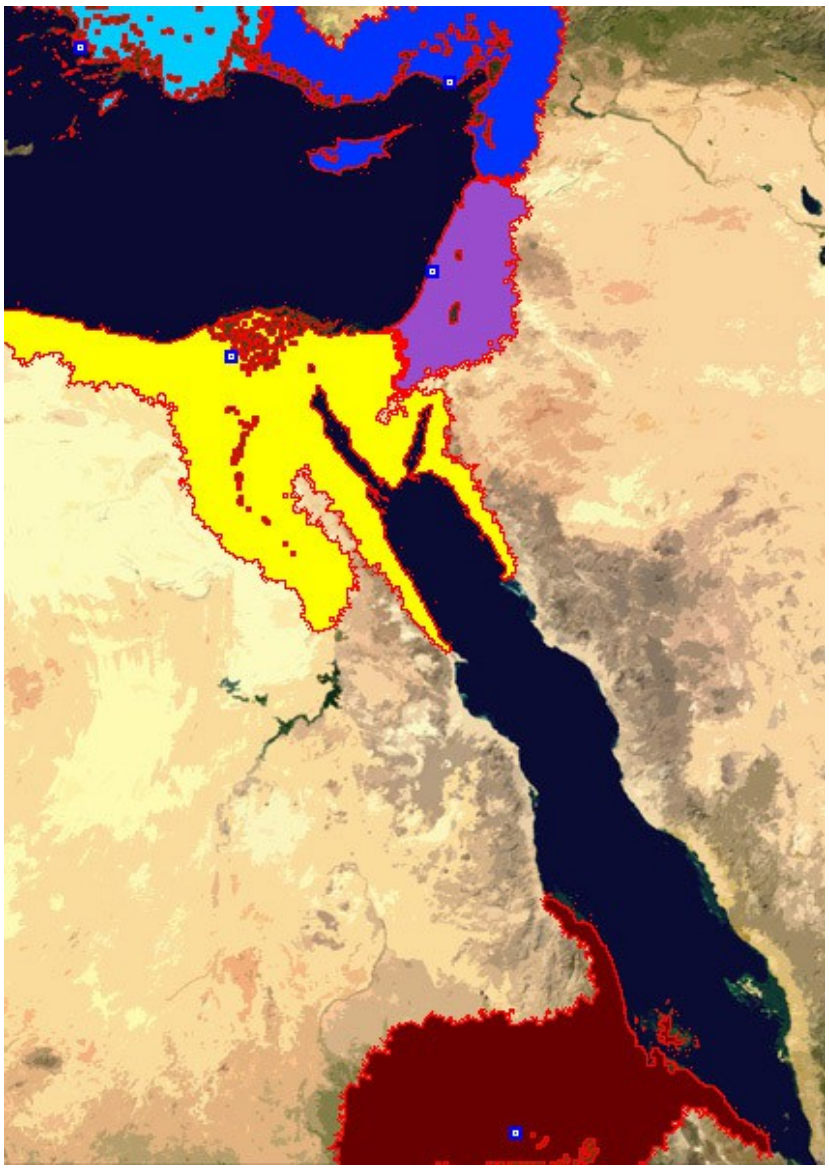
Rys 4. Szczytowy zasięg królestwa Egiptu

Z map jakie znaleźliśmy wynika, że zanim Egipcjanie dokonali ekspansji wzdłuż brzegów Morza Śródziemnego i Czerwonego, zajęli dolinę Nilu sięgając za granicę dzisiejszego Sudanu. Nasza symulacja nie naśladuje tego zachowania należycie, gdyż „nasi Egipcjanie" z równą chęcią rozwijają się na północny wschód jak na południe. W tym konkretnym przypadku może się okazać, że model jest poprawny, ale nasze dane są niedokładne – wystarczy zaś tylko że zwiększymy atrakcyjność brzegów Nilu w stosunku do terenów dzisiejszej Syrii i być może otrzymamy lepsze wyniki. Drugą rozbieżnością jest brak Egipcjan na wschodnim brzegu Morza Czerwonego, zasiedlanego w naszej cywilizacji prawie tak samo szybko jak zachodni. Tu problem wydaje się bardziej złożony, gdyż jest więcej sposobów podejścia do rozwiązania go. Można sprawdzić, czy postęp Egipcjan w tamtym rejonie nie był hamowany cywilizację na wschodzie, którą moglibyśmy tam wtedy umieścić. Należy też sprawdzić, czy Egipcjanie dysponowali technologią pozwalającą zakładać kolonie na drugim brzegu morza i czy takie przedsięwzięcie byłoby dla nich opłacalne. Niestety żaden z nas nie ma wykształcenia historycznego pozwalającego udzielić na te pytania odpowiedzi. Będziemy musieli poszukać ich, jeżeli chcemy dokonać kalibracji i zwiększyć skuteczność walidacji.

Rys 5. Rozwój Egiptu. Widać, że ekspansja wzdłuż Nilu miała większy priorytet od rozrastania się wzdłuż brzegów morza śródziemnego



Rys 6. 211. krok symulacji. Widzimy, że „automatyczni" Egipcjanie nie dotarli do źródeł Nilu, a dosięgnęli już współczesnej Syrii.



Rys 5. Również krok 211. dla symulacji z większą ilością cywilizacji. Cywilizację fioletową można interpretować nie tylko jako wrogiego sąsiada, ale również jako każdy czynnik niesprzyjający rozwojowi Egipcjanów na tym terenie (na przykład rozwój jakiejś choroby – również silniejszy w rejonach bardziej uroślinnionych, bo bardziej wilgotnych).

Zastosowanie większej ilości cywilizacji poprawiło wyniki, ale jest to swego rodzaju sztuczka i działa na krótką metę – wprowadza tylko inne zniekształcenia do modelu. W kalibracji należałoby zwiększyć stosunek desirability wody słodkiej do słonej, gdyż obecne proporcje są niewystarczające.

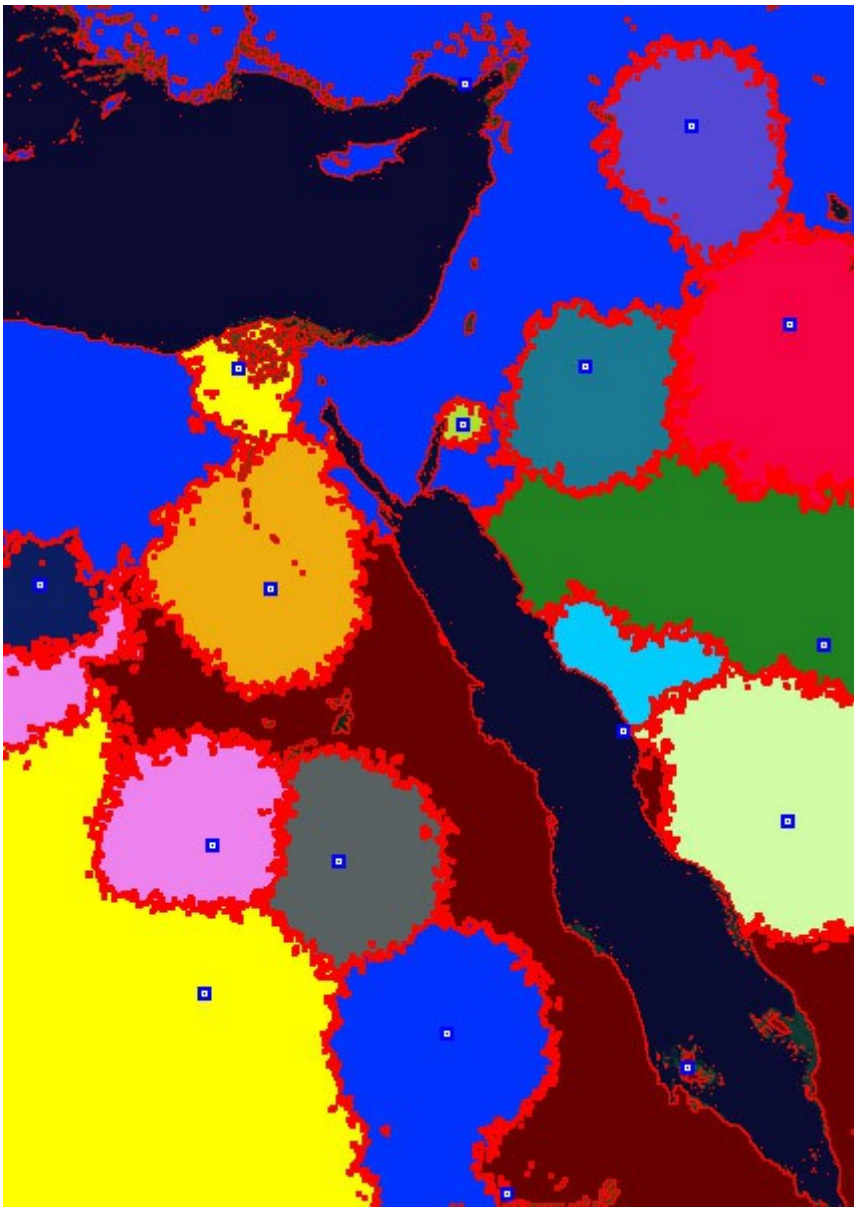
6 Potencjał rozwoju

W obecnej sytuacji nie możemy uznać, że nasza symulacja poprawnie przechodzi walidację. Musimy się zastanowić, czy będziemy kontynuować kalibrację Egiptu, czy przeniesiemy się w lepiej udokumentowane rejony.

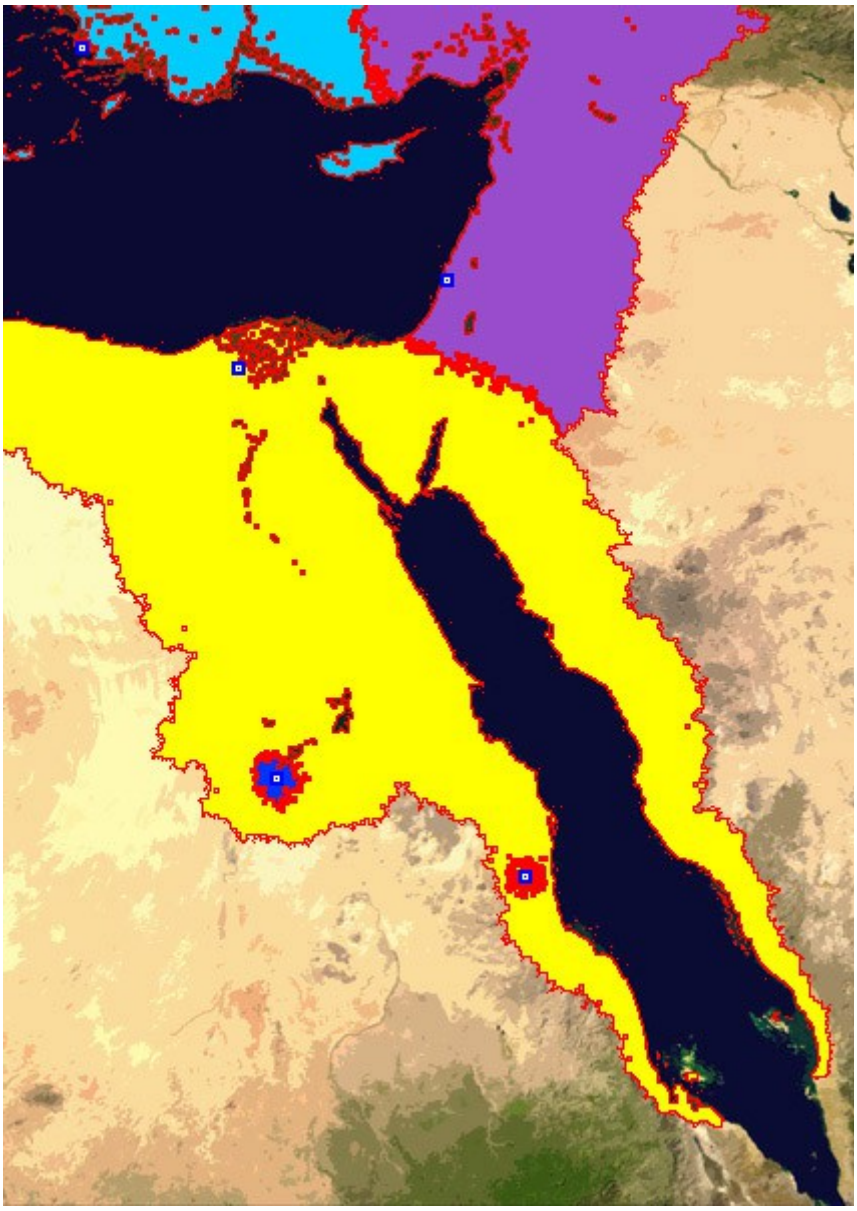
Oprócz kalibracji powinniśmy też wziąc pod uwagę inne możliwości poprawy naszej symulacji. W obecnej wersji programu cywilizacje nie słabną same z siebie. Nie dochodzi nigdy do utraty własności komórki bez przejęcia jej przez inną cywilizację. Jeżeli pominiemy powstawanie buntów, to w symulacji dla jednej cywilizacji będziemy obserwować nieustający rozwój, który chociaż będzie stale słabł, to cywilizacja nigdy nie osiągnie stagnacji, nie mówiąc już o regresji. A przecież wiele zjawisk historycznych mogłoby powodować takie osłabienia – epidemie, kataklizmy, rozdrobnienie władzy, słaba monarchia, i tym podobne.

Przydatne może także okazać się dodawanie początkowych cywilizacji z poziomu GUI. Uprości to i przyspieszy testowanie modelu i pozwoli na oddzielenie funkcji testera/analityka od programisty.

Należy sparsować więcej map, a także usprawnić samego parsera, aby rozróżniał wodę słodką od słonej (pomijając takie ewenementy jak słone jeziora, jest to najzupełniej wykonalne).



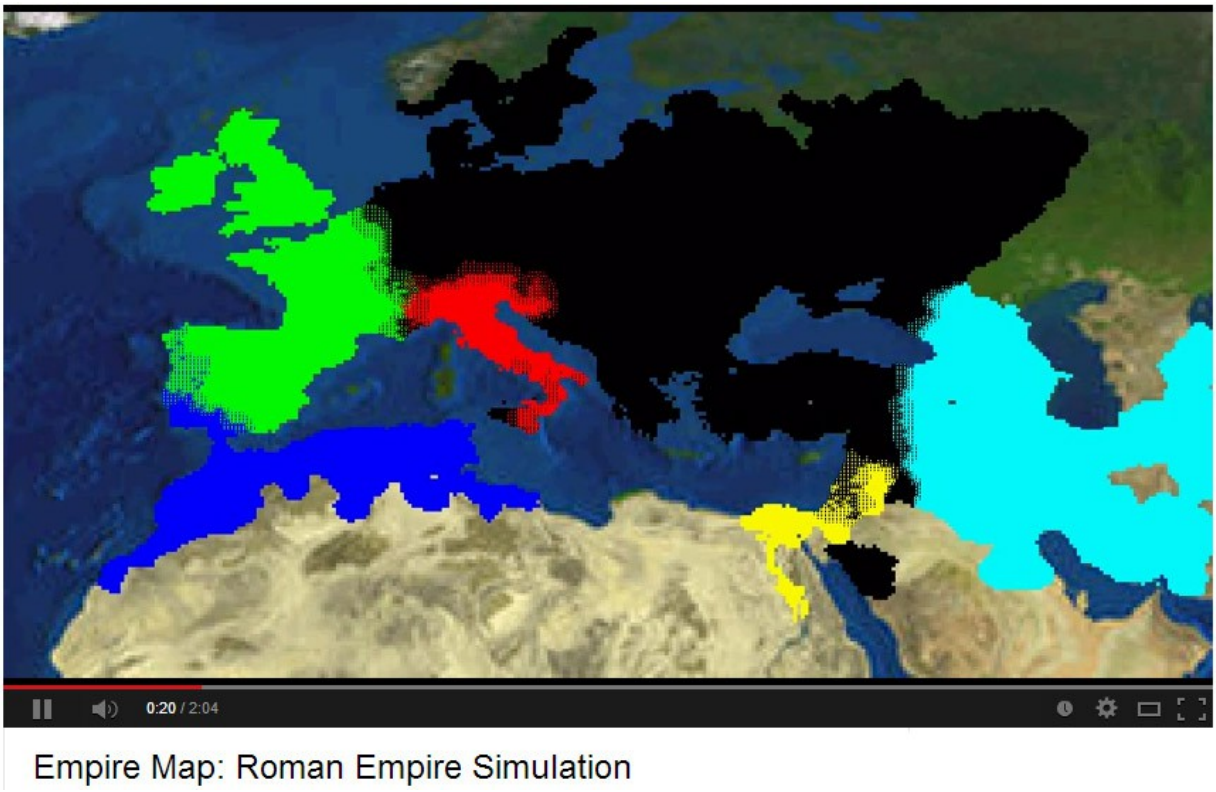
Rys 6. W kroku 9000. widoczne jest zjawisko swoistej bałkanizacji. Niemniej należy traktować to jako ciekawostkę, gdyż przy tak długim czasie istnienia symulacja wymknęła się spod kontroli.



Rys 6. W tej symulacji, w kroku 501 Egipt jest już na tyle rozległym państwem, że pojawiają się w nim tendencje separatystyczne. O ile większość takich buntów nie trwa długo (kilkadziesiąt – kilkaset iteracji), o tyle ognisko znajdujące się w górnym biegu Nilu ma spore szanse na przetrwanie, ze względu na duże desirability swojego terytorium.

7 Istniejące rozwiązania, podobne projekty

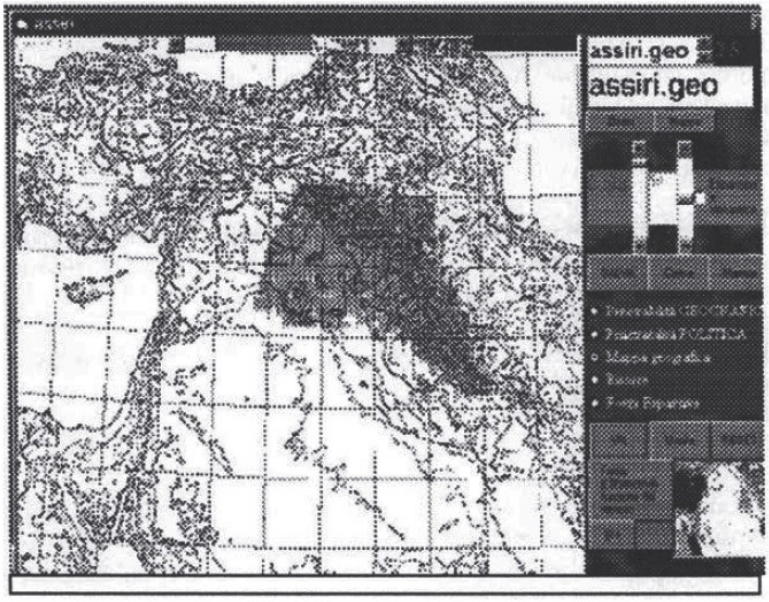
Inspiracją dla naszego projektu była symulacja rozwoju starożytnego Rzymu, aż do jego schyłku i rozpadu. Znaleźliśmy wideo w serwisie youtube.com przedstawiające działanie tego programu. Byliśmy pod wrażeniem wierności jego modelu względem faktycznych historycznych granic Imperium Rzymskiego, jednak po rozmowie z autorem okazało się, że zaprogramował on na sztywno pewne punkty zwrotne, wyzwalacze historyczne, które bardziej niż symulować rozwój, po prostu nim sterowały. Nie mieliśmy wglądu do kodu, toteż wiemy do jakiego stopnia jaka część symulacji wynikała z przypadku, a jaka była zaprogramowana, ale mimo wszystko efekt jest ciekawy.



Rys 7. Symulacja rozwoju Imperium Rzymskiego. Ciekawym rozwiązaniem są zachodzące na siebie strefy wpływów – być może poszczególne komórki przechowują procentową wartość wpływów wielu sąsiednich cywilizacji.

Należy zwrócić uwagę, że Egipt z omawianej właśnie symulacji rozwija się z błędem analogicznym jak w naszym projekcie.

Podczas szukania literatury przedmiotu natrafiłśmy na artykuł naukowy „A Cellular Automata Model of the Expansion of the Assyrian Empire”, opisujący zagadnienie symulacji rozwoju Imperium Assyryjskiego za pomocą automatów komórkowych w sposób analogiczny do tego, jaki ostatecznie obraliśmy. Okazało się, że wiele z proponowanych przez autora publikacji rozwiązań i tak planowaliśmy wykorzystać (na przykład desirability i defensibility), aczkolwiek w kilku miejscach zastosowaliśmy uproszczone podejście (autor dzielił defensibility na militarne i polityczne; siatka była zbudowana z oktagonalnych komórek).



Rys 8. Użyty w publikacji zrzut ekranu z działającej symulacji. Zaciemniony obszar to powierzchnia Imperium w 25. kroku symulacji