How to make your AI win at FreeCiv

CS395 GAI

Spring, 2005

A Motto

- Know the dynamics of the game; in a hundred battles, you will never be defeated
  - With apologies to Sun Tzu

Overview

- Dynamics of FreeCiv
- Winning via military conquest
- Winning via the space race
- Implications for AI design

FreeCiv is about exponential growth

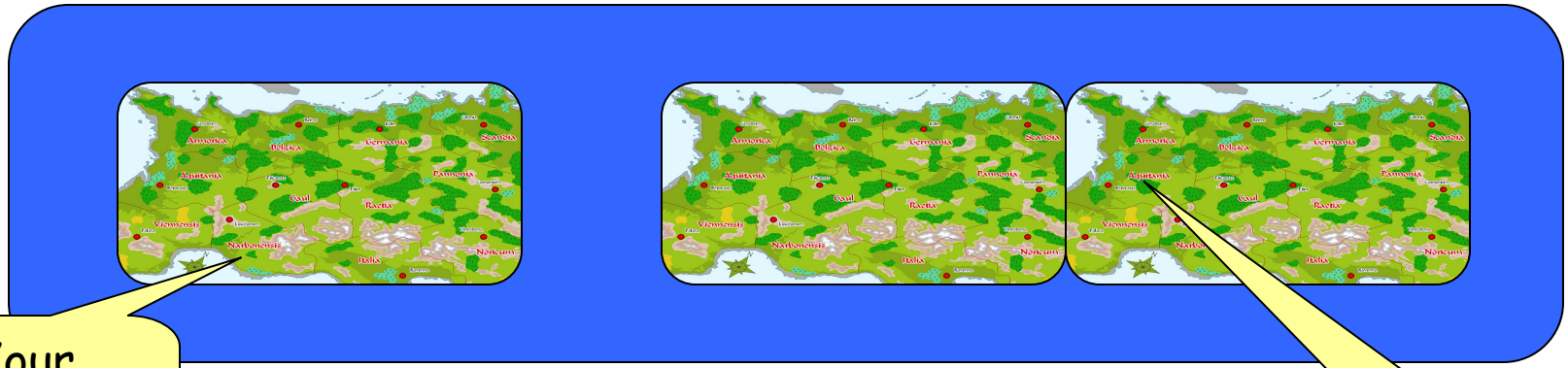
- The more cities you have, the more you can build
- This includes cities
- Nature abhors an unconstrained exponential
  - Terrain, opponents are the limits to growth in this world

Implications for game phases

- Beginning = Slow start
  - Head-start here can make you unstoppable
  - Handicaps here can leave you crippled
- Territorial expansion
  - Exploration and growth
  - War
  - Your resources here determine your options for the endgame
- Struggle phase
  - Spoilers keep the game interesting even if behind

Timing

- Let Tc = # of turns to build a new city
  - Tc drops as a civ progresses



Your starting continent

You just finished settling your continent

Q: How much longer before your opponent settles theirs?

Your opponent's starting continent

Space = Options

- Large empire → more expansion room
- Compact empire → Harder to attack
- Control the avenues of approach
  - Sea power, coastal defenses often important
  - Air power even more so
    - But severe deployment limits

Sowing discord

- You want to act inside your opponent's decision cycle
  - Less relevant for bots, since less global picture built up

Issues for AI design

- What to represent?
- What to pay attention to?
- How to decide what to do next?

Winning by Military Conquest

- What to represent
  - Who are they?
  - Where are they?
  - How strong are they?
  - What are their *centers of gravity*?

Representing the enemies

- List of each civ
- For each civ,
  - Where are its cities, how good are their defenses, what technologies do they have,…
  - Reputation, typical behaviors
- Possible implementations
  - List structures
  - Specialized objects
  - Assertions in a FIRE WM

Why assertions?

- Can use multiple ways to encode knowledge about the game and strategies
  - LTRE rules
  - FIRE-level axioms for queries
  - Suggestions for SOLVE in FIRE
  - Much easier if you've had CS 344

Example: LTRE rules

- Automatically triggered when matching facts arrive in working memory

```
(rule ((:true (isa ?red EnemyCiv) :var ?f1)
       (:true (hasTechnology ?red NuclearPower)
              :var ?f2)
       (:true (hasTechnology ?red Rocketry)
              :var ?f3))
  (rassert! (:implies (:and ?f1 ?f2 ?f3)
                      (suggestedInvestment ABM-System
                                           (HighFn Priority)))
            :avoid-doom))
```

Tradeoffs with LTRE rules

- Useful for automatically noticing interesting conditions
- Requires that information from the FAP's perceptual system be automatically dumped into the WM of a reasoner
  - Hint: You don't want to do this with everything.
  - Best for information that changes slowly, making grand strategy decisions

Example: Chainer axioms

- Requires making explicit query to find currently believed instances

```
(<== (enemyUnitThreatens ?u ?c)
     (enemyUnit ?u)
     (ourCity ?c)
     (withinNervousDistance ?u ?c))

(<== (withinNervousDistance ?u ?c)
     (evaluate ?d (FreeCivDistanceFn ?u ?c))
     (evaluate ?n (FreeCivNervousDistanceFn ?u))
     (LessThan ?d ?n))
```

Tradeoffs with chainer axioms

- Useful for representing and testing complex combinations of conditions
- Once the representations are built up, can quickly describe reasoning and strategies declaratively
- Need to poll explicitly
- Need to implement a *reasoning source* for FreeCiv
  - Reasoning source makes assertions available on demand, from FAP's "perceptual system"

Centers of Gravity

- The source of a system's strength
  - Freedom of action, physical strength, will to fight
  - Take it out, and you cripple the system

What to pay attention to?

- Where are they?
- What are they probably doing?
  - If they are mounting an invasion force, a swift counter-strike can distract them.
  - If they are focusing on expansion, lure them into military actions

How to decide what to do next?

- Recall descriptions of tasks and activities from planning discussions
  - Create and maintain explicit representations of tasks and activities
  - Evaluate possible actions in terms of their value for current tasks and activities
  - Often nicely implemented via assertions, with TMS used to update beliefs as conditions change
    - Requires formulating interesting conditions to automatically compute from the FAP's perceptual system

Winning the Space Race

- What to represent?
  - How to organize the necessary production
  - How to defend yourself
  - How to spoil your competitor’s efforts

Organizing production

- Identify subset of cities that will be the parts-makers
- Use the rest for defense/spoilers
- Keep these activities distinct unless desperate!

Defending yourself

- Once it's clear the race is on, count on attacks
- See last week's lecture for more on defense

Spoilers

- One of your opponents is trying to build a starship, too.
  - Declare war, count on winning or at least delaying them more than you are delayed
  - Get allies to do it for you

Implications for AI design

- Representation, representation, representation
  - Good representations make it easy to think about a problem.
  - Good implementations of representations make it easy for your program to reason about its situation
  - More declarative descriptions reduce amount of code required, once enough infrastructure is in place.