Abstract

This document contains an overview of AI modifications in Freeciv – free clone of Civilization game, which is being created as a project for Artificial Intelligence for Games course. It describes some ideas we would like to implement in this game.

Introduction

Our course task is to implement some algorithms which will make AI to specialize in different cities. Freeciv itself is a multiplayer, turn-based strategy game based on Civilization. It is available for most operating systems. Freeciv is free and open source software that's why we can modify it and add our fixes.

The game starts in 4000 B.C. and allows us to lead chosen nation to global domination. During the gameplay we build new cities, expand our country, wage wars and build diplomatic relationships.

The game ends when we destroy all other nations or build a space ship and fly into space.

Considering how complex and big this game is we only work through little piece of AI which is responsible for cities improvements.

Ideas

Our basic idea is to define a few different type of cities (city profiles), make AI to choose one of them for each city controlled by specified player and try to follow this scenario as tight as it can. Notable fact is that we won't affect city placement (this is a thing to be improved by another team), but choose the type based on town surroundings and relative position to other cities.

City types defined by us:

- **Border cities** – these cities are focused on warriors and military preparations. They are placed near the borders and in case of attack they create a shield against enemies. In case of this city AI needs to focus on creating barracks, factories, and leads it's profile to military domination (Sparta !!!)

- **Food cities** – when most nearby tiles generate big amount of food, the city is dedicated to producing food and upkeep of units created in another towns. AI develops only such buildings, which are necessary for maintaining happy citizens and increasing food production.

- **Production cities** – similar as above, only this time the city is surrounded with production-boost tiles. It will mostly create new units and develop improvements further increasing productivity. City level increase is not top priority.

- **Trade cities** – once again this city's purpose is to maximize trade output,
which later is divided between tax, luxuries and inventions. For this kind of city there are also priority improvements.

**Decision strategies**

A type is assigned to the city based on few factors:

- **relative location** – cities considered to be on the borders of the country will become border cities with strengthened defenses and more military units residing in; in current implementation we only check if enemy city exists in certain radius;
- **all resource tiles nearby** – AI will check what kind of tiles is dominant in city area (21 fields), no matter if the area intersects with another city's area and part of the resources will become (or already are) unavailable at some point;
- **available tiles** – similar as above but resources reserved by another city will be excluded from decision making;
- **used tiles** – tiles which have citizen already assigned to, this parameter probably shouldn't have much importance since all cities begin development in similar manner;

All these parameters should be taken into account with different weights. If there are no improvements available for certain type, we don't affect AI's choice.

**Building classification**

Legend:
M – military
F – food production
P – production boost
T – trade boost
E – is building essential for city development?

<table>
<thead>
<tr>
<th>Name</th>
<th>M</th>
<th>F</th>
<th>P</th>
<th>T</th>
<th>E</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Allow airlifts; Produces veteran air units;</td>
</tr>
<tr>
<td>Aqueduct</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Allows city size &gt; 8;</td>
</tr>
<tr>
<td>Bank</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With Marketplace gives +100% luxury &amp; tax;</td>
</tr>
<tr>
<td>Barracks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Produces veteran ground units;</td>
</tr>
<tr>
<td>Cathedral</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gives 3(4) happy citizens;</td>
</tr>
<tr>
<td>City walls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Triple defense;</td>
</tr>
<tr>
<td>Coastal defense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Double defense vs. ships;</td>
</tr>
<tr>
<td>Colosseum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gives 3(4) happy citizens;</td>
</tr>
<tr>
<td>Building</td>
<td>Requires</td>
<td>Effect</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Courthouse</td>
<td></td>
<td>Corruption &amp; waste red.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factory</td>
<td></td>
<td>+50% shield production;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Granary</td>
<td></td>
<td>More food with city level change;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harbour</td>
<td></td>
<td>+1 food for all sea tiles;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hydro Plant</td>
<td></td>
<td>With Factory or Mfg. Plant gives +75% production;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Library</td>
<td></td>
<td>+100% science output;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marketplace</td>
<td></td>
<td>+50% luxury and tax output;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mass transit</td>
<td></td>
<td>Reduces pollution;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mfg. Plant</td>
<td></td>
<td>With factory gives +100% production;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nuclear Plant</td>
<td></td>
<td>With Factory or Mfg. Plant gives +75% production;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offshore Platform</td>
<td></td>
<td>+1 shield per each sea tile;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Police Station</td>
<td></td>
<td>Reduces unhappiness because of presence of military units;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port Facility</td>
<td></td>
<td>Produces veteran sea units;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Plant</td>
<td></td>
<td>With Factory or Mfg. Plant gives +75% production;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recycling Center</td>
<td></td>
<td>Reduces pollution;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research Lab</td>
<td></td>
<td>With library gives +200% science output;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAM Battery</td>
<td></td>
<td>2x defense with air attacks;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDI Defense</td>
<td></td>
<td>Protects from nukes; 2x defense vs. missile;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sewer System</td>
<td></td>
<td>Allows city size &gt; 12;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solar Plant</td>
<td></td>
<td>With Factory or Mfg. Plant gives +75% production;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stock Exchange</td>
<td></td>
<td>With Bank gives +50% tax and luxury production;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Super highways</td>
<td></td>
<td>+50% trade resource for all tiles with (rail)roads;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supermarket</td>
<td></td>
<td>+50% food production on each farmland tile;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temple</td>
<td></td>
<td>Gives one happy citizen;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Plan of code modifications
This section was written before actual coding. Its purpose is to show evolution of our design.

AI uses module called Domestic Adviser, which is also available for human player. It gives recommendations on city development. While human can ignore these tips, AI relies completely on them. Main decision making code is in aicity.c file. For us, the most interesting function is adjust_improvement_wants_by_effects. It is used for calculating so called “wants” for various improvements. Function checks what would be the effects of developing each improvement and decides which is the best right now. This is probably the part where we'll introduce major changes.

List of changes

Changes were made in 3 files: ai/aitools.c, common/city.h and common/city.c. We have decided to completely bypass the “want” parameter, because there is no precise documentation of it. aitools.c is the module, where ai_advisor_choose_building function is implemented. In case of city.h and city.c - we needed to add some fields to ai_city structure like building queue or city type and cooldown for type detection.

aitools.c

First of all, we have added functions responsible for choosing appropriate building according to city type: whether city is food-, shield-, trade- or military oriented. These functions are: ai_improvement_for_military, ai_improvement_for_food, ai_improvement_for_production and ai_improvement_for_trade.

Some improvements are grouped in queues, because player gains additional bonuses when some buildings are created together in one city.

Then, we created ai_determine_city_type function, which determines city type based on its surroundings. Algorithm itself is described in “Decision strategies” chapter. Additional “nonbuilding” cooldown is introduced. We do not want to completely prevent AI from creating units or some other buildings of its choice.

All coefficients (weights used in determining city type, cooldowns etc.) can be adjusted to suit future AI improvements.

ai_city structure gained following fields: city_type, city_type_cooldown and nonbuilding_cooldown. aicity.c contains additional lines of code for initializing described variables.

Choosing best building - details

Each function contains it's own array (corresponding to table showed above) with particular building numbers. We created auxiliary function to bind city id with its name and we get those results:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Airport</td>
<td>34</td>
<td>Stock exchange</td>
</tr>
<tr>
<td>1</td>
<td>Aqueduct</td>
<td>35</td>
<td>Super highways</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>Bank</td>
<td>36</td>
<td>Supermarket</td>
</tr>
<tr>
<td>3</td>
<td>Barracks I</td>
<td>37</td>
<td>Temple</td>
</tr>
<tr>
<td>4</td>
<td>Barracks II</td>
<td>38</td>
<td>University</td>
</tr>
<tr>
<td>5</td>
<td>Barracks III</td>
<td>39</td>
<td>Apollo program</td>
</tr>
<tr>
<td>6</td>
<td>Cathedral</td>
<td>40</td>
<td>A. Smith's Trading Co</td>
</tr>
<tr>
<td>7</td>
<td>City walls</td>
<td>41</td>
<td>Colossus</td>
</tr>
<tr>
<td>8</td>
<td>Coastal defense</td>
<td>42</td>
<td>Copernicus' Observatory</td>
</tr>
<tr>
<td>9</td>
<td>Colosseum</td>
<td>43</td>
<td>Cure for Cancer</td>
</tr>
<tr>
<td>10</td>
<td>Courthouse</td>
<td>44</td>
<td>Darwin's Voyage</td>
</tr>
<tr>
<td>11</td>
<td>Factory</td>
<td>45</td>
<td>Eiffel Tower</td>
</tr>
<tr>
<td>12</td>
<td>Granary</td>
<td>46</td>
<td>Great Library</td>
</tr>
<tr>
<td>13</td>
<td>Harbour</td>
<td>47</td>
<td>Great Wall</td>
</tr>
<tr>
<td>14</td>
<td>Hydro plant</td>
<td>48</td>
<td>Hanging Gardens</td>
</tr>
<tr>
<td>15</td>
<td>Library</td>
<td>49</td>
<td>Hoover Dam</td>
</tr>
<tr>
<td>16</td>
<td>Marketplace</td>
<td>50</td>
<td>Isaac Newton's College</td>
</tr>
<tr>
<td>17</td>
<td>Mass Transit</td>
<td>51</td>
<td>J.S. Bach's Cathedral</td>
</tr>
<tr>
<td>18</td>
<td>Mfg. plant</td>
<td>52</td>
<td>King Richard's Crusade</td>
</tr>
<tr>
<td>19</td>
<td>Nuclear plant</td>
<td>53</td>
<td>Leonardo's Workshop</td>
</tr>
<tr>
<td>20</td>
<td>Offshore platform</td>
<td>54</td>
<td>Lighthouse</td>
</tr>
<tr>
<td>21</td>
<td>Palace</td>
<td>55</td>
<td>Magellan's Expedition</td>
</tr>
<tr>
<td>22</td>
<td>Police station</td>
<td>56</td>
<td>Manhattan Project</td>
</tr>
<tr>
<td>23</td>
<td>Port facility</td>
<td>57</td>
<td>Marco Polo's Embassy</td>
</tr>
<tr>
<td>24</td>
<td>Power plant</td>
<td>58</td>
<td>Michelangelo's Chapel</td>
</tr>
<tr>
<td>25</td>
<td>Recycling center</td>
<td>59</td>
<td>Oracle</td>
</tr>
<tr>
<td>26</td>
<td>Research lab</td>
<td>60</td>
<td>Pyramids</td>
</tr>
<tr>
<td>27</td>
<td>SAM Battery</td>
<td>61</td>
<td>SETI Program</td>
</tr>
<tr>
<td>28</td>
<td>SDI Defense</td>
<td>62</td>
<td>Shakespeare's Theatre</td>
</tr>
<tr>
<td>29</td>
<td>Sewer system</td>
<td>63</td>
<td>Statue of Liberty</td>
</tr>
<tr>
<td>30</td>
<td>Solar plant</td>
<td>64</td>
<td>Sun Tzu's War Academy</td>
</tr>
<tr>
<td>31</td>
<td>Space component</td>
<td>65</td>
<td>United Nations</td>
</tr>
<tr>
<td>32</td>
<td>Space module</td>
<td>66</td>
<td>Women's Suffrage</td>
</tr>
<tr>
<td>33</td>
<td>Space structural</td>
<td>67</td>
<td>Coinage</td>
</tr>
</tbody>
</table>

At this point we assigned some of this numbers to arrays in functions like this:

```c
ai_improvement_for_military(...){
    int improvements[] = { 0, //airport
                           1, //aqueduct
                           3, //barracks
```
6, //cathedral
7, //city walls
8, //coastal defense
9, //colosseum
22, //police station
23, //port facility
27, //sam battery
28, //sdi defense
29, //sewer system
37 //temple

ai_improvement_for_food(...){
    int improvements[] = {
        1, //aqueduct
        6, //cathedral
        9, //colosseum
        12, //grannary
        13, //harbour
        29, //sewer system
        36, //supermarket
        37 //temple
    };
}

ai_improvement_for_production(...){
    int improvements[] = {
        1, //aqueduct
        6, //cathedral
        9, //colosseum
        11, //factory
        14, //hydro plant
        18, //mfg plant
        24, //power plant
        30, //solar plant
        19, //nuclear plant
        20, //offshore platform
        25, //recycling center
        29, //sewer system
        37 //temple
    };
    // (Hydro Plant + Mfg. Plant + Solar Plant + Nuclear Plant) = BONUS
    int scheduled_improvements[] = {
        14,
        18,
        30,
        19
    };
}

struct impr_type* ai_improvement_for_trade(struct city *pcity, struct impr_type *chosen) {
    int improvements[] = {
        1, //aqueduct
        2, //bank
        6, //cathedral
        9, //colosseum
        16, //market place
        34, //stock exchange
        10, //courtyhouse
        26, //research lab
        15, //library
        28, //university
        29, //sewer system
        35, //super highways
        37 //temple
    };
    // (Bank + Marketplace + Stock Exchange) = BONUS1
    // (Research Lab + Library + University) = BONUS2
    int scheduled_improvements1[] = {2,
As we can see despite existence of improvements arrays there are sometimes scheduled_improvements too. It is because when we build all the buildings from such package we get bonus. These scheduled_improvements arrays have higher priority than basic improvement, so they are considered first.

The algorithm basically iterates over scheduled_improvements, when it finds that some of impr. is built it gets next one from array, otherwise (no scheduled impr. or all built) it iterates over standard impr. and gets next free one. The result is returned, „want“ coefficient is assigned and control returns to Freeciv core.

City type evaluation - details

Most important part is being done in this function:

```c
void ai_determine_city_type(struct city *pcity){
    /* different weights for: all tiles, available tiles and worked tiles potential */
    const int ALL_COEFF = 1;
    const int AVAIL_COEFF = 4;
    const int WORK_COEFF = 2;
    const int PROD_TYPE_MAX = O_GOLD;
    const int ENEMY_RADIUS = 0;
    ...

    These are all adjustable weights and coefficients. First three are the weights of respective tiles types: production potential of all tiles around the city, tiles that are available (not being worked by another city) and tiles that are being worked by the city.

    if((nearest_enemy = dist_nearest_city(pcity->owner, pcity->tile, 1, 1)) != NULL) {
        xdiff = abs(nearest_enemy->tile->x - pcity->tile->x);
        ydiff = abs(nearest_enemy->tile->y - pcity->tile->y);
        distance = ( xdiff > ydiff ? xdiff : ydiff);
    } else {
        distance = ENEMY_RADIUS+1;
    }
```

We use already implemented function dist_nearest_city to find nearest enemy city and we calculate the distance. The metrics here is in how many moves the city can be reached by some unit.

```c
for(i = 0; i<CITY_MAP_SIZE; i++) {
    for(j = 0; j<CITY_MAP_SIZE; j++) {
        if((i != j) &&(i != -j)) {
            // check all resource tiles nearby
            for(k=0; k<PROD_TYPE_MAX; k++) all_prod[k] += pcity-
```
>tile_output[i][j][k];
// check available tiles nearby
if(pcity->city_map[i][j] != C_TILE_UNAVAILABLE)
  for(k=0; k<PROD_TYPE_MAX; k++) avail_prod[k] +=
pcity->tile_output[i][j][k];
// check used tiles
if(pcity->city_map[i][j] == C_TILE_WORKER)
  for(k=0; k<PROD_TYPE_MAX; k++) work_prod[k] +=
pcity->tile_output[i][j][k];
}
}
}

This loop iterates over available tiles around the city and sums their production output.

for(k=0; k<PROD_TYPE_MAX; k++)
total_prod[k] = ALL_COEFF * all_prod[k] + AVAIL_COEFF * avail_prod[k] + WORK_COEFF * work_prod[k];

Now for each production type (food, shields, trade) we sum their values for each tile type (all, only available, only worked) with weights defined earlier.

// choose best option
i = 0;
for(k=1; k<PROD_TYPE_MAX; k++)
  if(total_prod[k] > total_prod[i]) i = k;

if(distance <= ENEMY_RADIUS)
  pcity->ai->city_type = CITY_BORDER;
else
  pcity->ai->city_type = i;

And here is very simple loop choosing best option for certain city. Danger (enemy city nearby) overrides everything and mobilizes military production in order to eliminate the threat.

And here are the modifications made in ai_advisor_choose_building:

void ai_advisor_choose_building(struct city *pcity, struct ai_choice *choice) {
  const int TYPE_COOLDOWN = 20;
  const int NONBUILDING_COOLDOWN = 15;
  const int NONBUILDING_DURATION = 10;
  int city_type;

  struct impr_type *chosen;
  struct impr_type *tmp_impr;

  Here are the neccessary constants and variables.

  if(pcity->ai->city_type_cooldown > 0) {
    city_type = pcity->ai->city_type;
    pcity->ai->city_type_cooldown--;
  } else {
    ai_determine_city_type(pcity);
    city_type = pcity->ai->city_type;

This code checks for the type of the city. The type is checked only once per (in this case, constant can be adjusted by programmer) 20 turns to avoid rapid changes of city type.

Sometimes it is good to let AI build something it desires. Therefore there is a counter for how often the AI should be able to decide and how long this “freedom” should durate.

Nothing special here: plain switch..case checking for city type and using helper functions to determine what building should be created this time. If no improvements according to the profile are available, AI will be allowed to choose what it wants to produce.

**AI Tests**

It is crucial for us to think about some way to test our code. We found out how to make several AIs play against each other and we are going to use this scenario to test if our AI makes progress. We will place some computer players on the map and observe their actions as a viewer. This solution lets us check different strategies at the same time.

Unfortunately, AI is not very good at researching new technologies, which enable a lot of new buildings, therefore we can simply observe that city type detection indeed works and AI builds improvements as planned. Modified code
attached to this document includes our custom logging system (bypassing Freeciv logger – our way is more convenient), which allows to observe all decisions made.

**Original Freeciv vs. modified version**

We observed two games with 5 AIs. First one was played using original Freeciv, second one – modified games. Our aim was to determine if AI indeed builds more improvements in cities located in suitable location.

We have discovered that in original Freeciv AI follows in general following schema: build pure-military impr. and ones that enable city development. Rest is secondonary We won't see any banks, courts, plants etc. Very developed city may indeed posses more varied impr., but still they are not profiled according to their location. Example:

![City Image]

Obviously this city is a typical food city but there aren't many buildings which in fact support food production.

Our AI chooses buildings more efficiently, leading to better specialized cities. Of course when there are no buildings available which are compatible with city type, AI may develop in another direction. And this is the aspect, where AI should be modified next: better scientific development. Sadly, our solution is quite limited by the way AI researches new technologies. Simply it puts much stress on military discoveries. But still, now we can see some specialized buildings in cities much smaller than in original version. Back then, we could only find impr. like banks or stock exchange only in the city of size like 17. Now they appear much more early:
Future ideas

As mentioned before, AI requires certain changes, which will improve AI's long term planning. Besides city type detection, AI should create cities to maximize natural resources usage.

Second required improvement is better border city detection. This option is similar to “fortress cities” planned feature and could be integrated with it.

Summary

We have successfully modified Freeciv AI, therefore increasing its sophistication. This tweak is in fact only a first element of whole chain of changes to be implemented. Sadly, we are limited by semester length and we have to leave the rest to other coders.