THE UNIVERSITY of NEW MEXICO

"Starvation Evasion"

A Serious Game of Policy Making and its Possible Effects on the World Food Supply throuh 2050

Project Specifications

version 1.2

1: Overview

The past decade has seen a rising of reports on the future of the world's food supply. Many of these have emphasized the challenge of providing for a growing, increasingly wealthy population, or potential obstacles such as climate change. A growing number of projects throughout many countries are attempting to plan for a future that may be radically different than the present in difficult or impossible to predict ways.

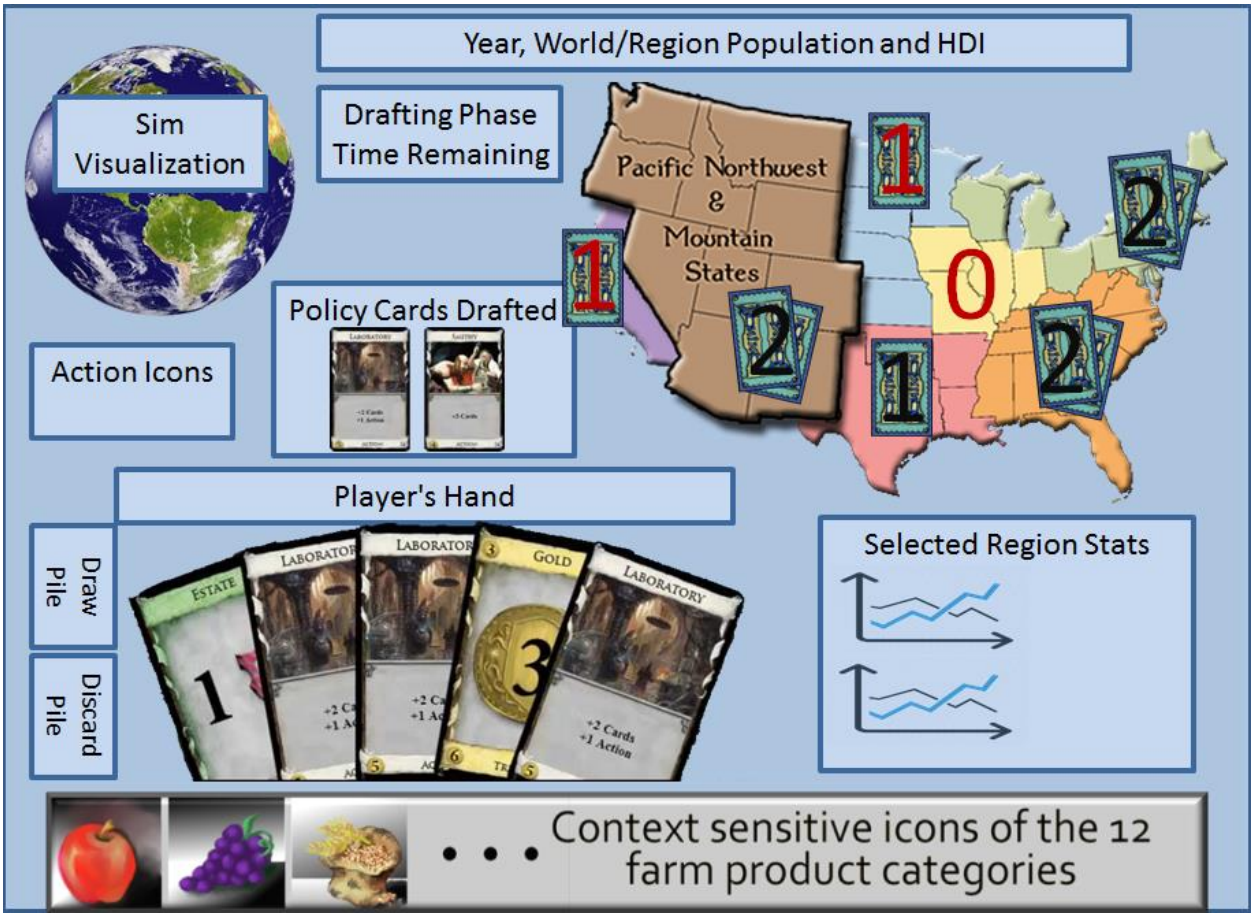
"Starvation Evasion" is a single player and multi-player, educational computer game wherein player(s) assume the role of state or regional policy makers governing, regulating, stimulating and facilitating agricultural activities within their region of the United States of America (with game expansions allowing players to represent and explore the interests other countries).

Underneath the game you will be implementing/extending a software simulator incorporating: (a) historic and future projections of world climate data from the World Meteorological Organization, (b) historic and future projections of world population distributions from the United Nations Economic and Social Affairs Population Division, (c) historic farm income, market demand, consumption, trade, location based yield, location based use of irrigation, pesticides, fertilizers, organic farming, GMO farming and conventional farming extent, type and distribution data from United States Department of Agriculture, The World Bank Agriculture and Rural Development Division, the European Commission of Agriculture and Rural Development, and other sources,(d) life expectancy, nutrition, hunger, education, and per capita income data for each of 194 countries from the United Nations Development Programme. The model incorporates and integrates this wealth of data into a complex adaptive system that attempts to accurately respond to player actions within the scope of world food including supply, demand, yield, and nutrition.

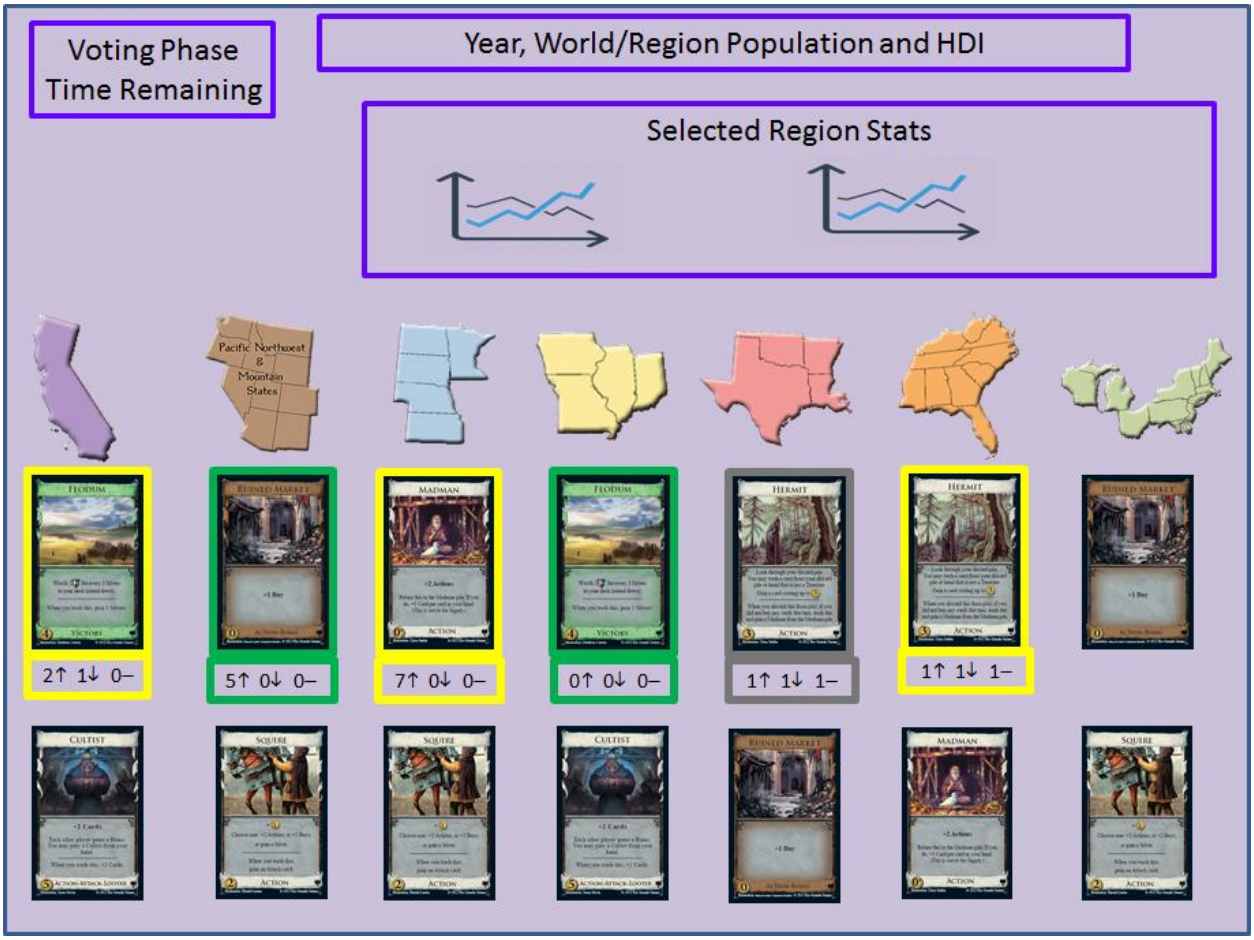
Game play occurs in turns wherein all players simultaneously draft policies, each from his or her "hand" of 7 available "policy cards." Drafted policies are played "face down". That is, when one player plays a policy card on his or her computer, each other player, on his or her computer, sees which players have drafted policies, but cannot yet see the content of those policies. After all players have drafted their policies (or pass or time has expired), all the face-down policies are revealed to all players. Any policies not requiring support are immediately enacted (meaning their direct effects are applied to game simulator). Any policies that require support are displayed in the voting pool and highlighted on each player's screen. As players vote to Support, Oppose or Abstain on a policy particular policy, the total counts are dynamically

displayed. Once a policy receives the required votes, it is applied to the simulator. Voting ends on a particular policy when it has the required support or when enough Oppose or Abstain votes have been cast so that it cannot go pass or when the total voting time (2 minutes) ends.

At the end of the voting phase, all enacted policies are applied to the simulator and the simulator calculates the complex web of interacting effects over the next 3 years. During the next policy phase players may use the GUI on their computer to view changes in world population, population distributions, world hunger distributions, crop yields by locations, crop market prices and other factors relating to or likely to affect agriculture.



Concept art of GUI for Policy Drafting Phase



Concept art of GUI for Voting Phase

While the simulator makes use of the best available source data and results from current supercomputer simulations, this game is a work of fiction. One game feature to help make this clear is that the game timeline begins in the past (1980). By beginning in the past, the players will create what are clearly fictional alternate recent histories, pass through a clearly fictional alternate present and on into what should be clearly understood as a fictional future. The education that we hope players glean from this game includes some facts about the present and past, but mostly is for the player to gain a better understanding of different possible effects of the same policy decisions when those decisions are made at different times, under different global environments in combination with different policies.

1.2: Winning the Game: Team and Individual Player Goals

Like in real world, in Starvation Evasion cooperation benefits everyone, yet each policy maker wants to see her or his own region benefited the most.

1. In this first version of the game, players can only play the part of policy makers within the U.S. The underlying simulation, however, models the world food market, world climate and world population. The main goal of the game is for the players to see the world through 2050 while avoiding widespread famine throughout any of the modeled world regions. If such a widespread famine does occur, the game ends and everyone loses. Successfully preventing widespread famine through 2050 is the win condition for all players.
2. The secondary goal is only achievable when all players have won by preventing widespread famine through 2050. In this case, the player whose region ends the game with the highest Human Development Index is hailed by all as a great leader and immortalized in song and sonnet.

1.3: Policy Drafting Phase

1) At the start of the game, each player gets a set of 7 policy cards chosen at random from a policy card deck specific to that player's region.

2) Each player may draft up to two policies from the policy cards in his or her hand.

3) When drafting a policy, the player may need to assign some variables on the card. For example, a card that says:

> Purchase in state commodity food for redistribution to in state low-income persons.
>
> Purchase A$ of X.

Requires that the player specify both A and X.

4) During a single turn, a player may not draft more than one policy card that requires support.

5) In place of playing a policy card, a player may immediately discard up to three cards and immediately draw three new cards. If the player still has a remaining policy draft action, then the player may draft either one of the new cards or one of those not discarded.

6) At any time during the Policy Drafting Phase, any player may discard one policy card. This card is not replaced until the draw phase at the end of the turn. Each player may only do this once per turn.

1.4: Policy Voting Phase

After all players have drafted their policies (or pass or time has expired), all the face-down policies are revealed to all players. Any policies not requiring support are immediately enacted.

Any policies that require support are displayed in the voting pool and highlighted on each player's screen. Different colors of highlighting indicate which players are eligible to vote on those policies. As players vote to Support, Oppose or Abstain on a policy particular policy, the total counts are dynamically displayed. Once a policy receives the required votes, it is applied to the simulator. Voting ends on a particular policy when it has the required support or when enough Oppose or Abstain votes have been cast so that it cannot go pass or when the total voting time (2 minutes) ends.

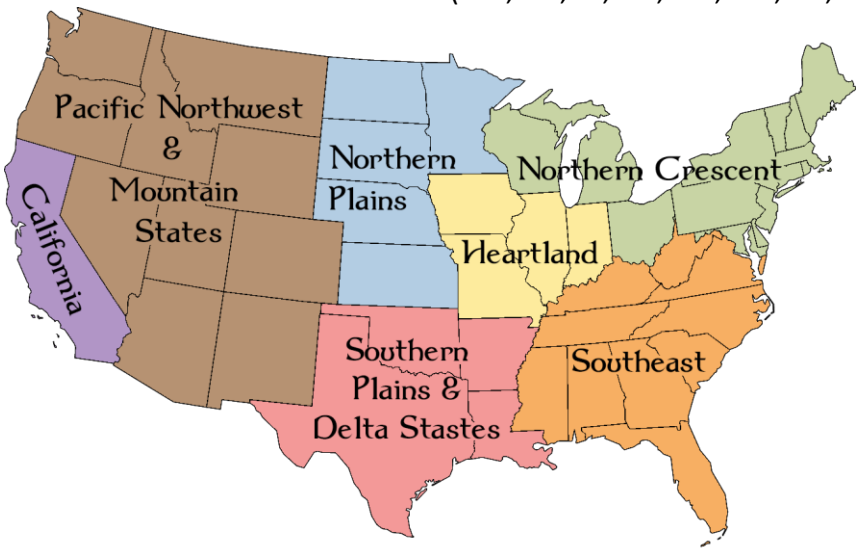
1.5: Policy Draw Phase

1) The simulation deals random cards from each player's policy card deck until that player has 7 policy cards in hand.

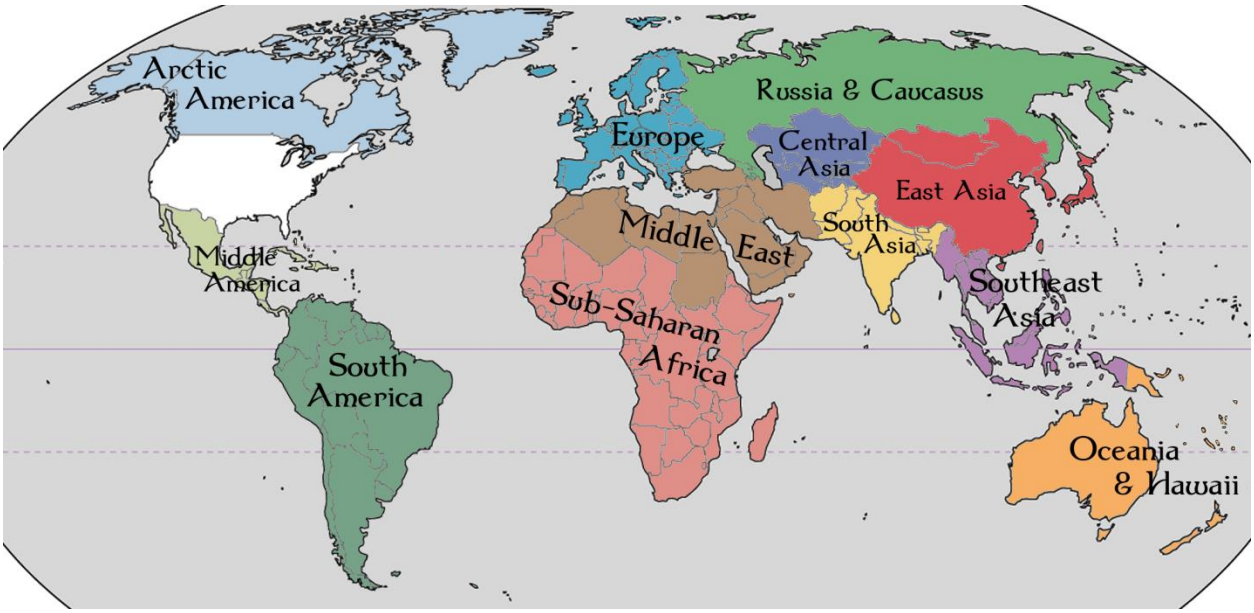
1.6: Game Environment

1) This version of the game must support 7 player regions. Each player region is a continuous land area of the continental U.S. composed of one or more states so that each has approximately equal total net farm income as measured in 2014:

    1) California (CA)
    2) Heartland (IA, MO, IL, IN)
    3) Northern Plains (ND, SD, NE, KS, MN)
    4) Southeast (FL, MS, AL, GA, TN, NC, SC, KY, WV, VA)
    5) Northern Crescent (ME, VT, HN, MA, RI, CT, NJ, NY, PA, DE, MD, OH, WI, MI)
    6) Southern Plains & Delta States (TX, LA, OK, AR)
    7) Pacific Northwest & Mountain States (WA, OR, ID, NV, MT, WY, UT, CO, AZ, NM)



Player Regions



Non-Player World Regions

2) The GUI displays aggregated population, crop production and trade into 12 Non-Player world regions: Arctic America, Middle America, South America, Europe, Middle East, Sub-Saharan Africa, Russia & Caucasus, Central Asia, South Asia, East Asia, Southeast Asia and Oceania.

3) The game focus is on major U.S. farm related policy, income and trade including major food crops, crops for biofuels, crops for animal feed, meat production and dairy production.

4) Players will NOT directly plan which crops are planted nor where they are planted. Players will not directly buy and sell farm products on the world market. All of this is done "under the hood" by the simulator.

5) Income for policy spending comes from a percentage of the region's net farm income.

6) Between turns, the simulator advances the clock by three years while it adjusts climate, population, crop yields, consumption, trade, etc. When spanning past years (1980-2014), the simulator includes actual occurrences of major droughts, early freezes, floods, hurricanes, earthquakes, wars and other events that have significant impact on the world food market. When spanning future years (2015 – 2050), significant events are generated randomly with probabilities affected by climate, land usage, each region's absolute and relative Human Development Index (HDI), etc. For example, if some combination of player policy, climate and economic assumptions result in monocropping extending over a large area, than that area will be more likely to experience a devastating blight, pestilence or insect invasion. The simulated effects of such events are limited to impacts on the world

food market: War disrupts production and increases human death rates. A huge surplus yield of wheat in Russia can cause a sharp drop in world wheat price, etc.

7) Most policies have direct effects that are applied during a single turn, but some have lasting direct effects. For example, a Loan Policy, if approved, might transfer 5 million dollars from the player controlling the New England states to the player controlling California on the turn it is played. Then, for the next 5 turns, 1.1 million dollars are automatically transferred from California back to New England.

1.7: Farm Products

The game models the following 12 farm product categories:

1) Citrus Fruits (Grapefruit, Lemons, Oranges, Tangerines, ...)
2) Non-Citrus Fruits (Apples, Apricots, Avocados, Cherries, Grapes, Olives, Strawberries, ...)
3) Nuts (Almonds, Pecans, Pistachios, Walnuts, ...)
4) Grains (Rice, Wheat, ...)
5) Oil Crops (Safflower, Sunflower, ...)
6) Vegetables (Beans, Potatoes, Carrots, Celery, Sweet Corn, Cucumbers, Pumpkins, Onions, Peppers, Tomatoes, Cantaloupe, Watermelon, ...)
7) Specialty Crops (Sugar beets, Mint, Mushrooms, Honey, Miscellaneous crops)
8) Feed Crops (Barley, Corn, Hay, Oats, Alfalfa, ...)
9) Fish (Fresh and salt water, wild caught and farm raised)
10) Meat Animals (Cattle, Hogs, Lamb...)
11) Poultry and Eggs (Chickens, Turkeys, eggs, ...)
12) Dairy Products (Milk and Cheese)

All farm products throughout the world used for food, animal feed and/or biofuel are modeled as belonging to one of the above 12 categories.

Farm products not modeled include Lumber, Cotton, and Wool.

2: Language, System Requirements and Deliverables

1) Each team must, throughout the project, make proper use of a Git version control system for software development (GitHub or Bitbucket).
2) The final code freeze is Monday, December 7th at midnight. For purposes of testing and grading, the version that will be pulled form each team's repository will be the last commit to the main branch dated before some unspecified time in the *very* wee hours of Tuesday morning.
3) The final project presentations will take place on Tuesday, December 8th from 12:30 to 2:30 PM in our usual lecture hall, 2018 Popejoy Hall (this is the time and location scheduled by UNM for our final exam and will serve that purpose). Other faculty from the CS department and some from the UNM Honors College will be attending the final presentations. Additionally, some community business partners as well as family and friends of the presenters are invited. Cookies, some healthier snacks and non-alcoholic drinks will be provided during the presentations.
4) Each ***individual team member*** must submit self/team evaluation form. This form is due in Blackboard Learn by midnight on Tuesday, Dec 8th.
5) One person from each team must submit slides used during the final presentation into Blackboard Learn by midnight on Tuesday, Dec 8th.
6) All source code must compile using Oracle's Java 1.7 JDK (no use of Java 1.8 extensions).
7) In addition to all source code and assets, each team's repository must include an executable JAR that runs using JRE 1.7. Note: If LWJGL 3 is used, then place Windows, MacOS and Linux natives in the same folder as the executable JAR and modify the JAR's manifest to include all dependent JAR files in the classpath so that double clicking the executable JAR runs the program.
8) In addition to any of the Java libraries included in Java SE, teams may freely use the LWJGL 3 (Lightweight Java Game Library) and Oracle's SAX library. Use of any other libraries requires explicit prior authorization from the course instructor. Note: As of JavaFX 2.2 and Java SE 7 update 6, the JavaFX libraries are installed as part of Java SE. It is preferred that JavaFX be used over Swing, but that is not a requirement.
9) While LWJGL does support game controllers and other input devices, the game must be designed to run on hardware likely to be found in an American high school computer lab (keyboard and mouse).
10) The game must run with adequate performance on the UNM Windows workstations in the Engineering and Science Computer POD 110 (these computers, overloaded with security, are likely to be much more like what high schools will have than are the workstations in the CS Linux lab).

11) Original art assets are being supplied by collaborators from other departments. Additionally, you may ask friends or family to create art assets provided they give explicit permission for those assets to be used in this game, posted on the Internet and allowed to become a continued game asset to be used and modified at the discretion of any future developers. Any third party images, sound or other resources (those made by friends, those downloaded from the Web or other sources) must be 1) clearly sited and 2) licensed for free redistribution with or without modifications to those resources. Premade art assets produced by teams will have no effect on the project grade. Particle effects, real-time procedurally generated art, artistic layout of graphical user interfaces, and artistic use of real-time visual effects including 3D and lighting effects all DO count towards the project grade.

12) All student developed source code must meet the CS-351 coding standards (see course website for details).

13) All classes and all ***public*** methods must include JavaDoc. You can use an auto generator for creating place holders, but it will be obvious if you do not edit the auto generated text. Class JavaDoc must explain how the class is designed to be used. Method JavaDocs must list and explain input, return values and any side effects. You may include explanations of how a method does what it does when such explanations seem likely to help future developers continuing this project.

14) Source code must be modular and follow good OOP design principles.

15) Teams may freely use and/or modify in any way code from the spring 2015 predecessor of this project. Executable JARs containing source code for these predecessors can be downloaded from the class website.

16) Game wide enums and constants: Country names, game regions, food product categories, and game constants are defined in the package **`starvationevasion.common`**. Whenever such values are needed within any student modules, the **`starvationevasion.common`** package must be used.

    The common repository is:
    https://github.com/castellanos70/StarvationEvasion

17) **Simulation Team**: One group will be making the simulator that all other teams must use **unmodified**. If groups fine problems with this simulator, they should post questions/comments in the Blackboard Learn discussions. The simulator team must develop using a public repository. All simulator code must be in the package **`starvationevasion.simulator.`** The simulation repository is:
    https://github.com/castellanos70/StarvationEvasion

18) **Visualization Team**: One group will be making a 3D visualization of simulator results and events. This will be displayed in a small window of each client's policy drafting and voting views. If the user selects this window, it must expand to a larger overlay panel

that darkens the background panel and is centered over the background panel. All game groups must use the **unmodified** If groups find problems with this visualization tool, they should post questions/comments in the Blackboard Learn discussions. The visualization team must develop using a public repository. All visualization code must be in the package **`starvationevasion.simviz`**

19) **Server Team**: One person will be making the server. The server must be started before any clients can connect. The server must be able to run headless or from a GUI. The server user must be able to queue a game accepting 1 – 7 players, instantiate the simulator and facilitate / verify interaction between the clients and the simulator. All source code for the server must be in **`starvationevasion.sever`** or in a subpackage that package.

20) **Game Client Team**: All teams not assigned a specific role (Simulation, Visualization, or Server) must create a game client. All source code of client must be in **`starvationevasion.`***teamname* or in a subpackage that package.

21) All image, sound, 3d models and other media assets must be in a subdirectory of <*projectroot*>/assets. Required and optional resource files can be found in:

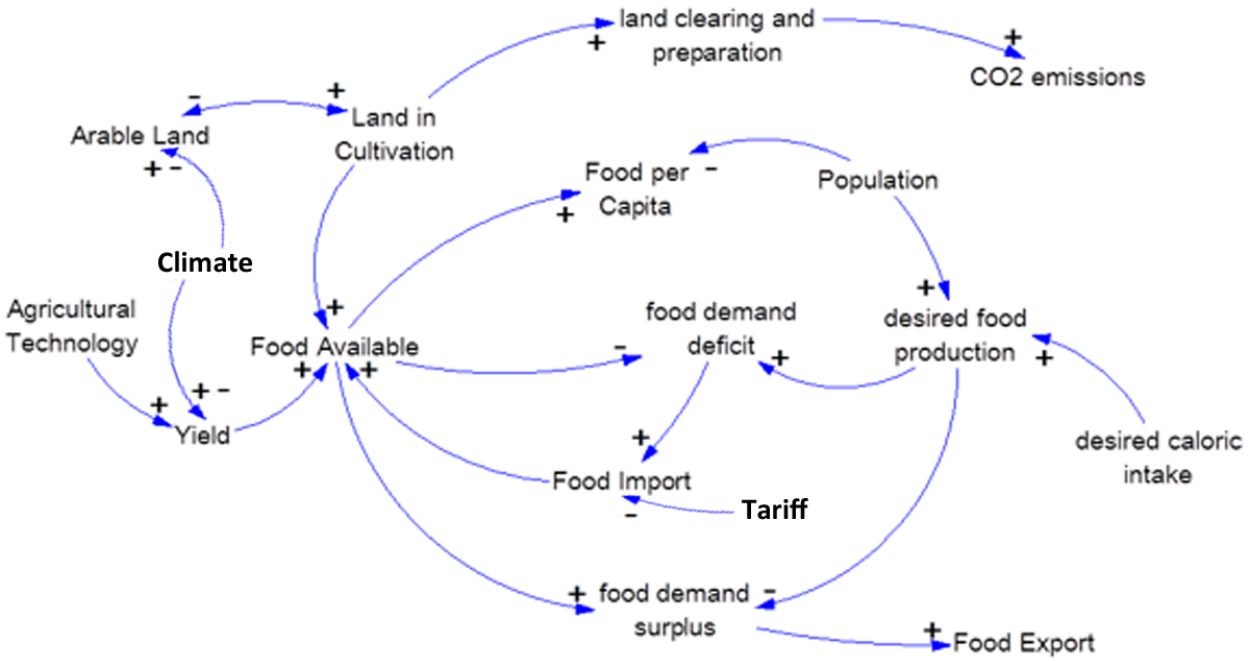
    https://github.com/castellanos70/StarvationEvasion

22) All data files must be in a subdirectory of <*projectroot*>/data. Required data files can be found in:

    https://github.com/castellanos70/StarvationEvasion

3: Simulator

1) Much of the simulator code form spring 2015 should be reused in this project.
2) From the user's prospective, one turn is three years. The simulator, however, must implement a timestep of exactly one year.
3) Many aspects of the simulator are interdependent. However, the use of differential equations are avoided in this zero-order simulator by calculating each year's parameters in a strictly specified order such that each parameter is a function of values from the previous year and from preceding calculations of the current year.
4) The model used it this game assumes the causal relationships in the image below. Arrows with (+) indicate a direct relationship and (-) indicates an indirect relationship.



3.1: Simulation Calculation Order

As can be seen in the arrowed diagram above, many values are interdependent. The simulator's timestep is one year. Calculations of each modeled category are performed each year is a strict order. Each calculation may depend on any category's value from a past year, but may only depend on current year values preceding the current category. The calculation order is:

1) Most Policy Card Effects: Any changes in land use, fertilizer use, and world trade penalty functions are calculated and applied.

2) **Changes in land use**: At the start of each simulated year, it is assumed that farmers in each region of the world each adjust how they use land based on currently enacted policies, the last year's crop yields and the last year's crop prices so as to maximize individual profit while staying within any enacted laws.

3) **Population**: In this model, each region's population is based only on data from external population projections and a random number chosen at the start of the game. Note that the occurrence of wide spread famine causes the game to end with all players losing. Thus, it is not necessary to model the after effects of a famine. Random game events such as hurricanes, typhoons, and political unrest are assumed to have negligible effect on population.

4) **Sea Level**: This depends only on external model data, a random value chosen at the start of the program and the year. Sea level only has two effects on the model: 1) higher sea level reduces costal farm productivity and increases damage probabilities of special storm events (hurricane, and typhoons).

5) **Climate**: In this model, climate consists of annual participation, average annual day and night temperatures and the annual number of frost free days on a 10 km $\times$ 10 km grid across all arable land areas of the Earth.

6) **Occurrence of Special Events**: Each year, there is a probability of each of many random special events occurring. These include major storms, drought, floods, unseasonable frost, a harsh winter, or out breaks of crop disease, blight, or insects. Special events can also be positive the result in bumper crops in some areas. While these events are random, their probabilities are largely affected by actions players may or may not take. For example, policies encouraging improving irrigation can mitigate the effects of drought, preemptive flood control spending can mitigate the effects of floods and major storms and policies that encourage / discourage monocropping can increase / decrease the probability of crop disease, blight, or insects problems.

7) **Farm Product Yield**: The current year's yield or each crop in each region is largely a function of the current year's land use, climate and special events as already calculated.

8) **Farm Product Need**: This is based on each region's population, regional dietary preferences, and required per capita caloric and nutritional needs.

9) **Policy Directed Food Distribution**: Some player enacted policies may cause the distribution of some foods to take place before the calculation of farm product prices and such distributions may affect farm product prices. For example, sending grain to very low income populations reduces the supply of that grain while not affecting the world demand. This is because anyone who's income is below being able to afford a product, in economic terms, has no demand for that product.

10) **Farm Product Demand and Price**: Product price on the world market and demand are highly interdependent and therefore calculated together.

11) **Food Distribution**: In the global market, food distribution is effected by many economic, political, and transportation factors. The Food Trade Penalty Function (see below) is an attempt to assign each country a single number that adjusts the efficiency of food on the global market being traded into that country (even if it originated in that country). The game simulation allocates food products to each country by maximizing the number of people feed under the application of the country's penalty function.

12) **Human Development Index**: Each country, based on the extent to which its nutritional needs have been met, has its malnutrition, infant mortality, and life expectancy rates adjusted. These are then used to calculate the country's HDI.

13) **Player Region Income**: Each player receives tax revenue calculated as a percentage of the player's region's total net farm income with any relevant enacted policy applied.

3.2: Simulator Data Granularity

The various data input into and calculated by the simulator are of three categories:

Global Data: Each global data item is a single number shared by all locations on the globe.

Year: This simulator uses a full year as a timestep. The game's current year is the year of every place on the globe.

Sea Level: In this game, sea level refers to the mean sea level (MSL), averaged over the current year at many points across the Earth's oceans. In this game, the sea level in 1980 is taken as the 0 cm point. For years 1981 through 2014, the sea level will always be historical difference from recorded in that year from the sea level recorded in 1980. For future years, projected values reported by the Intergovernmental Panel on Climate Change (IPCC) are given in ranges with increased variance for years more distant in the future. At the start of each game the simulator selects a random 2050 sea level (=h_{2050}), with a Gaussian distribution about the IPCC projected mean. During that game, for all years after 2014, the simulator must report the sea level as the value given by the quadratic equation passing through (h_{1980}=0, 0), (h_{2014},29) and (h_{2050},65).

Country and US State Data: The model requires as input and maintains data for each of the 193 countries with full membership status within the United Nations system as of Dec 31, 2014 (see http://en.wikipedia.org/wiki/Member_states_of_the_United_Nations) and This does not include, for example, Vatican City and Palestine (which the UN recognizes as "observer states"). Kosovo is recognized by 108 UN member states, but it is not itself a UN member state and, therefore, not in the list of 193 countries included in the model. Additionally, for the United States, the model requires as input and maintains data or each of the 50 states. In particular, required country and US state Data is:

1) Population (in people)

2) Average age of population (in years).
3) Population that is severely malnourished per 1,000 people.
4) Birth Rate (total number of live births per 1,000 people).
5) Net Migration Rate (immigration - emigration per 1,000 people).
6) Mortality rate (deaths per 1,000 people)
7) Life expectancy (years).
8) Net Farm Income for each of the 12 farm product categories (total revenue less costs).
9) Total cost for each of the farm product 12 categories.
10) Total square kilometer for each of the 12 categories.
11) Total production in metric tons of each of the 12 categories.
12) Total import in metric tons of each of the 12 categories.
13) Total export in metric tons of each of the 12 categories.
14) Percentage of land used for GMO, Organic and Conventional farming.

Region Data: While the internal simulation maintains data or individual countries and US states, the user's view of this data is limited to aggregations of this data into the 7 player regions of the continental US and 12 non-player world regions (see the two maps above).

Grid Data: Most of the climate data is maintained across all regional land areas (US and world) on a 100 square kilometer grid. Required grid data is:

1) Surface elevation
2) Annual maximum temperature.
3) Annual minimum temperature.
4) Average annual daytime temperature.
5) Average annual nighttime temperature.
6) Average number of frost free nights per year.
7) Average annual precipitation.

3.3: Simulator Data Types

User controllable country data: In this game, the enacted policy cards are only inputs directly controllable by the user.

Scenario Assumptions Data: This is all data that is completely independent of user actions. The scenario assumptions data comes from external sources of historical records and the results of external government simulators and forecasting. Scenario assumptions data includes both the data points researched form external sources and interpolated values from those sources for times and locations that are between the researched data points. The scenario assumptions data are:

1) Start year (1980) values of all Resultant Data that in later years is calculated.
2) Climate data for all arable land areas.

Derived Data: This is data calculated by the model from the past year's resultant country data, the current year's user controllable country data and the scenario assumptions data that is used as intermediate steps to calculate the current year's resultant country data.

Crop Zone Data: For each of the modeled crops (wheat, corn, rice and soy) find approximate values of the crop's ideal range of:

a) Annual maximum temperature (in degrees Celsius).
b) Annual minimum temperature (in degrees Celsius).
c) Average annual daytime temperature (in degrees Celsius).
d) Average annual nighttime temperature (in degrees Celsius).
e) Total annual precipitation (in centimeters of rainfall).

For the fifth crop (everything else) assume that whatever are the country's current average values are optimal for that country's "everything else". This is a poor assumption as in large countries it averages very different climate zones and "other" includes widely different crops; however, it is a starting point.

In milestone 2 of this game, it is assumed that crops can be planted in three different growth zone:

Ideal Zone: All 5 land statistics are within the crop's ideal range. Crops planted in an ideal zone produce at 100% of that country's yield rate for that crop.

Acceptable Zone: The land is outside the crop's ideal range in at least one of the crop's ideal requirements, but not more than ±30% out of the ideal range. Crops planted in this zone produce at 60% of that country's yield rate.

Poor Zone: The land is outside the crop's ideal range by more than ±30% in at least one of the crop's ideal requirements. Crops in this zone produce 25% of that country's yield rate.

For example, if rice's ideal precipitation range is 90 to 150 cm/year, then ±30% is:

$$90 - (150-90)\times 30\% \text{ through } 150 + (150-90)\times 30\% \quad = \quad 72 \text{ through } 168 \text{ cm/year}$$

Of course, some countries have well developed infrastructures for irrigation, greenhouses and other environmental modifications. This will be reflected in the country's yield rate. See below for calculation of the country's yield rate.

3.4: Model Requirements at Start of Game

Starting Land Use: The input to the model includes the square kilometers of land used for each of the 12 crops being modeled in 1980. This game will assume that each country's land is used optimally for the set of crops it produces in 1980 on a granularity of 100 square kilometers. The simulator must determine an optimal assignment of land with food products within the each country and each US region. This distribution will not, in general, have a unique solution. The model has no knowledge of the location of cities, protected areas, military areas, surface or ground water supplies or other geographical details within a country or US region. Thus, the "optimal" assignment of crop locations is based only on the 100 square km climate data and the crops climate needs.

The user should never be able to see where within a country or region the simulator as assigned the crops. The reason crops need to be assigned to a particular location is so that in the starting year, it can be known which crops are producing at that country's optimal or suboptimal level. This is important for calculating production rates in future years.

Crop Yield by Country: The input to the model includes the 1980 through 2014 square kilometers of land used for each farm product. It also includes the 1980 through 2014 production in metric tons of each crop. A country's yield for a particular crop is defined as:

$$yield\,(country, crop) =$$

$$= \frac{production}{idealLand + (acceptable\,Land \times acceptable\,Rate) + (poorLand * poorRate)}$$

where,

production is the metric tons of the crop produced by the country in 2014.

idealLand is the square kilometers of the crop assigned to ideal land.

acceptableLand is the square kilometers of the crop assigned to acceptable land.

poorLand is the square kilometers of the crop assigned top poor land.

See "Starting Land Use" for how ideal, acceptable and poor land is assigned. Also see "Crop Zone Data" of the "Climate, Crop, and Land Data" section for the three rates.

It is assumed that a country's yield for each crop in 2014 is its base yield. It is used in the start year of the game and remains constant throughout the game unless affected by player policies such as improving transportation or irrigation or GMO research.

Add Noise to Temperature and Precipitation: Independently, for each temperature and precipitation array, visit a randomly selected 10 percent of the land cells in a random order.

For each cell visited, calculate *delta* from the appropriate equation below.

$$delta = 0.1 \times (Tmax_i - Tmin_i) \times randomizationPercentage \times (r1 - r2)$$

$$delta = 0.1 \times (Tday_i - Tnight_i) \times randomizationPercentage \times (r1 - r2)$$

$$delta = 0.1 \times rain_i \times randomizationPercentage \times (r1 - r2)$$

Where, $r1$ and $r2$ are uniformly distributed random numbers from 0.0 through 1.0.

The first equation is use to when modifying the annual maximum temperature and again when modifying the Annual minimum temperature. The second equation is used when modifying the average daytime and average nighttime temperatures. The last equation is used when modifying the annual precipitation.

Modify all of the appropriate array's land cell values within a 100 kilometer radius of the visited cell by the equation:

$$array_k = array_k + \frac{delta}{\ln(e + d(i,k) \times r3)}$$

Where,

$d(i,k)$ is the distance in kilometers form the visited cell i to the neighbor cell k.

$r3$ is a uniformly distributed random number from 0.0 through 2.0. Note: choose a new $r3$ for each neighbor cell k.

Note: Since the locations are on a sphere, the actual distance between the two cells is a path along a great circle. However, that would be a silly amount of accuracy for adding random noise.

Food Trade Penalty Function:

The distribution of food around the world via trade is far from independent of the source and destination. Many factors affect the distribution of food including distance, access to sea ports, railways and other transportation, government tariffs, and government corruption. To evaluate the influence of trade policy on food distribution, Professor Jason Moore's the Big Data class of the UNM Honors College created a penalty function for how difficult it is to transport food into different countries. The data used for this came from the Food and Agricultural Organization of the United Nations and was for 2011. The specific data sets we examined included the food deficit per person per day in each country, the import values for each crop, and the calories per crop necessary to feed people. With these values, we found which countries had a surplus or deficit per year and how much food was needed to meet this caloric deficit. To determine the deficit or surplus, multiplied the minimum calories needed per day (1800 kcal) by the population and by 365.25 to find the calories needed for the entire country per year. Then we used the reported depth of food deficit per capita from the World Bank to find which countries lacked food. Some of the data for countries was missing from the World

Bank website, so to fill the missing data we used Index Mundi for the corresponding year of 2011. The other nations we examined experienced a food surplus. This was more difficult to calculate, because these values are not readily available. To create values for the yearly surplus per country, we used the minimum caloric need per country. We evaluated this along with the total amount of calories imported, to find how much excess food in the four major crops- corn, rice, soy, wheat- the countries had each year.

For a penalty function value based on a scale of -1 to 1, we divided each country with a food deficit by the value of the country with the greatest deficit (Turkey). This output a positive value as a portion of 1 to demonstrate the difficulty of importing that crop in the particular country. Similarly, for countries with a surplus, we divided the total surplus by the country with the greatest surplus per capita. The output for this was a ratio of negative 1 to demonstrate the ease of importing to countries that had more lax food import policies (tariffs). The negative value was required to place it on a scale to compare it to the deficient countries. The more negative value, the easier importing is, the more positive means the country has a greater difficulty importing this particular crop.

The resultant table giving each country's penalty function is in the repository:

https://github.com/castellanos70/StarvationEvasion/

In the file: **data/WorldData/Food_Trade_Penalty_Function.csv**

Use the following equation to calculate the amount of food product received per 1 million dollars in food product shipped:

$$received = \frac{shipped}{penalty + 2}$$

Human Development Index: The United Nations Development Programme uses what they call the Human Development Index (HDI) to emphasize that people and their capabilities should be the ultimate criteria for assessing the development of a country, not economic growth alone. The United Nations claims that HDI can be used to question national policy choices and stimulate debate about government policy priorities. The United Nations's HDI is a composite statistic of life expectancy, education, and per capita income indicators, which are used to rank countries into four tiers of human development. In the UN's 2014 report, the nations with the highest HDI are Norway (0.944), Australia (0.933), Switzerland (0.917), Netherlands (0.915), and United States (0.914). The 2014 lowest HDI countries are: Sierra Leone (0.374), Chad (0.372), Central African Republic (0.341), Congo (0.338) and Niger

(0.337).

This first version of the Starvation Evasion game estimates HDI as:

$$HDI = \frac{population - (ProteinEnergyMalnourished + MicronutrientMalnourished)}{population}$$

Where,

ProteinEnergyMalnourished is the number of people in the region who get less than 700,000 calories per year in carbohydrates, proteins, and fats where at least 10% is from protein.

MicronutrientMalnourished is the number of people who get 80% of calories from two or less of the 12 farm food products modeled in the game.

4: Game Client

4.1: Game Client Overview

Each player and each AI runs a game client on a networked computer. When the game client starts in human player mode, it should show a welcome animation. Then prompt the user to start a solo game or a multiplayer game. In both single player and multiplayer games, the game client is the player's interface to the game simulator, other players (both human and AI), and the simulation visualization module.

4.2: Game Client / Server Communication

In both single and multiplayer games, the client does not instantiate the simulator. Only the server may instantiate the simulator. The client always communicates with the simulator via a server using TCP/IP (not UDP) via the java.net.Socket API.

Selecting a solo game must open a new JVM that starts a server on "localhost" (IP=127.0.0.1). Selecting a multiplayer game must prompt the player for a server host name and attempt to connect to that host ***once per 15 seconds*** (NOT IN A TIGHT LOOP).

After the server connection is established, the client must display a staging screen that shows the seven player regions of the U.S., the server specified number of human players required by the hosted game with a live display showing which networked player(s) have already selected a region. Depending on the server specified mode, the client's player is either assigned a particular region or may select any region not already selected. After the specified human players have connected, any remaining regions are assigned to AIs and a 10 second countdown screen must be displayed.

4.3: Policy Drafting Phase

At the start of the game, the server deals 7 policy cards to each player. Policy cards are selected randomly form a policy card deck specific to each player region.

During the drafting phase, the client is responsible for displaying its player's hand of policy cards, processing user commands, any local validation of those commands, updating the client display with results of the local validation, communicating locality valid commands to the server, and updating the client display with data sent by the server.

The figure showing "Concept art of GUI for Policy Drafting Phase" is not a required layout, but does show required elements of the policy drafting phase:

Info Summery Bar: This must display the current game year, the population in the current game year of both the world and of the player's region, and the Human Development Index (HDI) in the current game year of both the world and of the player's region. This field must also display the current balance of the player's farming sector government revenue. All of these values are calculated by the simulator, and reported to the game client via the server.

Drafting Phase Time Remaining: The drafting phase has a time limit of is 5 minutes. The drafting phase ends when either all players click a button indicating they are done

playing policy cards or when the time limit expires. The Time Remaining widget must provide a digital display in minutes and seconds that starts green, becomes orange at 2:00 minutes and red at 1:00 minute remaining. Note: a requirement that states something is "green" does NOT mean it must have an RBG value of (0,255,0). There are many shades of green of which (0,255,0) is just one (and not a particularly pretty one). A requirement that something be "green" does not even mean it must be a single solid shade of green. It may be a gradient, or a texture, or pattern that generally can be described as "green". You are required to make something that looks good. There is significant latitude in that requirement, and what looks good in this is a function of other choices your group makes on the GUI, but looking good is NOT completely subjective and you will be held to objective aesthetic standards.

Sim Visualization: The simulation visualization is instantiated by the game client, passed game status form the simulator by the game client, passed user events by the game client and displayed by the game client. The simulation visualization must implement two update methods: updateMini(), and updateFull(). The first updates the small buffered image shown within a JPanel of the policy draft GUI. The second displays in a much larger game client created JPanel overlaid on the policy draft GUI. The game client must implement an intuitive method for the user to toggle between these displays.

U.S. Region Map: This must be an interactive regional map of the contiguous U.S. You do not need to implement the same graphic shown in the concept art but your map must implement the following features:

1) It must be a map clearly showing the region boundaries and the state boundaries within the regions.
2) The seven regions must be selectable with some type of animation, such as the selected section appearing to be lifted up above the rest of the map.
3) Region titles must display on mouseover.
4) While a region is selected, its title must be displayed.
5) There must always be some region selected and the default selected region must be the client's player's region.
6) At all times during the drafting phase, each region must clearly display the draft phase status of all seven players (whether human or AI). A player's draft status is the number of policy cards played face down (zero, one or two), the number of cards discarded, and whether the player has ended his or her policy draft phase.

Draw Pile: An interactive icon showing the facedown draw pile of the selected player. Normally, the player cannot interact with the draw pile. However, some policy cards allow the player to view some number of cards in target player's draw pile.

Discard Pile: An interactive icon showing the selected player's discard pile. The Icon must show the last card discarded faceup. It must also somehow indicate the number of cards in

the discard pile. The user must be able to somehow look through all the cards in the selected player's discard pile.

Player's Hand: An interactive view of the cards in the player's hand. Of course, the Dominion cards shown in the concept art will not be the cards used in this game. The player will never have more than 7 cards in hand. The fan view shown in the concept art is a suggestion, but not a requirement. Find some nice way to present a quick view of the hand. Ways in which the view must be interactive include:

a) On mouseover, the card must come to the front, be axis aligned and enlarged enough to clearly view the card image and all text.

b) On click, the client must verify the illegality of playing the clicked card. If the card cannot be legally played at the current time, then the client must display an appropriate error message. If the card can be legally played or discarded, then the card must be brought to the foreground, enlarged and axis alighted, and the player must be prompted to discard or draft the card.

c) If the player selects discard, and the card can be discarded, then the card must be moved to the top of the discard pile, faceup. If the player selects draft, the player must be prompted for selecting any targets or quantities required by the card, and after any such targets and/or quantities have been specified, the card must be moved to the Policy Cards Drafted area.

Selected Region's Statistics: The Selected Region's Statistics area must display charts population, HDI, balance of farming sector government revenue, and total cost and total revenue of each of the 12 farm products. The charts must include a selectable legend (when a product is unselected, it is removed from the chart). By default, a farm product for a particular region is selected if and only if it has >0 production). The farm product graphs must be able to toggle between displaying in millions of dollars and in metric tons of product. It may be that in the default size view, only some of this information can be displayed. The GUI must NOT display the text "Selected Region's Statistics" but rather be titled with "X", where "X" is the name of the selected region. The graphs must show values at the start of past turn and the current turn in the game (every 3 years). Each data point must be indicated as an open circles connected by straight lines. Mousing over the open circle must display its value.

Farm Product Category Toolbar: The game models 12 different farm product categories. Icons for these are being created by Beth Zahn. The icons can be found in the

`assets/farmProductIcons` directory of the

`https://github.com/castellanos70/StarvationEvasion` repository in sizes 64×64 and 256×256 pixels. The smaller sized images are, in general, not just resized, but cropped and then resized. The toolbar icons must be 64×64. Mouseover must display the category name. The toolbar must be context-sensitive: When the user

is being prompted to select a target farm product for a policy card, then clicking the icon must select the product. However, when not being prompted to select a product, clicking the icon must open an overlay panel that displays the 256×256 pixel image of the category, the category name, a list of farm products included in the category, a random question and answer from **`trivia.xml`** about something within the category, a graph of the price per metric ton of the category on the world market for each past year (not turn) of the game and a list of the five regions (US or world) that are the largest producers and the five regions that are the largest consumers of food products in the category. The panel must also display the ideal range of climate conditions for foods within the category.

Action Buttons: During the policy draft phase, the player may undo any card action up until one of the following: a) the player gains knowledge that he or she would not have had without having taken the action, b) the player clicks the button indicating he or she is done with the phase, c) the policy draft phase ends. The action buttons include an undo button and a button to declare the player is done with the phase.

Trivia Questions: Some policy cards offer a bonus when the player correctly answers a trivia question. The client is trusted to actually ask these questions and verify the answers. The game client must read these questions from the given **`trivia.xml`** file. These are multiple choice questions with 5 options. When the client asks a trivia question, the options must be displayed in random order:

https://github.com/castellanos70/StarvationEvasion/tree/master/data/trivia.xml

4.4: Policy Voting Phase

At the end of each policy phase, the client must display a voting phase panel. The figure showing “Concept art of GUI for Voting Phase” is not a required layout, but does show required elements of the voting phase.

There are 7 player regions. Every game will always have seven players (although 1 through 6 of them may be AI controlled). During the policy drafting phase, each player may draft no more than two policies and no more than one of which may be a policy that requires other players to vote. Thus, the GUI will need to display up to 14 policy cards of which as many as 6 could be cards that require a vote (the player who drafts a policy always auto votes for that policy). Some may be voted by all players and others may be voted by only some of the players.

Info Summery Bar: Same as in Policy Drafting Phase

Voting Phase Time Remaining: The voting phase has a time limit of is 2 minutes. The voting phase ends when either all required votes have been cast or when the time limit expires. The Time Remaining widget must provide a digital display in minutes and seconds that starts green, becomes orange at 1:30 minutes and red at 0:45 minute remaining.

Region Icons and Names: It must be visually clear to the player which policy cards have been drafted by which player regions. In the concept art for the voting phase, this was done by grouping each pair of cards with an icon representing the region. Whatever icon art used here should be consistent with the representation used in the U.S. Region Map used in the client's Policy Drafting Phase. It need not be sections of a map. It might, for example, be a collage of member state flags or a symbolic icon, but each must visually connect with a region on the map of the drafting phase, perhaps by matching a background color.

Selected Region's Statistics: Same as in policy drafting phase, except selected by selecting the region icon, name, or one of the region policy cards.

Policy Cards: In any single voting phase, there can be at most 7 policy cards requiring a vote by one or more players. Policy cards requiring votes must have three visually clear states:

a) Voting is open and may be voted on by the client's region.

b) Voting is open, but either the client's region may not vote or has already voted on this policy.

c) Voting is closed. Voting closes when it has the required number of votes or when it becomes impossible for it to receive the required number of votes.

Policy cards not requiring a vote should, of course, not have any of the above 3 visual cues.

Mouseover the card must enlarge its display for clear detail viewing and reading.

Vote Counts: Each policy that requires votes must display, in realtime, its vote count in Support, Oppose and Abstain votes. Mouseover the vote counts must display a list of which regions cast which votes. The Mouseover effect is not just to save screen real estate, but also to make the information obtainable with a slight effort, but not blatant. Do not have mouseovers auto disappear.

Action Buttons: There must be an intuitive way for the player to vote Support, Oppose or Abstain on each policy that player may vote.

4.5: Client Artificial Intelligence

There are 7 player regions. The game depends on each of these regions being played. In a single player game, 6 of the regions must be controlled by AI players. Like real players, the AI players must be given policy cards, must select and play policy cards, must vote on other player's policy cards and must cooperate with other players offering and deciding on tit-for-tat deals. Each AI

player is separate client JVM process running on the same or on a different computer. The AI player runs headless, communicates with the server using the same interface as human players.

4.6: Player to Player Interactions

Aside from voting, players must have other ways of interacting during both the policy draft and voting phases. Much of the interface is open to each development team to decide, but the following types of communication must be supported:

1) AI clients must be able to participate in the player interactions. This will be almost impossible to do with a simple chat window as the AI would need to process natural language communications.

2) A player must be able so show target card(s) from his or her hand to target player(s) or to all players.

3) When a player is showing a card to another player, the player holding the card must be able to show the card either before or after card targets have been selected. For example, if a card says “Each player may buy X million dollars of surplus commodity foods form his or her region for free redistribution in region Y”, then the player holding this card can show it to another player either before or after he has selected the value X and/or the region Y.

4) When showing a card, a player must be able to ask questions to target player or to all players including “will you support this?”, will you support this if I support something for you?”, “What would you like me to choose as X (where X is any target on the card)”.

5) The interface must support players being able to answer the required questions. Note: answers given through these player-to-player interactions are binding only in so far as players choose to make them binding. With the exception of the formal voting phase, the game must not inforce any player-to-player agreements.

6) The player to player communication system may be implemented in a separate window or within the drafting and voting windows.

5: Server

The server has the following main functions:

1) The server must be able to hosts a game allowing the user to specify the number of human players and the usernames of allowed clients.

2) When specifying specific allowed clients, the server user must be able to assign each of those users to a specific player region. The server user must also be able to allow clients to choose player regions first-come-first-choice.

3) The server must authenticate user logins with username and passwords. This must be done is a reasonably secure way as is appropriate for a game.

4) The server must instantiate a client running an AI for each region not controlled by a human.

5) The server must instantiate the game simulator.

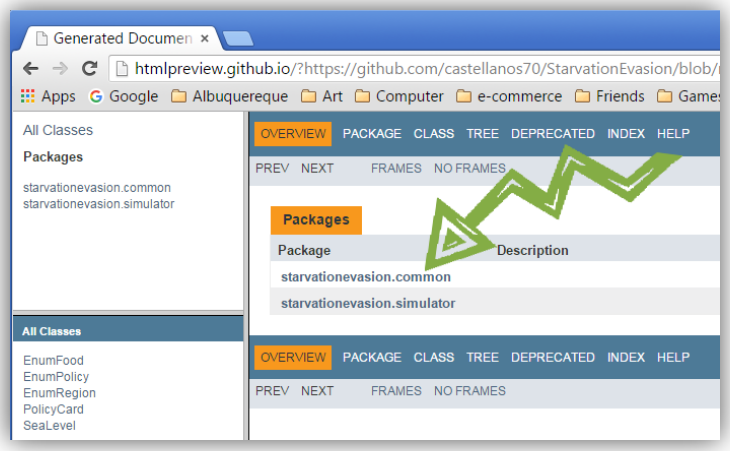
6) The server must manage all communication between each player client and the simulator and between different player clients.

7) If a client disconnects from the game, the server must continue the game with all that players actions assumed to become “pass”. Each “pass” action is triggered when all connected clients have finished their phase.

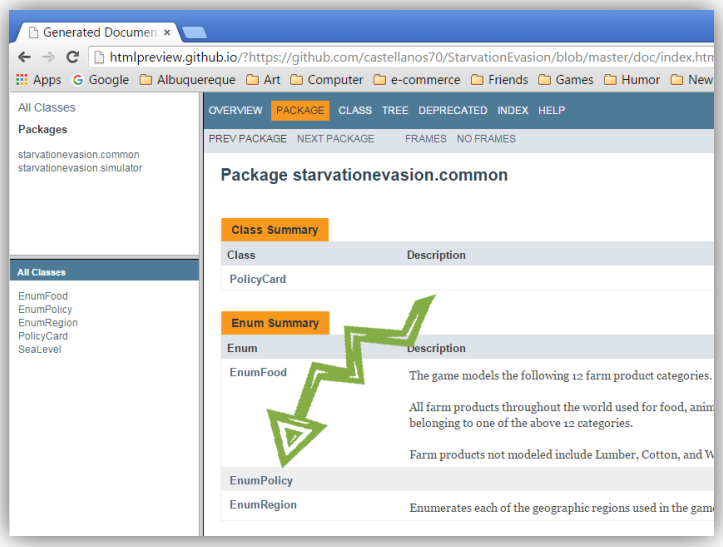
8) If a client reconnects to a game, then the server must return that client to control of the region to which it was previously managing.

9) The server must run on a 6th generation Intel® Core™ i7 computer with the simulator in real-time and not introduce human noticeable lag when seven human clients are connected via a dependable 18Mbps Internet speed. Note: For this performance test, the simulator will be confined to a single core.

6: Policy Cards: See http://htmlpreview.github.io/?https://github.com/castellanos70/StarvationEvasion/blob/master/doc/index.html



Click “starvationevasion.common”



Then click “EnumPolicy”