Rural Cultivation & Atmospheric Emulation Application (RCAEA) Design Document

|  |  |
| --- | --- |
|  |  |
| **Document:** | Design |
|  |  |
| **Authors:** | Zisis Damianidis  Richard Dyer  Tsanko Hadzhiev  Mihail Hadzhinikolov  Al-Mohaiminul Islam Khan  Raima Khan |
|  | Tsanko Hadzhiev |
|  | Richard Dyer |
|  |  |
|  |  |
|  |  |
|  |  |
| **Creation Date:** | 21/11/16 |
| **Last Revised:** | 22/11/16 |
| **Group Name:** | Tanks & Co.™ |

INTRODUCTION

Purpose of Design Document

The Design Documents describes and shows the details of the classes, methods and variables involved in building the application. The Class Description provides a description for all the classes. The Method Description clarifies some of the methods. In the interaction section some of the more abstract methods are shown how they will interact with each other.

Class Description

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Contains** | **Returns** |
| Simulation | The Simulation is in charge of playing, stopping, and restarting the simulation. | FileHeader, DataBase, Plot, Statistics |  |
| SimulationStorage | Uses the FileHeader to connect to the database and save or load a simulation. | FileHeader |  |
| File Header | Connects directly to the database and provides read and write functionality. |  |  |
| DataBase | On creation the DataBase object connects to the Applications database to specifically retrieve weather, and crop information. It will hold all information of crop and weather objects stored in the Applications Database. This object can always be used to retrieve a crop or weather object. |  | Weather, Crops |
| Weather | The weather object is created from the database and contains weather data properties that pertains to a particular province and month. |  |  |
| Plot | A plot is the center point where all the conditions and factors for a crops status are expressed. It keeps tracks of itself as the user moves through the timeline by creating and holding the necessary amount of PlotWeek objects. A plot can also calculate a crops status at any given time and return the details in a CropData Object. A plot can add and remove a crop the necessary week(s). In order to determine the status of a crop The Plot utilizes the database object from its base class the Simulation to retrieve weather and crop details located in the database. A Plot will also draw itself and determine when it needs to be redrawn based on the current date then simulation is on. | PlotWeek | CropData |
| PlotWeek | Represents the status of a plot in a given week. | Crop,Weather |  |
| Crop | Crops are originally created by the database. A Crop contains all its requirements to survive. It is also capable of creating and managing CropWeeks that is stored inside. | CropWeeks |  |
| CropWeek | Crop Week houses data of the crops status that pertains to a particular time in its growth. |  |  |
| CropData | CropData is created by the plot, it is a summary of information that useful to the user. It information that was created based on a particular crop, time, plot values, crop values, weather values, and user input such as watering options. |  |  |
| Images | Class “Images” contain four image for each crop |  |  |
| Statistics | This class generates a summary or specific information and interacts directly with the GUI to display information. | Simulation |  |
| Report | The Report generates a text file based on the simulation(s) object it will make use of the statistics object to create a useful detailed overview for the user. | Simulation |  |
| SoilType | The soilType object is created from the database and contains soil properties that pertains to a particular province. |  |  |
| RCAEA | RCAEA is a core class (manager class).It contains objects of all other classes. | Simulation, Report |  |
| Price | The Price object is created from the database and contains buy and sell price for each crop. |  |  |

Method Description

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Returns** | **Class** |
| **Simulation** | | | |
| + run( ) | The simulation runs from the start date until the end date. Run will update the currentDate variable based on the elapsed time from the start date. Unless paused, the plots will be updated based on the currentDate. |  | Simulation |
| +stop() | Stop the time objects which results into pausing a the running simulation. |  | Simulation |
| +getNumAtSpeficDate(Datetime date) | Takes the given date an returns an integer which is the number of weeks | int | Simulation |
| -CurrentWeekChanged() | Whenever the date(week) is changed this method raises SimilutionChangedEvent. |  | Simulation |
| +WeekToDate(int week) | Translates the given amount of weeks into a date object. | Datetime | Simulation |
| +LoadSimulation(string name, string province, string begindate, string enddate, string[] settings, List<Plot> PLOTS) | Loads all the necessary information needed from the database and assigns it to simulation variables. |  | Simulation |
| +AddCrop(Crop c, Plot p) | Adds selected crop object to the selected plot space |  | Simulation |
| +RemoveCrop(Crop c) | Removes selected crop object |  | Simulation |
| -DrawPlot(Plot p) |  |  | Simulation |
| **Simulation storage** | | | |
| +saveSimulation(string fileName) | Save Simulation checks to see if the simulation has made chances since its last save and then turns the simulation into a binary file,uses the FileHandler class to go into the database and save the current simulations name and object. |  | SimulationStorage |
| +LoadSimulation( string fileName) | Enters into the database using the FileHandler class and searchs for the given the FileName and turns the binary file into a simulation object and subsequently returns that object. | Simulation | SimulationStorage |
| **Plot** | | | |
| +AddCrop(Crop crop) | Checks if inserting crop is possible. If it is, it creates a Crop objects for every week and returns true. Else return false | Bool | Plot |
| +RemoveCrop(int atweek) | Checks if the selected plot is not empty. If it’s not , deletes the crop for each week and returns true. Else returns false |  | Plot |
| +getCurrentCropData() | Calculates the crop details and returns the information as a CropData object. This method however will take into account the currentDate from the Simulation base class. | CropData | Plot |
| +GetCropSummary() | Goes through the plotweek objects and for every week returns a list of cropdata objects | List<CropData> | Plot |
| +GetCropDataByDate(Date d) | Similar to the GetCurrentCropData method, this method returns information about CropData object for a given data. | CropData | Plot |
| -NurishLand(int week, Crop crop) | Depending on the selected amount of water to be added and the Weather Factors, this method calculates and assigns the amount of water. |  | Plot |
| -CalculateWeatherFactors(int week) | Changes the soil conditions for the given week. |  | Plot |
| -drawSelf( ) | Each plot will be in charge of drawing itself. As the simulation moves back and forth between weeks drawSelf will be called ONLY when the current image displayed needs to be changed based. |  | Plot |
| +deleteAllCrops() | Will firstly look at the surrounding Plots and if the surrounds plots hold the same type of crops then deleteAllCrops will also be called in the other plot. Until no crop is found. Then it will delete the crop in the plot all the weeks until its |  | Plot |
| -manageWeeks( ) | Monitors based on the start and end date how many PlotWeek objects are needed and creates/delete them as necessary. |  | Plot |
| -calBegintoEnd( ) | When a significant change has happened in a plot that will affect the rest of the plots time line. This method will be called to re-calculate the plot from the week 0 until the end. Such as Inserting a new crop in the middle of the timeline. |  | Plot |
| -calCurrentDate( ) | This methods considers the current date the simulation is at will call drawself if needed. This method is called when CurrentDate has changed. |  | Plot |
| -CalculateCropGrowth( int PlotWeek, int CropWeek, Crop crop) | This method gets the given crop maturity and creates 3 growth stages , depending on the stage crop image is changed. |  | Plot |
| -reasonOfDeath(Crop crop) | Returns a string with description of the reason of death for given crop, if the crop is dead. | string | Plot |
| **Database** | | | |
| -loadImage( ) | When the database object is created, this method will be called. Connects to the database and fills the databases list of images with images from the database. | List<Images> | DataBase |
| -LoadAllCrops() | Takes the whole information for ever crop and assigns is to the variables, returns a list of all crops. | List<Crop> | DataBase |
| -getImage(String CropName, int images number) | Finds the specific image requested by the user in the List of images the database class holds and returns an Image. | Image | DataBase |
| +getDefaultSoilType() | When the plots are first initialized they require a some values set by the SoilType, this method returns the non-user specified or default soil type, set in the database. | SoilType | Database |
| **Images** | | | |
| + getImage(int image) | Each crop has four images which shows different stages of a crop growth. This method return an image based on the number pass through the parameter. | Image | Images |
| **Crop** | | | |
| + setCropHealth(int addOrSubtractHealthPoints,int atWeek) | If the crop hasn’t been matured, this method set the crop health of each weak. |  | Crop |
| +SetInitialValues() | Set initial value of every crop week for each crop |  | Crop |
| **Statistics** | | | |
| +getTotalCostsByCrop(string crop) | This method calculates the total cost of a given crop passed as a parameter | decimal | Statistics |
| +getTotalWaterCostsByCrop(string crop) | This method calculates only water cost of a given crop passed as a parameter | decimal | Statistics |
| +getTotalFertilizerCostsByCrop(string crop) | This method calculates only fertilizer cost of a given crop passed as a parameter | decimal | Statistics |
| RCAEA | | | |
| + loadSimulation(string simulationName) |  | bool |  |
| + SaveSimulation() |  | bool |  |
|  |  |  |  |

Crop Rules

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Description** | **Death Rule** | **General Rules** |
| Maturity Length | The time (in weeks) it takes for the crop to reach full maturity. When a crop reaches maturity it will be automatically harvested. |  |  |
| Water Minimum | The minimum amount water the crop needs. A crop that can last longer without water has a higher minimum. |  | The lower the minimum water the more water needed to keep it alive. |
| Thirst | The amount of water absorbed why the plant each week. |  |  |
| Water Maximum | The maximum amount of water a crop can handle. |  |  |
| Sun Light Minimum | The minimal amount of weeks a crop can go without Needed Sunlight | If a crop goes over the sun light minimum amount of weeks below the needed sunlight |  |
| Temperature | The recommended temperature for survival Celsius | If weather temp is 5 degrees below temperature for 2 week, if weather temp is below 10 degree for 1 week. If weather temp is 10 degrees above temperature. |  |
| NutritionRate | The amount of nutrients absorbed each week. |  |  |
| Needed Nutrition | The amount of nutrition from the soil needed. | If the soils nutrition is below the needed nutrition for 2 weeks. |  |
| Needed Sunlight | The amount of sunlight needed per week to survive | If a crop goes over the sun light minimum amount of weeks below the needed sunlight |  |
| Water | The amount of water currently held by the crop | If water is above the WaterMaximum or below the Water Minimum for 2 weeks. | Will be added from rain. Or from selected watering rate (water given from watering rate will depend on current water and water minimum and maximum)  Water decreases each week based on the Soils Water Saturation and crops thirst |
| Maturity |  | When maturity is -1 the crop is dead | Increased 1 per week |
| Image Changed | True if the crops image has changed from the previous or following week. For example reached the next stage of growth or dead. |  |  |
| Current Image | The current image number to display for the crop |  |  |
| Crop Week | The number of Crop Weeks are determined by the maturity length. |  | If a crop dies before it reaches maturity |

Plot Rules

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Description** | **Decrease Rule** | **General Rules** |
| SoilType | Sets the initial values for a plot (week 0) |  |  |
| Is Empty | If a crop is in the Plot for a specific week. |  |  |
| Water saturation | The amount of water in the soil. | Decreases each week by weather temperature, crop thirst, and saturation lose rate. |  |
| Saturation lose rate | The ability for the Soil to hold water. |  | A plot with a Lower saturation lose rate keeps more water in high temperatures. |
| Soil nutrition | The amount of nutrition in the soil. | Absorbed each week by a crop. | Increased by user input fertilizer given. Amount given is based on current soil nutrition and maximum soil nutrition. |
| Maximum Soil Nutrition | The maximum amount of nutrition that a crop can handle. |  |  |
| Plot Week | The number of Plot weeks are determined by the beginning and end dates. |  | When beginning or end dates are changed the number of plot weeks and position of those plot weeks within a plot are accordingly adjusted.added/deleted. |

Exception/Error Rules

|  |  |  |
| --- | --- | --- |
| **Exception** | **Description** | **Solution** |
| Crop Dies | A plot has detected that a Crop has died before it has reached maturity. | If the application has detected a crop will die during the course of its life, the plot will turn red until the end of the Crops maturity. If the user clicks on a red Plot in the information panel it will give the reason for its death. |
| DataBase Object cannot connect to the database. | When the application starts or if a call to the database is made somewhere during the run time of the application. |  |

Interaction

Sequence diagram – adding a crop

|  |
| --- |
| **Sequence Diagram - adding crop.png** |

Sequence diagram – adding a crop

|  |
| --- |
| **Sequence Diagram - Starting application.jpg** |