

APPENDIX A

INSTALLATION GUIDE

REQUIREMENTS

The program was developed using Microsoft's .NET framework and as such is not natively supported on operating systems other than Windows. Running .NET applications on other operating systems will require third party software, such as *Wine* for Unix-like distributions.

For Windows users' .NET 4 or higher has to be installed for the program to run. See Microsoft's website for the web installer. Supported operating systems at the time of this writing are:

- Windows XP Service Pack 3
- Windows Vista Service Pack 1 or later
- Windows 7 Service Pack 1
- Windows 8 (.NET 4.5)

WINDOWS INSTALLATION

The software is packaged as a .ZIP file and users have to extract the content in order to run the application. Windows is distributed with necessary software to extract .ZIP files. After extracting the contents, there are no additional requirements and the user only needs to run FHG2A.exe.

USER MANUAL

Running FHG2A.exe will automatically start up the default simulation which is a 9x9 cell world, with 3 bands each starting with 20 members. The initial screen is the grid screen which displays a graphical representation of the world. The world consists of three zones, each of which is represented with a different colour in the grid (green, yellow, red). The bands are represented by grey circles which will move around the world as the simulation runs.

Each cell contains up to four numbers. A cell which is occupied by a band will have four numbers while an unoccupied cell will only have three. In an occupied cell the first number represents the number of members in the occupying band, the second, third and fourth corresponds to the prey, barley and farm values respectively. In an unoccupied cell the first, second and third number corresponds to the prey, barley and farm values in that order. The values corresponding to the food resources can either represent the Net Acquisition Rate (NAR) or the concrete value ($\frac{mass}{km^2}$ for plants or $\frac{number}{km^2}$ for prey).

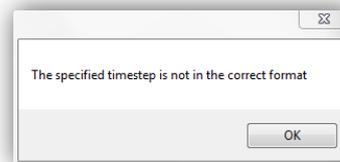
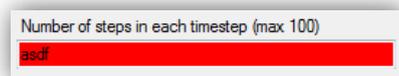
The interface and the tasks it accomplishes are explained on the following page.



The program after being launched – the default settings are displayed.

CHEAT SHEET (GRID VIEW)

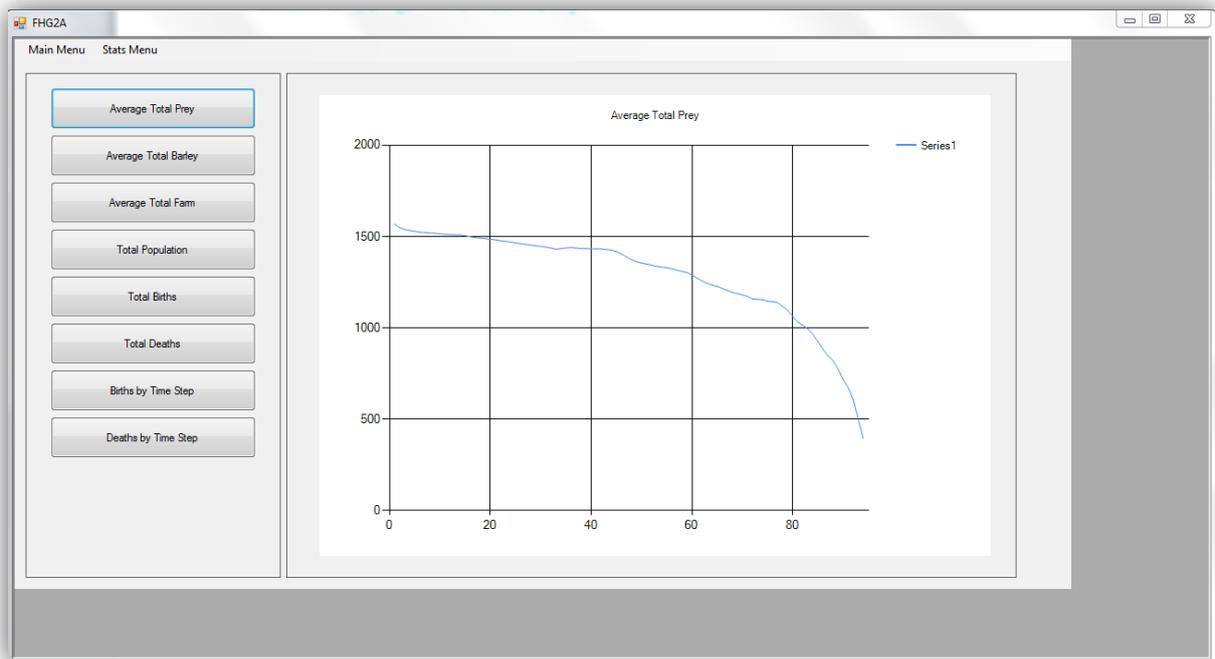
Run Time step. A simulation iteration is made when the button is clicked. The default amount of time steps during an iteration is 1 if the user does not enter text into the textbox. If a user enters a value less than 0, the time step is set to 0. If a user enters a value greater than 100, the value is set to 100. Any integer in the range [0, 100] is acceptable. Non-integer input prompts visual feedback indicating incorrect input.



Run Automatically. The simulation can be run automatically. To adjust the time steps for an iteration, the slider has to be used. Moving the slider to the right increases the time steps per iteration. The left most value is 1 time step, the rightmost value is 100 time steps.

Statistics. Switches between the grid view and statistics view. The statistics view is discussed separately.

Show NAR value / Show count value. The individual cells either display the NAR value or the concrete value. Clicking *Show NAR value* switches to the NAR value being displayed and changes the button to *Show count value*. Clicking *Show count value* does the opposite.



The statistics view of the program.

CHEAT SHEET (STATISTICS VIEW)

Average Total Prey. Displays the average number of prey as the number of time steps increase. This is calculated as the total number of prey in the world averaged over the number of cells.

Average Total Barley. Displays the average number of barley as the number of time steps increase. This is calculated as the total barley count in the world averaged over the number of cells.

Average Total Farm. Displays the average amount of cereal farmed as the number of time steps increase. This is calculated as the total amount of cereal farmed in the world averaged over the number of cells.

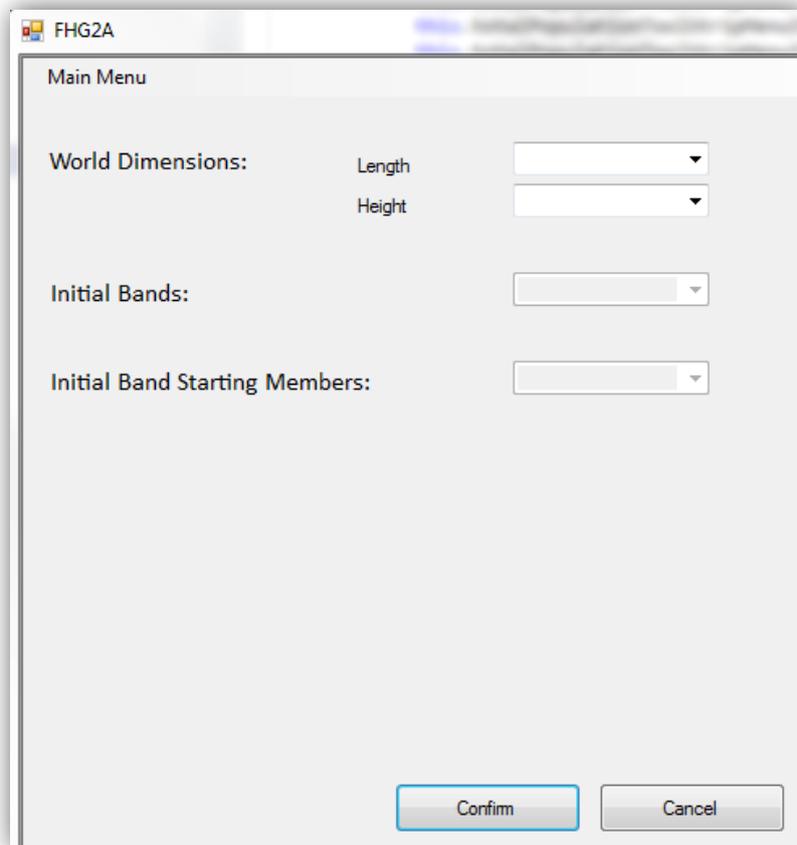
Total Population. Displays the population count of all the band members combined per time step.

Total Births. Displays a cumulative graph of the total amount of births as time steps increase.

Total Deaths. Displays a cumulative graph of the total amount of deaths as time steps increase.

Births by Time Step. Displays the number of band member births per time step.

Deaths by Time Step. Displays the number of band member deaths per time step.



The view when creating a custom simulation.

CHEAT SHEET (CUSTOM SIMULATION VIEW)

World Dimensions:

Length. Specifies the number of vertical cells to be used for the virtual environment. The value can be between 3 and 21, inclusive. If a number not divisible by 3 is selected, the cells will not be equally distributed over the environment.

Height. Specifies the number of horizontal cells to be used for the virtual environment. The value can be between 3 and 21, inclusive. If a number not divisible by 3 is selected, the cells will not be equally distributed over the environment.

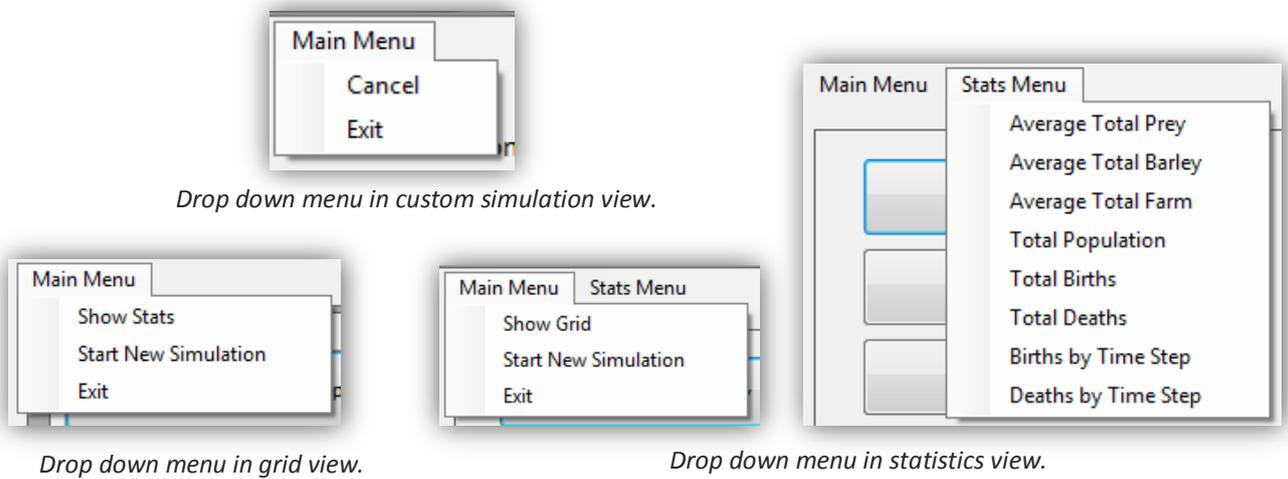
Note: *until the world dimensions have been entered in, the other editable options are locked.*

Initial Bands. Sets the number of bands the simulation starts with.

Initial Band Starting Members. Sets the number of members allocated to each band at the start of the simulation.

Confirm. The user accepts the custom settings and the new simulation is initiated. Data from previous simulations will be lost. The user is returned to the grid view of the new simulation.

Cancel. The user cancels the creation of a new simulation. The previous simulation is not lost and the user is returned to the grid view of the previous simulation.



CHEAT SHEET (MENU NAVIGATION)

Show Stats. Switches the view from grid view to stats view. This option is only available within grid view.

Show Grid. Switches the view from stats view to grid view. This option is only available within stats view.

Start New Simulation. Switches to the custom simulation creation screen.

Exit. Terminates the program. Running simulations cannot be recovered after terminating the program.

Cancel. Cancels the creation of a custom simulation. The user is returned to the grid view of the previously running simulation.

Stats Menu. A user can choose any of the graphs listed in the drop down menu. The graphs are explained on the previous page.

DATA LOGS

As the simulation runs the data of the world is written into five separate data log files. These files are stored in the same folder as the FHG2A.exe and are simple text files. The purpose of these files is to allow users to go and look at exact figures in the data or for the data of multiple runs of the simulation to be kept. By copy pasting the data logs to a different folder one can store the data of a simulation run for future reference for anyone who wants to see the results of your simulation. Note that the data logs are written over when a new simulation is started; this includes opening the program or using the new world creation screen within the program.

The format for each data log file is provided to allow reading of the logs.

CellBarley/Farm/Prey – these files contain the food resource for each cell for each time step.

Format: [TS: (Time Step Value) COORD: (cell x-coordinate),(cell y-coordinate) Val: (Concrete Value of the resource) NAR: (Net Acquisition rate of the resource)]

CellFile – Contains the data for each cell for each time step

[TS: (time step value) COORD: (cell x-coordinate),(cell y-coordinate) PreyVal: (Concrete value for the cell's prey) PreyNAR: (net acquisition rate value for the cell's prey) BarleyVal: (Concrete value for the cell's barley) BarleyNAR: (net acquisition rate value for the cell's barley) FarmVal: (Concrete value for the cell's farm) FarmNAR: (net acquisition rate value for the cell's farm)]

StatsFile – contains aggregated data for each time step

[TS: (time step value) ATP: (Sum of every cell's prey divided by number of cells) ATB: (Sum of every cell's barley divided by number of cells) ATF: (Sum of every cell's farm divided by number of cells) TP: (Total population of the world for that time step) TB: (Total births that have occurred in the world) TD: (Total deaths that have occurred in the world) BBT: (births that occurred that time step) DBT: (deaths that occurred that time step)]

PACKAGE: Engine

CLASS: Band

moveToBestCell(ref Cell[,] cells)

This method encapsulates the logic behind how a band selects a cell to live in for the next time step. It governs what the band sees as an optimal cell as well as how these optimal cells are chosen when they are of equal value.

consume()

This method encapsulates the behaviour of how the band attains its energy. It contains the logic that governs what foods to eat and in what quantities as well as when the band should consider farming.

breedOrDie(float netEnergy, Random birthRandom)

This method contains the functionality for splitting the band into two when necessary as well as deciding when the band has lost members because it was unable to gather enough energy

PACKAGE: Engine

CLASS: Barley

The Barley class is just a simple extension of the FoodResource class and contains no methods of note. It exists in order to create a specialised member of the FoodResource class.

PACKAGE: Engine

CLASS: Cell

calculateScore()

This method updates the score values of the cell's resources which are used by the bands to judge the cell.

addBand (Band newBand)

Adds a band to the cell provided the cell was not already occupied.

removeBand()

Removes the current band from the cell.

PACKAGE: Engine

CLASS: Engine

createNewWorld(int length, int height, int initialBands, int startingMembers)

This method checks the validity of a new simulation before creating a new World class with valid data.

runTimestep()

Increments the time step counter that exists in the engine class and then tells the World to update itself. Once the world is updated this method calls the *FileManager* class to write the Worlds data to the data logs.

PACKAGE: Engine

CLASS: Farming

grow()

A specialised growth function, sets the farm value to its maximum value.

calculateNAR()

Specialised NAR calculation, the farm resource has a static NAR value that is returned since its beginning value is always its max value.

calculateEnergy(float energyNeeded, float timeAvailable)

Specialised energy calculation for the farm resource as it contains unique variables for use in this calculation.

PACKAGE: Engine

CLASS: FileManager

writeWorldToFiles()

Writes the worlds current data to the data logs. Includes the logic behind the creation of the statistical data.

readFromFile()

Reads in the data that has been written to the data logs in order to allow for the statistics to be displayed for every time step without having to store the data in active memory.

closeWriterStreams()

Closes the writer streams so that the class can be safely destructed or so that the files can be read from instead.

PACKAGE: Engine

CLASS: FoodResource

grow()

Increases the current value of the food resource according to the default growth algorithm as explained in the paper.

calculateNAR()

Calculates the Net Acquisition Rate (NAR) of the food resource according to the algorithm provided.

calculateEnergy(float energyNeeded, float timeAvailable)

Calculates the energy that a band is able to acquire from the resource for a time step based on the standard variables of a *FoodResource*

repopulate(int viableCells, int chance)

This function handles the repopulation of the *FoodResource* according to the standard algorithm provided.

PACKAGE: Engine

CLASS: Habitat

The Habitat class is simply for storing data relevant to many cells and contains no methods of note.

PACKAGE: Engine

CLASS: Prey

The Prey class is just a simple extension of the FoodResource class and contains no methods of note. It exists in order to create a specialised member of the FoodResource class.

PACKAGE: Engine

CLASS: World

runTimestep() This method calls the appropriate methods of each cell and band as well as keeps track of dead and newly created bands in order to modify the band data structures at the end of the time step.

placeInitialBands() Contains the logic behind how the initial bands are placed, since the bands move immediately their placement isn't actually that important provided they do not occupy the same cell.

createCells() This method creates the initial cells as well as contains the logic of how these cells are assigned a habitat.