EKoSim

An ecological simulation game
For the K Desktop Environment

Project brief

EKoSim programming workshop – summer 2003
Uri Blatt 034211284

Table of Contents

1. Description

EKoSim is an ecological environment simulation game for the K Desktop Environment.

EKoSim can be used for educational purposes for demonstrating the Lotka-Volterra mathematical model of interaction between different species in an environment and displaying the output through the graphical user interface.

In EKoSim the user can choose between several scenarios, adjust the parameters affecting the environment behavior, and adjust the virtual time "speed". According to the user defined preferences the appropriate model is chosen for the simulation. This flexibility allows for the creation of many different environmental situations so that each time the simulation is run the outcome is necessarily different.

The output of the simulation is displayed on the screen as an above view of the environment with the life forms moving on the ground and two graphs displaying the changes across time in the quantity of the individual life forms.

This document serves as a user manual, a description of the program structure and as a guide to further development.

EKoSim User Manual

2. Setup

2.1. Requirements

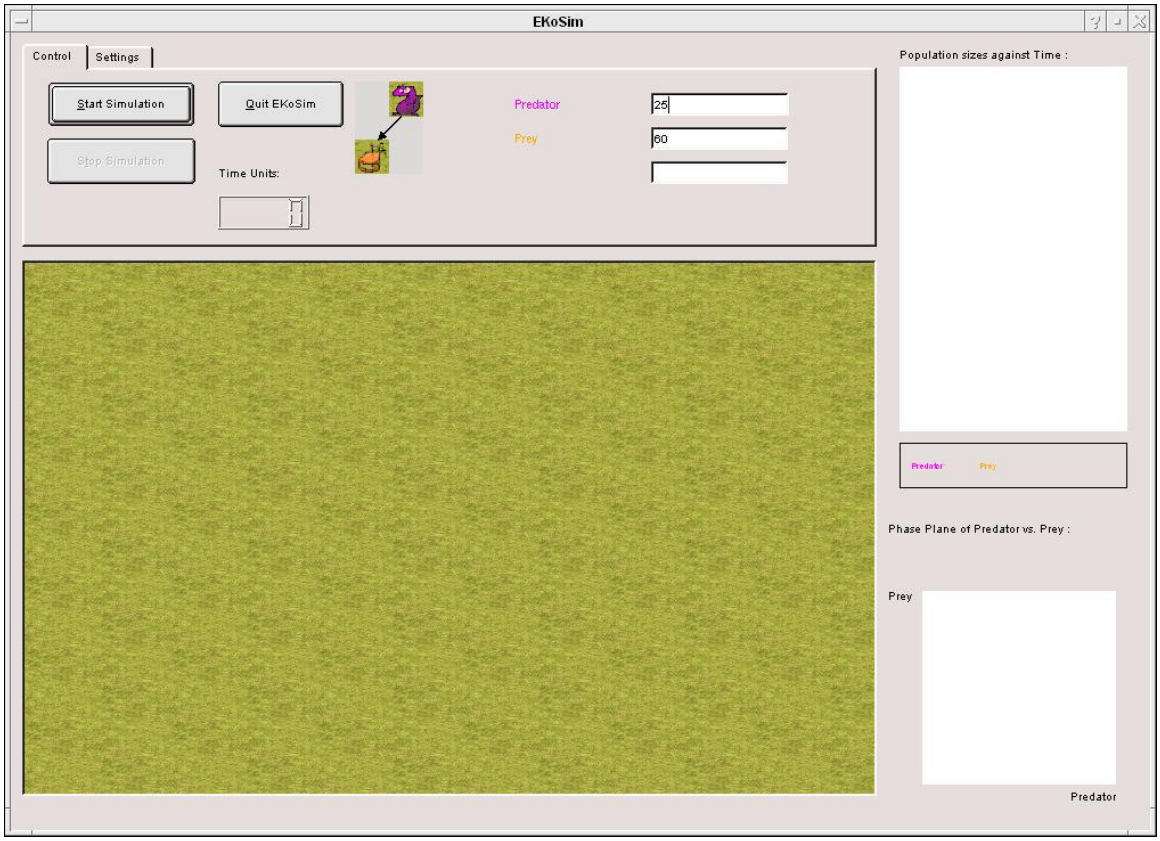
In order to install EKoSim Qt 3.1 should be installed on the system. In KDE that is not an issue, as KDE is Qt based.

2.2. Installation

Decompress the tar.gz file (making sure that the “pic” directory is intact) and run “make” (or “make all” or "make install") to compile and generate an executable
Run "make clean" to remove all intermediate files.
Run "make dist" to create a tar.gz file containing the code, the *.png files (picture files), Makefile, readme.txt and the license.
Once the executable “EKoSim” is generated you are ready to go.

3. Running

When EKoSim is run the following window will appear:



3.1. Controls

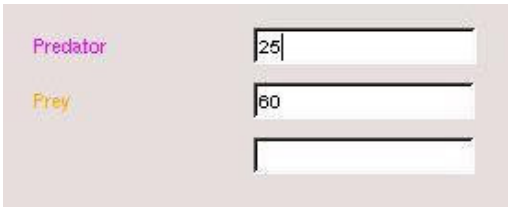
The control buttons: “Start Simulation” begins the chosen simulation
“Stop Simulation” stops the current running simulation
“Quit EKoSim” ends the program.
The “Time Units” display – counts the amount of time units passed so far in the simulation

The Picture to the right of the “Quit Button” button portrays the food chain scheme used in the current simulation chosen.

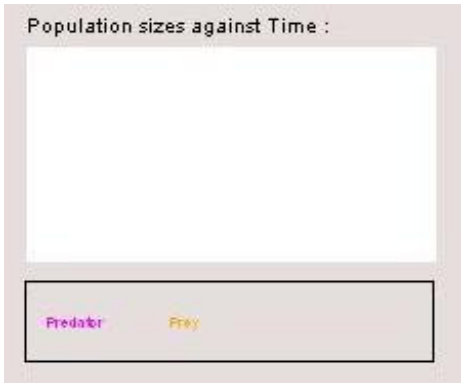
3.2. Display

EKoSim has four display fields which depict the simulation status to the user:

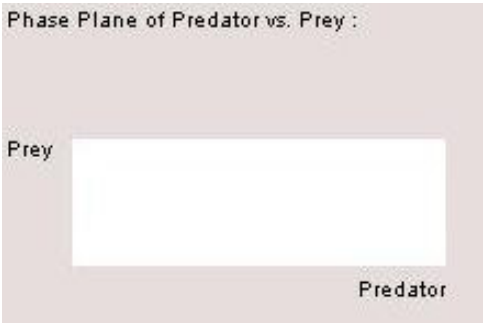
The numerical population size display: Displays the numerical value of the population size for the current clock cycle.



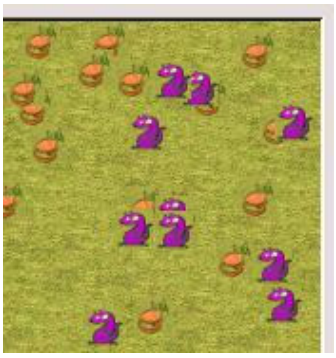
Population against time graph display: Displays a graph of change in population sizes against time.



Population phase plane graph display: Displays a graph of prey population against predator population.

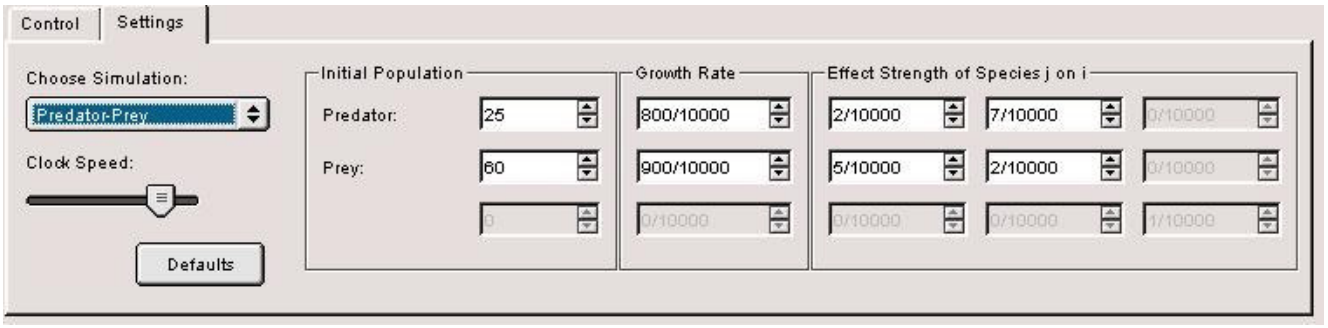


The final display is the above view of the organism environment: Displays the organism chasing each other in the environment and visually displays the change in population sizes. Where the purple “dragons” are the predator, the orange cat is the prey and the green frog (not shown here) is the second prey/lowest prey (depending on the simulation scenario).



3.3. Simulation preferences

The parameters of the simulation and the simulation scenario can be altered through the "Setting" tab:



Here using the "Choose Simulation" combo box the user can choose between three scenarios:

Predator – Prey: Two organisms exist in the simulation, one hunting the other.
Predator –Two-Prey: One predator hunting two different types of prey.
Three – Chain: A three organism linear food chain.
(When a 3 organism scenario is chosen the numerical boxes are activated)

The speed of the simulation may be adjusted using the "Clock Speed" slider.

In the "Initial Population" frame the user can set the initial population sizes for each organism.

The "Growth Rate" values are the intrinsic rate of change for the organism. For a predator this value describes the rate of decline in population size in the absence of food (prey). For the prey this value describes the rate of increase in population size in the absence of predators.

In the "Effect Strength of Species j on i" frame the values describe the effect of species on themselves and the others in the scenario. The diagonal values describe the effect of its own population size on its population decrease (effect of population density). The value of the second box in the first row (a_{12}), for example, describes the effect of the prey on the increase of the predators population (if it is larger then more predators are produced for every prey consumed). The value of the first box in the second row (a_{21}), on the other hand, describes the effect of the predator on the decline in the prey population.

Generally this can be portrayed as the following equations:

$$\frac{dN_i(t)}{dt} = N_i(t)\left(r_i + \sum_{i=1}^{n} \alpha_{ij} N_j(t)\right)$$

$$\Rightarrow N_i(t+1) = N_i(t) + \frac{dN_i(t)}{dt}$$

N_i is the population size of species i.
r_i is the intrinsic rate of change of species i (is positive for prey and negative for the predator).
a_{ij} is the interaction coefficient between species i and j (the effect of species j on the growth of species i. a_{ii} is negative.)
n is the number of species.

Here the growth rate values are r_i, the "Effect Strength of Species j on i" frame values are the matching a_{ij} and n is 3. From these equations the population sizes are calculated for each clock cycle. This ecological model is based on the generalized Lotka-Volterra model (multispecies quadratic model).

The "Defaults" button sets default values for the parameters. These values are also set when a different simulation scenario is chosen. The user is encouraged to run the simulation with these default values and then experiment with other values and their affect on the simulation output.

Project design and future development

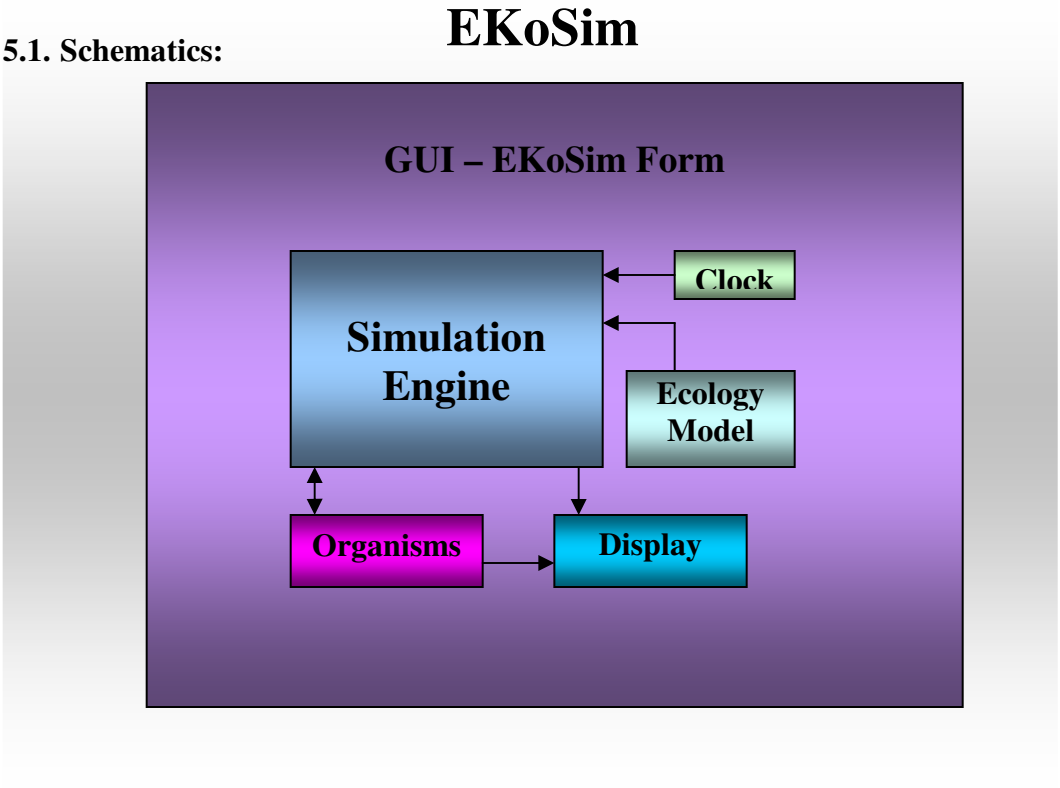
EKoSim is written in C++ using the Qt library 3.2 and the Qt designer

4. List of Files

EKoSim.pro	**- EKoSim Qt project file**
ekosimform.ui.h	**- EKoSim form method source file**
ekosimform.ui	**- EKoSim form**
organism.h	**- class organism header file**
organism.cpp	**- class organism source file**
model.h	**- class model header file**
model.cpp	**- class model source file**
main.cpp	**- main source file**
grass.png	**- canvas grass background pic**
dragon.png	**- dragon pic**
cat.png	**- cat pic**
frog.png	**- frog pic**
sim1.png	**- food chain 1 schema**
sim2.png	**- food chain 2 schema**
sim3.png	**- food chain 3 schema**

5. Software Engineering

5.1. Schematics:



5.2. Module Description

5.2.1. Simulation Engine

The heart of EKoSim is the simulation engine. The simulation engine is formed by the majority of the methods in the *ekosimForm* class. When the simulation is started an instance of the appropriate model is created, with the user defined parameters, and the initial amount of organisms is created and displayed on the "display canvas". With each clock cycle the engine, after receiving the new population sizes for this clock cycle, decides for each organism its new direction of movement, and then all organisms are moved on the display canvas.

5.2.2. Ecological Model

The ecological model is based on the generalized Lotka-Volterra model (multispecies quadratic model):

$$\frac{dN_i(t)}{dt} = N_i(t)\left(r_i + \sum_{i=1}^{n} \alpha_{ij} N_j(t)\right)$$

$$\Rightarrow N_i(t+1) = N_i(t) + \frac{dN_i(t)}{dt}$$

N_i is the population size of species i.
r_i is the intrinsic rate of change of species i (is positive for prey and negative for the predator).
a_{ij} is the interaction coefficient between species i and j (the effect of species j on the growth of species i. a_{ii} is negative.)
n is the number of species.

At each clock cycle the model derives the state of the system, based on the state from the previous cycle, and updates the Simulation Engine. Currently 3 model scenarios are supported:

Predator – Prey: Two organisms exist in the simulation, one hunting the other.
Predator –Two-Prey: One predator hunting two different types of prey.
Three – Chain: A three organism linear food chain.

5.2.3. Organism

Each animal in the simulation is created from the *Organism* class and is attached to the *display canvas* upon creation. With each clock cycle each organism calculates its distance from every other predator/prey in the environment, this minimal distance affects the choice of x/y velocity of the organism in the next clock cycle. Once each organism's new movement has been decided each organism checks for potential collisions on the canvas and the direction is altered if necessary**.**

6. Logic and application flow

6.1. Initialization and Simulation Start

Once EKoSim is started the *ekosimForm::init()* method loads the default simulation values, creates the clock, the display graphs and the display canvas. There is a matching *ekosimForm::destroy()* method for deleting memory allocated in the initialization.

When the "Start Simulation" button is pressed *ekosimForm::StartSimClock()* takes the user defined values from the gui and an instance of the *MathModel* class is created. The appropriate amount of each organism is created and attached to the *display canvas* at random locations and the clock is started.

6.2. One Clock Cycle

The *SimClock* is attached to the *ekosimForm::TickSimulation()* slot. According to the user defined interval (50-300 ms) this method is called every clock tick. The new population sizes are retrieved from the *MathModel*, the graphs are drawn and the numerical display updated. According to the delta between the population sizes from the previous tick and the new population sizes animals are added/removed from the simulation. The delta variables are defined as local static variables, as the delta is usually numerically small from one cycle to the next, so it is accumulated over the cycles until it is an integer then the appropriate amount of organisms is added/removed.

The next step is to calculate the movement of the organism's by calling *ekosimForm::CalculateMove* for each organism. This method is divided into three cases according to the three types of organisms. The organism then calculates (using *Organism::distancefrom* method) it's distance from every other organism of the matching type. When the organism with the minimal distance is found the appropriate x/y velocity is set. *Organism::distancefrom* takes into consideration the fact that the virtual world the simulation is run in is doughnut shaped. So if it finds that an organism's distance to another organism is closer when calculated "around the world", the distance is returned with a negative sign. The world is divided into 4 equal quarters, so when this happens, the 12 possible situations are taken into consideration, according to the location of both organisms in question (the first is in quarter 1 the other in 2/3/4 or the first is in quarter 2 the other in 1/3/4 etc. One may note that the distance between the organisms is shorter when calculated "around the world" only when they are in different quarters). When each organism's movement direction has been set the *QCanvas* method *DisplayCanvas->advance()* is called.
This method automatically calls the *advance()* method of each of its animated *QCanvasItems*. (*Organism* class inherits *QCanvasSprite* which is an animated *QCanvasItems*). The movement of the canvas items is performed in two stages managed by *QCanvas*. First the item's *advance* method is called with 0 as a parameter. In this stage the organism checks of its new (x,y) is outside the canvas, if

so it is moved "around the world". Now potential collisions are checked with other canvas items using a *QCanvasSprite* class method, if detected then the x/y velocity is adjusted. In the second stage each canvas item's *advance* method is called with 1 as a parameter here the actual movement is performed after updating the organism's quarter.

7. Future Development

Several issues are far from perfect and should be (with new additions as well) dealt with in future contribution/development of EKoSim:

7.1. Graphics Display

Originally OpenGL was considered as the library in which to create the above view of the environment. While working with the Qt library the QCanvas seemed the natural 2D solution and did not require the use of an additional library like OpenGL. Currently I am not completely satisfied with the graphics display and believe I am not fully utilizing the potential of QCanvas or it is not suitable for the graphics needed for EKoSim. The issue of overlapping pixmaps and collision do not produce the graphics display I desire (maybe using a grid could solve this…). Additionally when over 600 organisms are on the canvas the speed drops significantly (maybe not a graphics issue…). Another graphics issue is changing the pixmap of the grass background (I know the color isn't amazing) and the animal pixmaps.

7.2. Simulation Diversity

What will make EKoSim more attractive (besides its looks) is its diversity. The addition of other ecological scenarios, other theoretical models for comparison and more complicated mathematical models will make EKoSim more of an educational tool.

7.3. Code Quality

This is my first crack at C++, so I am sure it is not elegant and does not always take the right OO approach to things. The class design needs to be tweaked and there is redundant code that could be split into separate methods. The source code contains documentation before each method and inline comments where appropriate, together with this document I believe it is fairly easy to continue where I left off. Other than the methods described here the rest are short and very straight forward so the documentation in the code should suffice.

Appendix A: Original project definition

Note: During implementation and code writing several changes were made to the project and its aims. This appendix is added for comparison and does not hold as the project definition in its present state.

EKoSim is an ecological environment simulation game for the K Desktop Environment.

EKoSim can be used for educational purposes for demonstrating mathematical models of interaction between different life forms in a virtual environment and displaying the output through the graphical user interface.

In EKoSim the user can choose between several environments, a number of different life forms, apply different attributes to each life form (i.e. position on the food chain, mobility, reproduction attributes, life expectancy, and initial life form quantity) and adjust the virtual time "speed". According to the user defined preferences the appropriate model is chosen for the simulation. This flexibility allows for the creation of many different virtual environments so that each time the simulation is run the outcome is necessarily different.

The output of the simulation is displayed on the screen as an above view of the environment with the life forms moving on the ground/water (similar to simulation games – i.e. civilization) with additional graphs displaying the changes across time in the quantity of the individual life forms. All of the preferences can be defined through the gui.

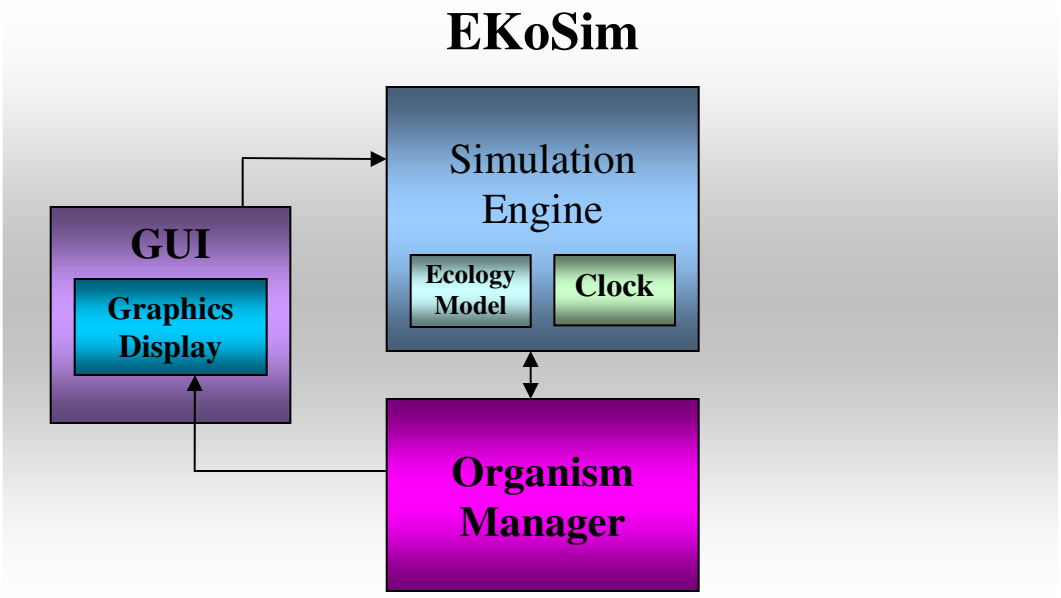
There are many biological simulation programs but these programs largely deal with the issue of evolution (the development of organism characteristics over time affected by environmental conditions or mutations) not ecology (the interaction of organisms in an environment based on the amount of resources and organism population). The few existing ecology simulations are for the Windows\Macintosh environment and are commercial software, there are no free\open source ecology simulations and none designed for KDE. In most cases these programs limit the user to the number of life forms and/or the mathematical model which is simulated and limit the user to predefined life form attributes. EKoSim is designed to offer the user the flexibility these programs deny.

Appendix B: Original project design

Note: During implementation and code writing several changes were made to the project and its aims. This appendix is added for comparison and does not hold as the project design in its present state.

❖ Software Engineering

➢ Schematics:



➢ Module Description

▪ Simulation Engine

EKoSim is a user parameterized graphical simulation of an ecological system, thus, the heart of EKoSim is the simulation engine. Using the parameters defined by the user through the GUI the engine chooses the appropriate simulation of the ecosystem and initializes the ecosystem.

- **Ecological Model**

The ecological model is based on the generalized Lotka-Volterra model (multispecies quadratic model):

$$\frac{dN_i(t)}{dt} = N_i(t)\left(r_i + \sum_{i=1}^{n} \alpha_{ij} N_j(t)\right)$$

$$\Rightarrow N_i(t+1) = N_i(t) + \frac{dN_i(t)}{dt}$$

N_i is the population size of species i.
r_i is the intrinsic rate of change of species i (is positive for prey and negative for predators).
a_{ij} is the interaction coefficient between species i and j (the effect of species j on the growth of species i. a_{ii} is negative.)
n is the number of species.

At each clock cycle the engine derives the state of the system, based on the state from the previous cycle, and updates the Organism Manager.

- **Organism Manager**

The Organism Manager is in practice the data base of the simulation. It is initialized by the simulation engine by receiving the organism attributes and quantities and computes the location of each organism in the ecosystem. At every clock cycle it receives the state of the ecosystem from the simulation engine and updates the organism physical location in the ecosystem according to its prior location and its specific attributes. The manager then updates the graphics display module.

- **Graphical User Interface**

The GUI is the receiver of the user preferences relevant to the simulation. After the user has defined the parameters these are passed on to the simulation engine and the "game" begins. Inside the GUI is the graphics display sub-module which provides an above view of the ecosystem and is updated by the organism manager.

❖ Resources

Currently I am working on this project alone but hope to receive assistance from other contributors through the project internet page.

➢ Priorities

- A usable graphical user interface
- A running Simulation Engine
- The Organism Manager
- Graphics Display
- Expanding the number and type of organisms
- Expanding the number and type of ecosystems
- Deriving from the above – Expanding the number of mathematical models the simulation can handle
- Improve the organism physical location decision mechanism
- A more detailed graphics display

➢ Problem Handling

One major problem regarding the simulation has arisen - Portraying a mathematical model onto a graphical display. The data derived from the mathematical ecological model is only the quantity of each organism as a function of time and not the location of the organisms inside the ecosystem. When displaying to the screen each organism has a location but how is this location chosen at each clock cycle?
As can be seen in most ecology simulations this issue is usually dealt with by randomly positioning each organism on the display with every clock cycle. This creates a situation where "the rabbit chases the fox" on the screen and so the display has no actual value in portraying the events inside the ecosystem.

EKoSim tries to deals with this using the Organism Manger. The organism manager at each clock tick, after receiving the organism quantities for this clock cycle, relocates each organism according to its location in the previous clock tick and the organism's attributes (mainly location on the food chain). Introducing new organisms (by animal breeding) into the environment is done by physically locating the new organisms near other organisms of the same type. Reducing population (because of predators) is done from populations that are physically close to a predator. This causes the organism's location on the screen to have real consequences on the physical arrangement of the organisms in the future clock cycles making the simulation display much more accurate and closer to actual life.

❖ Development Infrastructure

The project will be written in the C++ language (as an object oriented language is the natural choice for such a project) and will be a good chance for me to expand my knowledge in C to C++. The graphical user interface will designed using the QT library, which is the natural one used with the K Desktop Environment. The graphical display will be done with OpenGL.
The development tool will be KDevelop, again as a natural choice in KDE and as it has a good integration with QTDesigner.