Darwin's Dream

Andreas Huber
Institute of Distributed and Multimedia Systems, Univ. of Vienna, Lenaug. 2/8, 1080 Vienna, Austria
e0340147@student.tuwien.ac.at

Markus Paulhart
Institute of Distributed and Multimedia Systems, Univ. of Vienna, Lenaug. 2/8, 1080 Vienna, Austria
e0325357@student.tuwien.ac.at

Christian Kloiber
Institute of Distributed and Multimedia Systems, Univ. of Vienna, Lenaug. 2/8, 1080 Vienna, Austria
christian.kloiber@gmail.com

Helmut Hlavacs
Institute of Distributed and Multimedia Systems, Univ. of Vienna, Lenaug. 2/8, 1080 Vienna, Austria
helmut.hlavacs@univie.ac.at

ABSTRACT
This paper introduces Darwin's Dream, a system for simulating artificial plant life on a planet wide scale. Darwin's Dream contains a sophisticated climate simulation, which produces different climate zones on a planet, taking into account things like rain fall, clouds, and sun light intensity. The climate simulation is used to drive the evolution of a complex plant ecosystem. As planet topology we use something like a "flat sphere", which eases the use of 2D height maps. An important focus of Darwin's Dream is also on the realistic visualization of the planet's surface.

Categories and Subject Descriptors
I.6 Simulation and Modeling: Miscellaneous

Keywords
Climate simulation, artificial life, evolution of plant live, weather visualization

1. INTRODUCTION
The availability of high computing power in off-the-shelf PCs has enabled the simulation of live evolution for everyone. Evolution of life is an extremely complex area, since evolution as a stochastic process requires the investigation of the genome of large populations, i.e., hundreds, better thousands of subjects should be observed for many generations.

Evolution here means the process of inheriting and mutating genomes which decide about how the individuals fit into their environment and whether species survive. If resources are scarce or the environmental conditions are unfavorable, then only the fittest will survive, as postulated by Charles Darwin. The simulated genome must represent basic properties of the simulated species which are necessary for survival under different environmental conditions. Simulation of evolution can be viewed from various aspects.

First, it is a model explaining why evolution itself works. Second, it might be a way for optimizing complex systems, as done by genetic or evolutionary algorithms, or genetic programming. Third, it might try to forecast live of our planet in a few million years, or it might try to model how live may evolve on alien planets with conditions totally different to ours. Apart from serious applications, evolution of live might be simulated for the sake of fun, in a game like fashion. The result of the evolution of artificial live, as unpredictable as it is, might have fascinated Charles Darwin (thus the name of the project).

2. RELATED WORK
The simulation of live can focus on the evolution[1] of animals, or the evolution of plants. For the evolution of animal life or more abstract "life" forms like computer programs, numerous systems exist[2], for example [2, 4]. Evolution of pure plant life is seen not as often as animal evolution, probably due to its more static nature. Panspermia [3] was a system very similar to ours, focusing on the evolution of plants with their various shapes, and using professional graphics workstations. Interactive Plant Growing [5] was more an art installation than the simulation of plant life in an ecosystem. Nerve Garden [3] simulates the evolution of plant life, but may also include insects and is based on VRML 2.0, but on a smaller scale (islands) than our planet wide system. Nerve Garden itself evolved into the Biota@Home initiative, which focuses on the creation of artificial nature systems run on peer-to-peer networks.[3] The company Maxis created several games focusing on the evolution of life, including SimEarth and SimLife [6]. Together with Electronic Arts, Maxis recently released the game Spore, which is the best professional artificial life game so far.[4]

The innovation of our system called Darwin's Dream is the complex interaction between a detailed climate simulation and the novel flat sphere topology on the one hand, and a planet wide plant ecosystem on the other hand. However, Darwin's Dream is rather meant for pleasure in a game like fashion than for studying plant evolution itself.

[1] http://www.google.com/Top/Science/Biology/Evolution/

[2] http://www.google.com/Top/Computers/Artificial_Life/

[3] http://www.biota.org/

[4] http://www.spore.com/

3. DARWIN'S DREAM

The aim of the interdisciplinary project Darwin's Dream is to simulate and visualize a planet's biosphere, including its climate and flora. Plant life of the planet is ruled by the laws of evolution, while the climate simulation adheres to knowledge from climatology and meteorology.

The main idea is to create a world similar to ours, with similar plant evolution and weather. It must be noted that though the used models try to mimic as much of real evolution and climate as possible, the project does not claim to implement models of the same complexity as found in the respective specialized sciences. Instead, it is an attempt to create a sandbox for playing with a world, similarly to games like "SimCity" or "Spore".

Since the computation of the behaviour of thousands of plants demands high computational power, a main goal has been to find a suitable software architecture for reducing the main task into smaller manageable subtasks. As a consequence, the system is split into a server part being responsible for computing the main simulation, while the visualization of the results is done on the client side. For doing this, the server transfers a snapshot of the world to the client, which then may roam freely through the world and its flora and current weather.

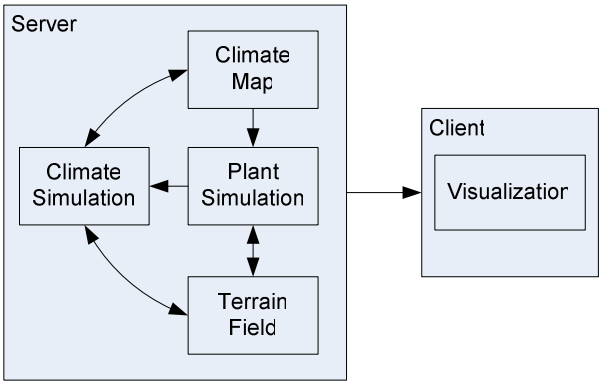

Figure 1. The system architecture.

The whole system has been implemented by using C++, for reasons of its object oriented nature and a probably better performance compared to languages like Java. The server side implementation has been implemented in standard C++ without any fancy libraries, thus enabling to run the simulation on different platforms.

4. CLIMATE SIMULATION

The server side simulation is responsible for simulating the climate of the world. Here, one month is split into 30 days and nights. The main difference between the months is the different inclination of the plant's axis, thus mimicking the different intensity of the sun light warming up the planets surface throughout a year.

The climate simulation is based on three main objects.

- The planet's surface. Here all plants are grown. The surface is realized using a 2D grid, each element of the grid represents an area equivalent to one square kilometer. These elements are called *terrain fields* and are able to retain water, or give off water to its surface.
- The planet's atmosphere. This is a medium above the planet's surface, which is able to store and transport air and water. It is modelled using a 3D grid, each grid element being called *climate block.*
- The climate map. This is the main simulation result of the world climate, storing average climate data of days and nights for each month and climate block. The climate map is then used as an input into the plant simulation with constant values for each simulated month. For instance, since in the month May rain is observed only rarely, the average rainfall of May is quite low. This value then is used for defining the average rainfall for each day in May. Since the climate of a planet is known to be very stable, the climate map has to be updated only infrequently, every few simulated years.

The planet's surface is also annotated by height information. This information is taken from a grey-level graphics file, each pixel representing the height information of the terrain field it represents. As already pointed out, a terrain field's area is 1 km^2, the height map contains exactly $n \times 2n$ terrain fields as shown in Figure 2, where the integer n depends on the graphics file's resolution and can be chosen arbitrarily.

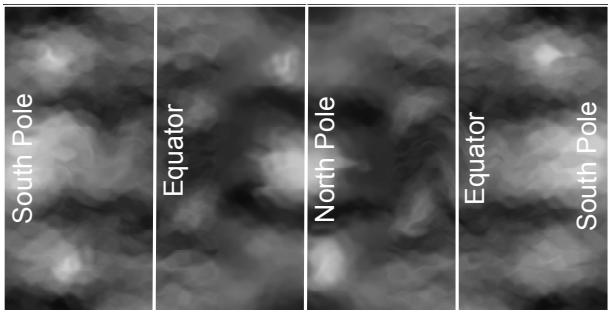

Figure 2. The planet's height map (transposed).

Figure 2 also shows the topology of the world map, and explains why the chosen size is $n \times 2n$ instead of $n \times n$. The height of the world map is twice the size of the width, the north pole being assumed to be in the *middle* of the map, rather than at the top. The south pole is split into two parts, one at the top, the other at the bottom. The equator also is represented two times, being between the north pole and the two south poles. The idea of this topology is as follows. The south pole at the top is bent down and is connected to the south pole at the bottom, thus creating a cylinder in 3 dimensions with only one north and one south pole. Of course, a planet always has the shape of a sphere, and not a cylinder. Since a sphere contains only one equator, the cylinder is then squeezed again into a flat form, thus connecting the left and right sides to their respective counterparts on the other side. The result is something like a flat sphere (Figure 3).

The reason for this peculiar topology is this. The resulting flat sphere contains distortions at the poles which are much smaller than the distortions found in other visualization mechanisms. Furthermore, at the equator no distortions are found. Assuming that the world map from Figure 2 is described by a coordinate system $(x, y), 1 \le x \le n, 1 \le y \le 2n$, then if someone for instance is at the position $(1, y)$ and goes one terrain field to the left, he will

enter the terrain field $(1,2n-y)$, this time *coming* from the left. On the other hand, someone standing at position (n,y) and going one field to the right will enter terrain field $(n,2n-y)$ from the right. This additionally makes sure that people travelling east or west always remain at the same latitude, and thus in similar climatic regions.

The result of the above described mapping is a way for representing sphere textures by a 2D grid. Throughout a simulation run the grid fields are updated row and column wise by using the climate map. During such an update it is tested whether a terrain field is able to emit water, which cannot be stored in the terrain field, to a neighboring field. This way it is possible to dynamically create and destroy stagnant and flowing water bodies.

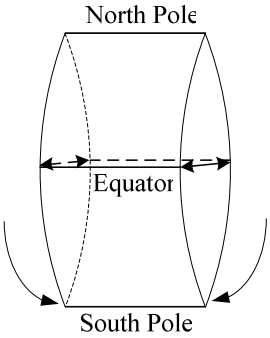

Figure 3. The planet's 2D/3D topology.

5. GENETIC MODEL

Darwin's Dream implements a mature genetic model of plant life and evolution. Plants develop according to their genome, which is a set of genes and which describe the plants physiology. Darwin's Dream implements a very complex genome which allows to describe not only the plant's appearance, but also its interaction with the environment, and additionally its interaction with other plants. As an example, one gene describes how sensitive the plant reacts to different light conditions, i.e., areas with much sun light, or with less sun light. As a design principle we tried to minimize the set of genes which have only either positive or negative properties, i.e., where the optimum profit (or maximal damage) for the plant is achieved either at the minimum or maximum value. Also the genome contains detailed properties of the plant's appearance. In total, the genome of each plant consists of 697 bits describing 58 genes.

5.1 Plant Status

An important part of the genome decides whether the plant can survive under certain environmental conditions. For instance, a plant needs water, nutrients, sun light and certain temperature conditions for its survival. Since these factors are usually not all satisfied optimally, the growth of plants depends on the environment, and the plants will show different properties depending on their position. Thus, additionally to its genome, each plant is equipped with individual status variables describing its current condition, for instance its current size.

In order to create a complex system of woven properties, the status variables also depend on numerous genes, but also on the current value of other status variables. In total, each plant is equipped with 36 status variables.

5.2 Plant Life Cycle

The life start of each plant is triggered by a seed which might be either created artificially by the user, for instance at simulation start up, or it might be created by a mother plant which passes on its genome. The seeds are first carried around by the wind, and later dropped down to the floor after some time. At some instances, the wind phase might be quite short, and the seed might drop down almost immediately. Once lying on solid ground, the seed starts to grow its roots, but only if the environmental conditions are within some bounds. If the conditions never fulfill the necessary preconditions, the seed will die after some individual random time has gone by.

Once a plant spreads out of its seed, the following stages are gone through:

First, the minimum amount of water needed by the plant is computed. If this minimum is not available in the plant's surrounding, then the plant suffers damage. The amount of damage depends on the difference of available to minimum required amount of water. Generally, any insufficiency of nutrition is recorded in the plant's status variables.

Only if the minimum amount of water is available, the plant starts growing its stem. Again the amount of growth depends on environmental factors, like enough sun light, water and nutrition.

The third stage is the growth of leaves and blossoms, again depending on environmental conditions. Since it would be very demanding to visualize the slow growth of leaves and blossoms, we decided to simplify the visualization at this point, meaning that leave and blossom growth happens instantaneously without time delay.

The final stage of a plant is its reproduction, which is carried out periodically throughout the life of a plant. The amount of reproduction again depends on the environment, the better it is, the more seeds are produced and sent into the world. There are two ways for reproduction. First, if a plant was not hit by similar spores from other plants, it may decide to reproduce autonomously and inseminates itself. If, on the other hand, it has been hit by spores from similar other plants, the produced seeds will be a combination of both genomes. This is done by randomly selecting genes from the two genomes and creating a new genome from them.

6. VISUALIZATION

As was explained earlier, a snapshot of the world simulation is transferred to the client to be visualized. The client enables to roam this world freely by flying in 3 dimensions above the surface. The client of course also allows to retrieve data about the climate, plants and plant evolution.

An important point for choosing a suitable rendering system was given by the fact that the visualization of a whole world and possibly thousands or even millions of plants is very resource demanding, thus requiring powerful PC technology and mature graphics cards. Furthermore it was clear that an important aspect of the visualization would also include the rendering of weather and climate.

As target platform we chose Windows XP with compiled C++ programs, a Java based approach was not chosen because of the already mentioned performance considerations.

For graphics rendering, the Object Oriented Graphics Rendering Engine (OGRE) has been chosen.[5] Amongst others, the reasons for choosing OGRE include its strict object orientation and C++ binding, its maturity, the good documentation and its free availability.

6.1 Terrain

OGRE already contains support for the visualization of terrains and surfaces (Figure 4). A possible obstacle here is given by the fact that OGRE only allows to use $n \times n$ maps for height information. In order to use the same maps for visualization and climate simulation, the $n \times n$ height map used for visualization has been stretched in its y-dimension by a factor of 2, thus yielding the corresponding height map for climate simulation. For improving the realism, further textures were put on top of the standard ground texture. For instance, in case of considerable vegetation, the surface is covered by green meadows, while the arctic regions are covered by white ice. Depending on the simulation outcome, several different textures can be combined.

Figure 4. Terrain with trees.

6.2 Day and Night

Since the visualization depicts only one particular snapshot, time is frozen during the visualization. This means that the areas with daylight and nightfall do not change when roaming through the planet. Especially the sun does not change its position. For reaching this effect the picture of the sun and the sun rays are put at a fixed distance from the observer. As soon as the observer moves, the position of the sun is moved and rotated around the observer (Figure 5).

Depending on the position of the sun, the direction, color and intensity of its light rays are computed. Additionally, the color of the sky also depends on the position of the sun and the observer. If the observer enters a region with night, the sky's color changes to dark blue or black, while the terrain fog gets more dense and darker. Furthermore, a starry sky is emulated, its transparency and thus visibility depends on the amount of darkness of the area. Stars are represented by pictures of stars, being similar to a sun, but without their own lightening source, and being placed at a constant distance above the observer. As a consequence, we do not use a sky box as done in many other computer games, although OGRE of course enables the use of such a tool.

Figure 5. A cloudy and sunny sky.

6.3 Water

As was said before, the simulation enables the dynamic creation and deletion of stagnant and flowing water. These are visualized by using squares, which are equipped with a texture showing a water animation. The position of these squares then depends on the water level above the average height of the respective terrain field. Since terrain fields have different heights, this way, small islands may be created. The positioning is done after the water levels of neighboring terrain fields have been counterbalanced with each other, resulting in a homogeneous water level. An additional animation of oscillating waves is then put over the common water surface (Figure 6).

The observer is also able to dive into the water, thus resulting in a darkening of the scenery, the light source is hidden and the used fog is made denser.

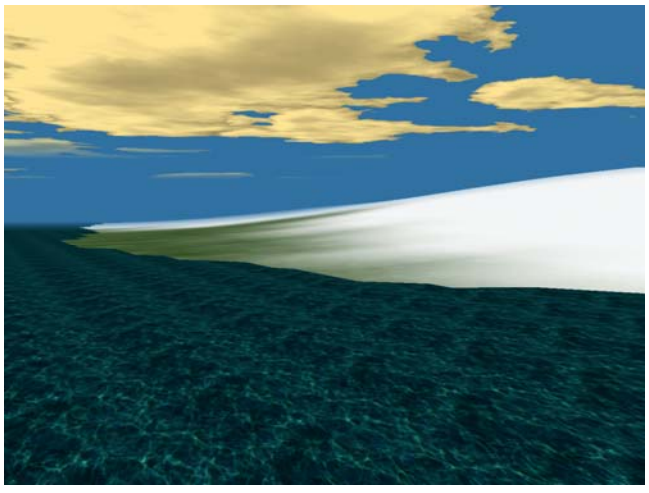

Figure 6. Arctic sea.

[5] http://www.ogre3d.org/

6.4 Weather

For the visualization of the weather, a major challenge was the dynamic creation of clouds. Clouds are made of a number of different cloud elements, which are chosen depending on the current weather conditions created by the simulation engine. A cloud is positioned above a terrain field and may overlap neighboring terrain fields. The higher the cloud density should be rendered due to simulation, the more cloud elements are used, and the darker the center of the cloud gets. If a cloud exceeds a certain size, it is equipped by a cloud bottom including a suitable cloud surface texture. The bottom is surrounded by small smooth silky cloud elements which always point into the direction of the observer. On top of a cloud, a number of large floating cloud elements form a kind of stack above each other. These stack elements are again equipped with realistic cloud textures. Smaller clouds do not possess a bottom, and their cloud stacks are created by using smaller stack elements. As a result clouds are complex dynamically created objects, neighboring clouds even can be combined into one larger cloud. This way almost all cloud types observed in real nature can be simulated (Figure 7).

The weather simulation also allows the visualization of rain and snow fall. Since the OGRE engine uses particle systems for both, heavy rain or snow fall may induce severe performance problems to the client.

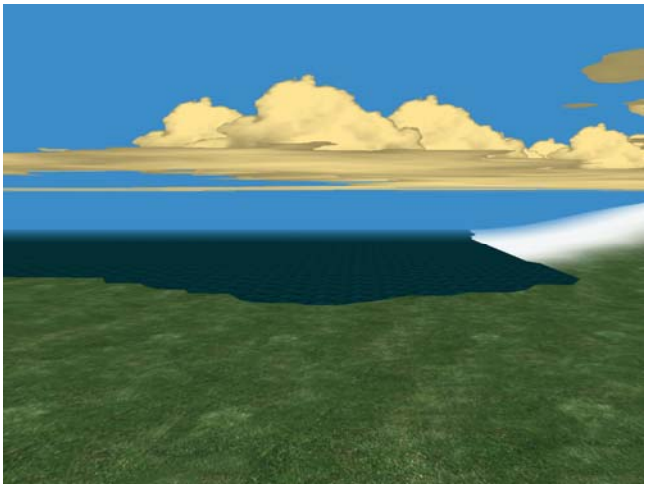

Figure 7. Clouds.

6.5 Plants

Similar to clouds, plants are also created by combining several elements into one plant object. According to the respective plant genome, the plant simulation delivers data for the plant stem and its treetop. However, the terms “stem” and “treetop” here represent the respective parts of all plant types, not only trees. For instance, in the simulation, herbages and bushes do not contain a stem but a “treetop” with suitable texture. Trees then are additionally equipped with a properly scaled stem. For delivering a suitable representation of different plant types, the simulation engine chooses the suitable textures from a large palette of plant parts. Since Darwin’s Dream has been designed for simulating large numbers of plants, the respective parts of a plant are created by placing the texture onto two orthogonal pictures. This way, a strong off-the-shelf PC, perhaps additionally equipped with a consumer graphics card, is able to visualize thousands of plants without showing loss of performance (Figure 8).

Figure 8. Plants at evening.

6.6 The User Interface

The graphical user interface allows to retrieve information about all presented plants. This is done by mouse clicking onto the respective plant. This information includes the plant age, the plant height, its fertility cycle, the genome, general plant status, and many other things.

The GUI also allows loading different simulation snapshots from the simulation server.

7. COMMUNICATION

For reasons of human readability the communication between server and clients is carried out via an XML file. Fortunately, many free libraries for the manipulation of XML structures, like Xerces-c[6] or TinyXML[7] exist.

A drawback of XML is the fact that documents containing thousands of objects might get very large. In order to limit this growth, we have decided to use abbreviations instead of full names whereever possible. The abbreviations are explicitly specified at the start of the document. An example for this is given by

```
<ulPlantInitDate value="A" />
…
<A value="0" />
```

Here the attribute “ulPlantInitDate” is replaced by the abbreviation “A“. Later in the file, for a specific plant, the attribute “A” is then assigned a certain value, in this case zero.

During simulation, the server periodically produces such a system snapshot, which may then be transported to the client, for instance by using HTTP. The file may be used for visualization at the client, but also as a starting point for additional simulations.

---

[6] http://xml.apache.org/xerces-c/

[7] http://www.grinninglizard.com/tinyxml/

8. CONCLUSION

In this paper, the evolution game Darwin's Dream is presented. Darwin's Dream is meant for creating a world of plants, and simulates artificial plant life and its evolution. The main focus of Darwin's Dream lies on an accurate simulation of a planet's climate, the evolution of plants growing in different climatic zones, and a realistic visualization of the simulation result.

The system now is in a very early beta state, requiring still large effort for reaching a stable version. However, all important parts, including climate simulation, plant evolution, and visualization already work and can be tested.

9. REFERENCES

[1] B. Damer, Nerve Garden, ACM SIGGRAPH 97, 1997.

[2] T.S. Ray, Overview of Tierra at ATR. *In Technical Information, No.15*, Technologies for Software Evolutionary Systems. ATR-HIP. Kyoto, Japan, 2001.

[3] K. Sims, Artificial Evolution for Computer Graphics, *Computer Graphics 25-4* (1991), pp. 319-328. http://web.genarts.com/karl/panspermia.html

[4] K. Sims, Evolving 3D Morphology and Behavior by Competition., *Artificial Life 1-4* (1994), 353-372.

[5] C. Sommerer and L. Mignonneau, Interactive Plant Growing, in *Ars Electronica - Facing the Future* (Cambridge, MA: MIT Press, 1999), pp.393-394.

[6] Wikipedia, SimLife, http://en.wikipedia.org/wiki/Simlife