

Modeling of ecosystems as a data source for real-time terrain rendering

Johan Hammes

PO Box 1354
Stellenbosch
7599
South Africa

jhammes@mweb.co.za

Fax: +27 21 880 1936

Telephone: +27 21 880 1880

Abstract. With the advances in rendering hardware, it is possible to render very complex scenes in real-time. In general, computers do not have enough memory to store all the necessary information for sufficiently large areas. This paper discusses a way in which well-known techniques for modeling ecosystems can be applied to generate the placement of plants on a terrain automatically at run-time. Care was taken to pick algorithms that would be sufficiently fast to allow real-time computation, but also varied enough to allow for natural looking placement of plants and ecosystems while remaining deterministic. The techniques are discussed within a specific rendering framework, but can easily be adapted to other rendering engines.

Keywords: ecosystem modeling, ecotope modeling, compression, real-time rendering, terrain modeling, terrain visualization

1 Introduction

With the advances in rendering hardware, it is possible to render very complex scenes in real-time. Scenes of up to 200 000 triangles are becoming possible with much more complex scenes looming in the near future. The ability to render very complex scenes necessitates databases that can supply the rendering engine with enough data.

Landscapes for flight simulators, games or visualization, poses a problem due to the large amounts of plants found on them. It is not uncommon for a flight simulator to have 500 million big trees in its database area, and countless smaller trees and shrubs. Due to the scale, and the amount of memory needed to hold the information, these trees have in general been left out of the simulation, or incorporated into the textures. In this paper, a system of ecotope¹ modeling, that calculates the plants in the view frustum in real-time, is presented.

The exact placements of these plants are seldom known, and seldom of real importance. For vast areas of countryside they add to the general feeling of realism. This lends itself to the modeling of ecosystems to determine the placement of plants. By modeling the ecosystems, and generating only the plants in the immediate vicinity, the overhead of storing millions of plants in memory can be avoided.

Ecotopes are a good predictor of ecosystems. Rather than modeling the ecosystems (modeling the complex interactions between plants), ecotopes are modeled and the landscape is populated with representative ecosystems.



Fig. 1. Two images created by Sam Bowling, using the World Construction Set [1] by 3D Nature. It shows the realistic natural scenery that can be generated with ecotope modeling

By using the basic parameters of elevation, relative elevation, slope, slope direction and multi fractal noise, it is possible to generate ecotope information that can be used to predict the ecosystems. An ecosystem is assigned a probability that it will exist as a function of the above five parameters. By evaluating them at a particular position in the world it is possible to find a probability for each ecosystem. The ecosystem with the highest probability is assigned to that particular area.

The appearance of each ecosystem is defined in advance. This includes descriptions of the type, frequency and placement of plants, as well as typical ground cover. This information is used to build a representation of the ecosystem. The information is passed on to the rendering engine where it is cached, and reused for all subsequent frames that look at the same area.

¹ Ecotope : A particular habitat within a region with relative uniform climatological and soil conditions. Typically, specific ecotopes will be associated with specific ecosystems.

In section 3, the framework that is used to render the scenery is discussed. The definition of ecosystems and modeling of ecotopes is structured around this. Section 4 looks at common parameters that is used to model ecotopes and ecosystems. In section 5 this is extended to show how ecosystems can be constructed to fit into the rendering framework. In section 6 the algorithms that is used to calculate the ecotopes and build a representative ecosystem is presented. I also look at some optimizations that are necessary for real-time performance. Section 7 looks at the results and is followed by a conclusion in section 8.

2 Background

Most of my previous work dealt with large-scale terrain visualization for both commercial, gaming and military flight simulators. As rendering speeds increased it became possible to add more and more objects into the terrain adding to the realism. At first it was adequate to develop off-line tools to place objects and store them in the database. With the current commercial rendering capacity far exceeding one million triangles per second, it is impossible to store all the objects in a limited amount of RAM. A way had to be found to efficiently compress this data and extract only the data in the immediate vicinity of the camera in real-time.

When viewing natural features it is the general patterns that define the area, rather than the exact detail. The specific position of a plant does not define a natural scene, but the relative placement due to competition for natural resources is very important. Ecosystem and ecotope modeling constitutes a form of compression for natural environments. While the exact position and type of plants are not preserved, the general statistical properties remain, allowing a representation to be built with the same feeling and character.

A number of commercial programs that model ecosystems (ecotopes) to synthesize data exist. World Construction Set (WCS) by 3D Nature [1], Terragen by Planetside Software [2], and Genesis II by Geomantics [3] to name a few. WCS is one of the best examples of ecotope modeling as seen in Figure 1. The realism in the images is a convincing argument that ecotope modeling is a good way of generating natural scenery.

While these programs can render very realistic scenery, they are not real-time, usually taking minutes to render a picture. This paper looks at the possibilities to adapt ecotope modeling to the time constraints of real-time rendering.

3 Rendering Framework

The rendering framework divides the area into a set of square tiles, that can recursively break into smaller tiles in a quad-tree structure. Tiles split at a fixed distance from the camera, relative to their actual size and the field of view of the camera. This result in tiles that are roughly the same size in view space. Figure 2 shows the way that the surface splits for a particular camera position close to the ground. This splitting can be stopped at any level, and is combined with a prediction algorithm that looks at the movement of the camera to predict tiles that will be visible in the near future. There are a lot of tiles present that do not fall within the bounds of the view frustum. This is due to the cache prediction algorithm. A fast rotation of the camera is the most difficult situation to handle, and requires a lot of tiles adjacent to tiles in the view frustum to be present.

Only the tiles included in Figure 2 need to be extracted from the database, and reside uncompressed in memory. The algorithm makes use of frame coherence to update only those tiles that change between frames.

Each tile in this representation consists of:

- A 17x17 grid of elevations covering the extents of the tile. As the tiles split and become smaller, the actual resolution of the grid increases. This is the first step (block based) in triangle optimization.

- A 128x128 pixel texture depicting the ground cover of the area. The size was chosen to ensure roughly a 1:1 pixel to texel ratio for screen resolutions between 800x600 and 1024x768. At higher resolutions it will be necessary to either use a bigger texture or change the tile split metric to maintain this ratio.
- A list of objects that reside on this tile (trees, rocks, houses etc). This is only a key showing the type of object and does not include any information on the geometry of the object or how to render it.

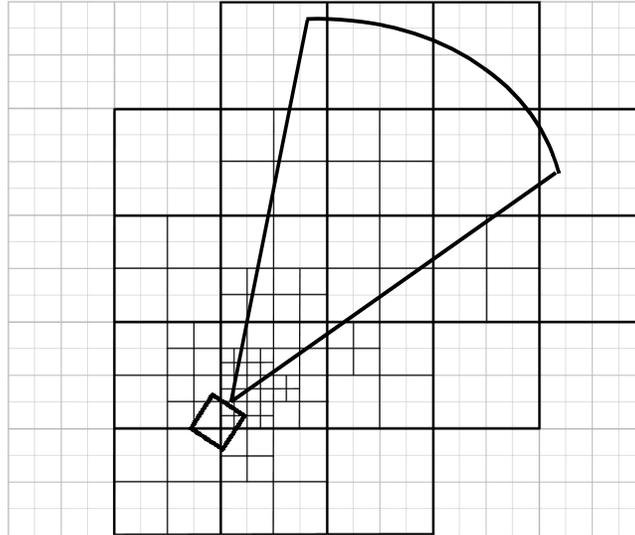


Fig. 2. The subdivision of tiles in the view frustum for a particular camera position

Because the parent tiles always exist, it is possible to render a representation of the terrain even if all of the tiles are not cached. While all four daughter tiles have to exist before any one of them can be drawn, it is possible to draw the parent tile for a few more frames while caching the daughter tiles. This enables the system to update only a few tiles every frame, while maintaining a consistent view of the terrain. It is very important since it allows a fixed time to be allocated to extract a tile from the database. Although drawing a parent tile affects the visual quality (lower resolution) it is possible to handle 400 degree per second turns with only three tiles being updated every frame. This is fast enough for the most demanding games or simulators. Due to the nature of the system a tile will never be cached unless its direct parent tile is already cached. This allows the use of information from the parent when caching a tile.

4 Modeling Ecotopes

This section looks at common variables that are used to model ecotopes in existing software and discusses their influence on ecosystems, and the way in which these parameters are combined to predict ecosystems.

Due to the constraints of real-time modeling it is important to use a fairly simplistic system that can be optimized. Table 1 shows a list of variables being used by WCS [1], with both Terragen [2] and Genesis [3] using a subset.

Table 1. Summary of the variables that is used to calculate ecosystem placement

Elevation	The height above sea level. With increases in elevation, the general conditions become harsher. All plants have an upper limit at which they can survive. Plants also tend to become smaller with increases in altitude.
Relative elevation	Relative elevation refers to the local changes in height, with negative values showing depressions, valleys etc, and positive values showing ridges. This is the higher frequencies of the terrain. Relative altitude affects plant growth since

	valleys are generally wetter, as well as more sheltered. Ridges on the other hand tend to be exposed to the elements much more.
Slope	The slope of the terrain has a direct bearing on the quality and depth of the soil, as well as water retention due to runoff. Steep slopes tend to have small shrubs and grass cover. Very steep slopes tend to be exposed rock with no vegetation.
Slope direction	The direction that the slope faces has a direct bearing on how many sunlight hours it receives each day, as well as being more sheltered or exposed to the prevailing winds.
Multi-fractal noise	Some plants and ecosystems also exhibit local grouping behavior independent of the above 4 variables. One reason is reproductive behavior. Plants that either drop their seeds, or reproduce vegetatively from roots tend to exhibit strong grouping behavior. A lot of multi-fractal noise functions exhibit similar patterns, and can be used to change the probability of ecosystems, or the density distribution of plants within ecosystems, to model this behavior.

Each ecosystem in the database is assigned a probability for each of the above variables. By combining the probabilities, a probability for each ecosystem can be determined at a specific position on the terrain. The ecosystem with the highest priority gets assigned to that position.

5 Defining an Ecosystem

Unlike most other systems (WCS [1], Terragen [2] etc.) that do a single ecosystem calculation for each position on the terrain, ecosystems are divided into layers to fit into the quad-tree rendering structure, and each layer is solved recursively as the tiles split. This section looks at the way that ecosystems split into layers, the sort of information available on each layer and the ways in which they can be combined to form complex landscapes.

Table 2. Summary of possible layers present in a savanna ecosystem. For each layer in the ecosystem an approximate size of the tile is given as well as the approximate size of plants present in that layer

Layer	Tile size	Average plant size when first used.	Vegetation canopy	Ground cover
0	8 km	64 m	None	Typical aerial photographs of the terrain with trees shrubs and grass. See Figure 3
1	4 km	32 m	Big trees	Smaller trees, shrubs and grass
2	2 km	16 m	Medium trees	Small trees, shrubs and grass
3	1 km	8 m	Small trees	Shrubs and grass
4	512 m	4 m	Big shrubs	Small shrubs and grass
5	256 m	2 m	Shrubs	Grass
6	128 m	1 m	Small shrubs and tall grass	Grass with patches of brown ground
7	64 m	50 cm	Short grass	Small plants, rock and ground
8	32 m	25 cm	Small plants and rocks (any objects with natural distribution can be defined in ecosystems, not just plants)	Soil surface with fallen leaves, and small rocks. This would be similar to the second row of textures in Figure 3

A layer consists of a vegetation canopy and ground cover and is defined for a specific layer in the quad-tree terrain. Since a tile will always fall within a pre-determined size range in screen space, it is possible to estimate the pixel size of all plants on the tile. The vegetation canopy is defined as all the objects (plants, rocks etc.) that are roughly one pixel in size when this layer is first used. Table 2 shows a typical

representation for savanna. There is no need to solve for plants that will be smaller than one pixel on screen, since they will make no contribution to the visual quality of the scene, and can be incorporated into the ground cover. Plants that are bigger are incorporated into a higher level and will already be solved at this stage. The ground cover is a texture with a representative image of the ground as well as all plants that are still smaller than one pixel in this layer, and can be seen in Figure 3.

5.1 An Ecosystem layer

Each layer in the ecosystem consists of the following items:

- A texture representing typical ground covers. This is used to build the texture that is draped over the terrain when rendering. A number of examples are shown in Figure 3.
- A list of possible objects in this ecosystem. This includes :
 - The type of object. (It is possible to define objects like rocks as well, as long as they have a natural distribution)
 - Object density. This is used in conjunction with a random offset to determine the number of objects in a specific area.
 - Size and color variation information. This is used to generate variations in the appearance of objects, and is particularly useful when billboards are used to render trees.
- A list of all possible layers on the next level that can follow this one (see Figure 4).

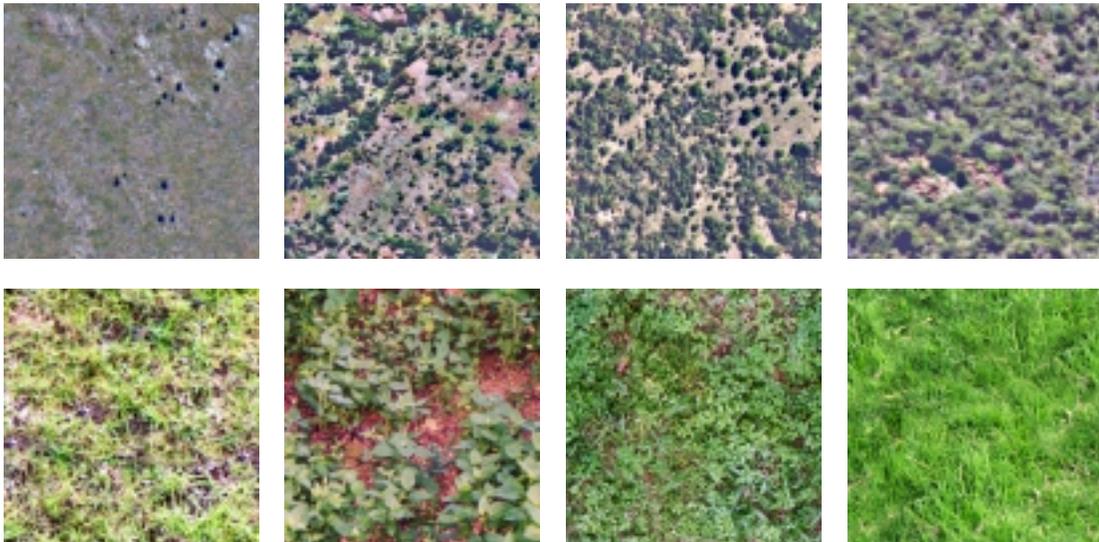


Fig. 3. The top four textures define four possible types of ground cover for savanna, all with different amounts of tree cover. These are layer zero ground covers (Table 2) with no 3D plants present, and were obtained from aerial photographs. The bottom four textures define ground cover on layer eight showing small plants, grass and soil

5.2 Combining Layers into Ecosystems

To facilitate in a diverse environment with the minimum of data, ecosystems are constructed from a tree of possible eco-layers. Each layer in the ecosystem defines which layers on the next level can exist under it. A schematic presentation of this can be seen in Figure 4. While there can be no trees at all in the grassland, it is possible for areas with partial or dense tree cover to have grass underneath them. In general, layers with many plants will have sparse layers under them, while layers with few plants will have more dense layers underneath them due to competition for sunlight.

When an area is rendered, the objects rendered are the sum of all the all the objects on this layer, and all the layers above the current one. No object intersection detection is done. The ground cover used, is the ground cover for the current layer.

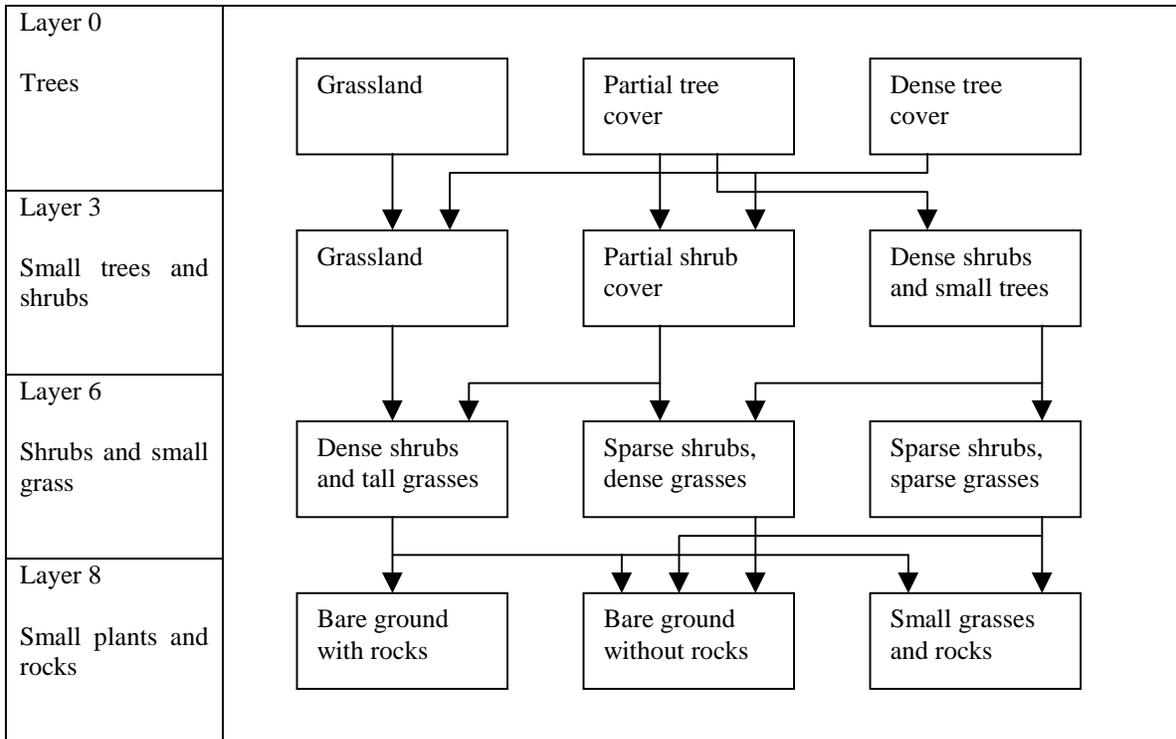


Fig. 4. A schematic presentation of possible combinations of ecosystem layers. By combining different layers it is possible to get much more variation in ecosystems. Some of the layers have been left out for clarity

6 Run-time Calculation of Ecosystems

The modeling of ecosystems can be divided into the five processes shown in Table 3. This section looks at their specific implementation.

Table 3. Pseudo-code showing the basic operations used to build a tile

```

CacheTile()
{
    BuildElevationGrid();
    // Extract a 17x17 elevation grid from the database
    BuildRelativeElevationGrid();
    // Builds a 17x17 relative elevation grid
    BuildEcosystemGrid();
    // Builds a 17x17 grid of assigned ecosystems
    CalculatePlantLayout();
    // Solve the plants and add them to the tile
    // Section 6.2
    BuildTexture();
    // Build a representative ground-cover texture
    // Section 6.3
}

```

6.1 Building the Elevation Grid

The elevation grid is a direct copy of the elevation data stored in the database (Figure 5a). The database has to define an elevation grid for all tiles on level 0. Most higher level tiles will not have elevation data in the database due to the availability of data and the amount of RAM available to store elevation data.

If no elevation data exist in the database, the elevation grid will be constructed from the parent tile's elevation grid. 9x9 elevations from the parent's elevation grid are copied to every second position in the child's elevation grid. It is interpolated linearly and a small random offset is added to ensure variation. This is similar to a fractal height field generator.

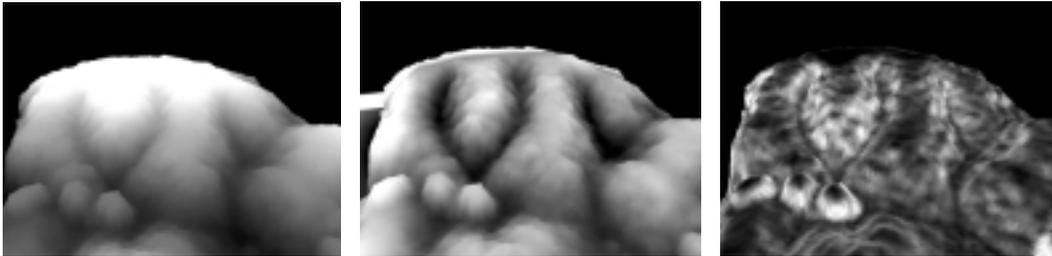


Fig. 5. Screenshots showing (from left to right) elevation, relative elevation and slope as calculated for a mountainous area

6.2 Building the Relative Elevation Grid

Each tile in the database has four average elevation values, one for each corner. These four values are retrieved from the database, and interpolated to generate a 17x17 grid of average elevations for this tile. While the database stores enough information to calculate quadratic interpolation, linear interpolation was implemented for speed. As can be seen in Figure 5b, the results of linear interpolation is convincing. The relative elevation (Figure 5b) is the difference between the elevation and the average elevation.

If a tile does not exist in the database (and its elevation grid is interpolated from its direct parent), no average elevation can be calculated. The relative elevation is calculated by linearly interpolating the relative elevation of the parent tile. This is halved, and perturbed again with the same random offsets that was added when interpolating the elevation grid. While this is not an accurate mathematical solution, it does yield acceptable results while being fast.

6.3 Building the Ecosystem Grid

The ecosystem grid is a 17x17 grid of ecosystem types coinciding with the 17x17 elevation grid defined for the tile. A complete evaluation of the parameters is done for each of the positions in the grid, and an ecosystem assigned to that element. Using a 17x17 grid instead of 16x16 allows for overlap between tiles. By accepting the penalty of re-computing data, tiles can be isolated from each other, and solved with no knowledge of any other tiles in the area other than its direct parent.

Table 4. Pseudo-code showing the calculation of the ecosystem grid

```
BuildEcosystemGrid()
{
    for (y=0; y<17; y++)
    {
        for (x=0; x<17; x++)
        {
            Eco[y][x] = CalculateEcosystem();
            // step through the grid, and calculate the
            // ecosystem with the highest probability at
            // each position
        }
    }
}
```

```

    }
}
CalculateEcosystem()
{
    CalculateSlope(); // Equation (3)
    for (i=0; i<NumEcosystems; i++)
    {
        CalculateSlopeSkew(); // Equation (7)
        prob[i] = CalcEcoWeight( // Equations (8) and (9)
            elevation + Skew,
            relative_elevation,
            slope )
            + (random offset);
    }
    return (the ecosystem with the biggest probability);
}

```

Calculate Slope. A 17x17 grid of slope values in both the x and y directions are defined by equations (1) and (2). The slope is defined in equation (3) and the result can be seen in Figure 5c. Unlike the other variables, slope is solved on demand as needed and not saved in a grid, since it is not used anywhere else.

$$delX[y][x] = (elevation[y][x+1] - elevation[y][x-1]) / 2 \quad (1)$$

$$delY[y][x] = (elevation[y+1][x] - elevation[y-1][x]) / 2 \quad (2)$$

$$Slope = \sqrt{delX^2 + delY^2} \quad (3)$$

All three of the above variables have to be normalized before they can be used in the probability equations.

$$delX_{norm} = \frac{delX}{\sqrt{delX^2 + 1}} \quad (4)$$

$$delY_{norm} = \frac{delY}{\sqrt{delY^2 + 1}} \quad (5)$$

$$Slope_{norm} = \frac{Slope}{\sqrt{Slope^2 + 1}} \quad (6)$$

Calculate Slope Skew. Slope skew is defined as an apparent change in elevation, to reflect conditions such as prevailing sunlight direction, rainfall and wind relative to the direction that a slope is facing. It is not a separate variable passed on to the ecotope modeler, but rather added to the elevation. This is done separately for each ecosystem that is evaluated. It is calculated using the normalized versions of delX and delY as defined in equations (4) and (5). The amount of skew is calculated with the following equation.

$$Skew[i] = (SkewX[i] \times delX_{norm}) + (SkewY[i] \times delY_{norm}) \quad (7)$$

SkewX[i] and SkewY[i] are defined as the amount of change in altitude for slopes in the x and y directions respectively for each ecosystem.

Calculate Ecotope Weights. For each ecosystem being evaluated, its probability is defined as the product of the individual probabilities in Table 5.

Table 5. Pseudo-code showing the calculation of an ecosystems probability

```
CalcEcoWeight()  
{  
    w_e = (probability due to elevation);  
    w_r = (probability due to relative elevation);  
    w_s = (probability due to slope);  
        // all three are calculated with equations (8) and (9)  
  
    return (w_e * w_r * w_s);  
}
```

Each ecosystem has a minimum, maximum and smoothing value defined for elevation, relative elevation and slope. The minimum and maximum values define the upper and lower boundaries where the probability is 0.5 The smoothing (S) defines how sharp this boundary is.

The value that is passed on to the function is first normalized with equation (8). X will be zero for a value exactly in the middle of the defined range, 1.0 at both the upper and lower boundaries, and bigger than 1.0 outside of the defined range.

The probability is calculated using equation (9). It will yield a value between 0.0 and 1.0. Useful values for S (smoothing) ranges from 1 (very smooth crossover) to 10 (sharply defined edge).

$$X = \text{abs}((\text{Value} - \text{AVS}) / \text{Range}) \quad (8)$$

$$w = 0.5^{X^S} \quad (9)$$

6.4 Calculate Plant Layout

For each plant type within the ecosystem, a number of plants are generated. This is a function of the density of the plants, and a random offset to ensure enough variation in the representation. For each of these plants, a position is determined by adding random offsets from the center of the tile.

Plants are added to the tile as a position and type. There is no information about the way that it will be rendered. This decision is left to the rendering engine that can choose any appropriate method for display.

6.5 Building the Ground-Cover Texture

All the ecosystems used in the database have a representative ground-cover texture associated with it (See Figure 3). For each of the 17x17 grid-points, the representative ground-cover texture is rendered into the tile's ground-cover texture using a semi transparent mask to blend textures together. Figure 6 show how a number of ground cover textures (the top row in Figure 3) was combined to form a new ground cover texture for a tile.

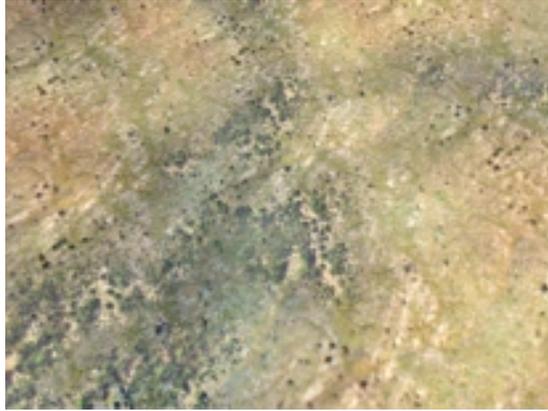


Fig. 6. Blending together of ecosystem ground-cover textures to form a new ground cover texture for a tile

6.6 Random Values

While a lot of random values are used to generate realistic variation of plants and ecosystems, it is very important to keep the results fully deterministic. This is done by pre-calculating a random lookup table and using a constant offset per tile into the table. The main reason is to ensure consistency of the visual scene. If the camera rotates through 360 degrees, the tile cache will be filled with new tiles, and all the old tiles (including their objects) will be replaced. When the camera looks in the original direction again, the tiles will be recreated and cached. The trees should be in the same position as they were before. By saving the random lookup table as part of the database, it is possible to ensure consistency across a network simulation as well.

7 Results

A program was developed in c++ using DirectX to evaluate the performance of ecotope modeling, and determine its suitability to real-time applications. This section looks at the visual appearance achieved, as well as the speed of the different sections of the algorithm. Five ecosystems were defined as shown in Table 6. All of them are level zero ecosystems. No further splitting of the tiles was done during the test.

Table 6. The five ecosystems used to evaluate the system

Ecosystem	Color		Elevation	Relative elevation	Slope
Dense bush		Min Max Sharpness	140 260 2	-0.5 0.1 1	0.0 0.7 2
Marshland		Min Max Sharpness	-50 50 2	-0.5 0 1	-0.2 0.3 2
Small bushes and grass		Min Max Sharpness	-50 350 2	0.07 0.3 1	-0.2 0.8 2
Grass on steep slopes		Min Max Sharpness	-50 350 2	0 1 1	0.7 1.2 2
Exposed rock		Min Max Sharpness	-50 350 2	-1 1 1	1.4 4.2 8

Figure 7 shows both a false color map (on the left) and a color representation (on the right) of the test scene. A single plant has been defined in the dense bush ecosystem.

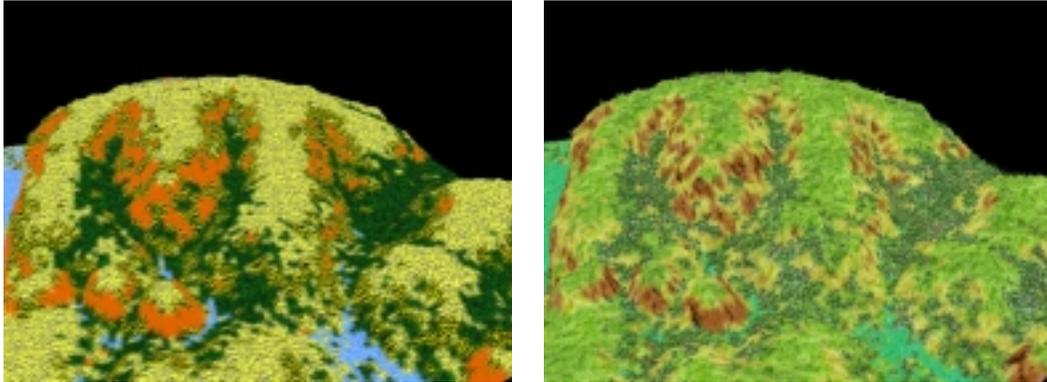


Fig. 7. False color map of the area (left) showing the placement of ecosystems. The color is in accordance with Table 6. On the right, the ground cover textures where replaced with more appropriate textures

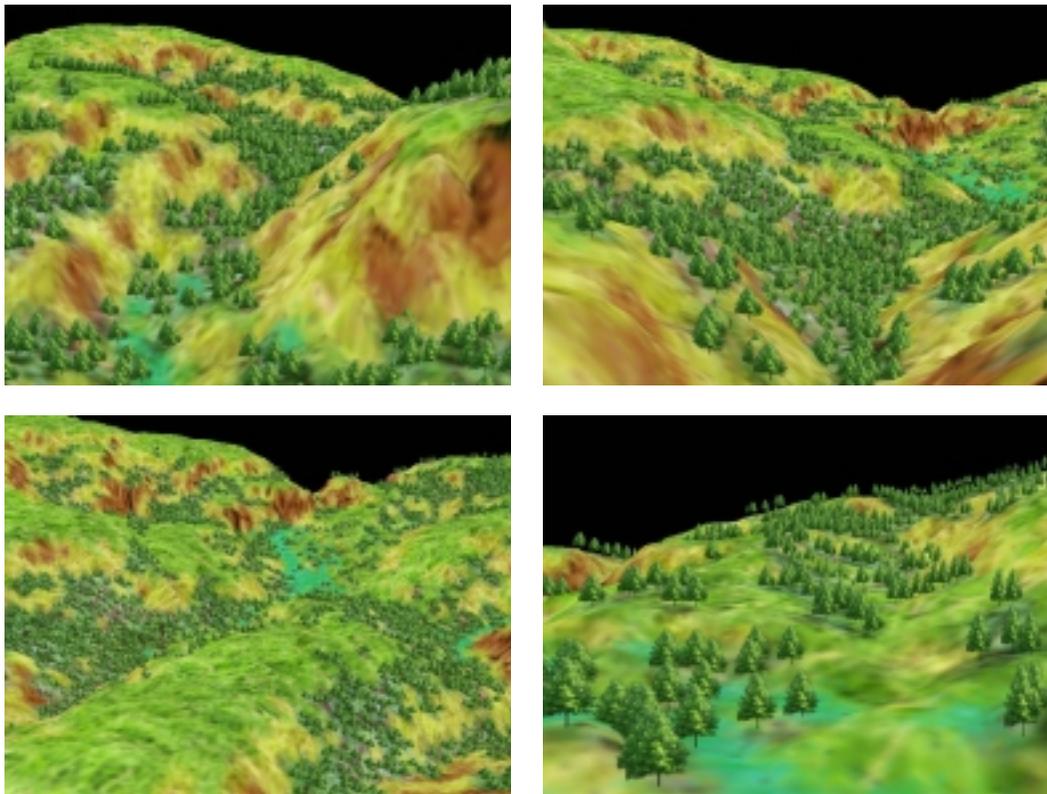


Fig. 8. Four views showing the placement of trees by the algorithm.

Figure 8 show four more views of the terrain. The lack of ecosystems on higher levels can clearly be seen close to the terrain. The trees where rendered as billboards facing towards the camera.

7.1 Performance

The time that it takes to calculate a complete tile is very important. It is possible to have fast simulations with as little as three tile updates per frame. Depending on the exact needs, this allows between one and five milliseconds to calculate a tile. Table 7 shows the average measurements for the different processes.

All measurement were made on an Intel PIII with 128 Meg of RAM, and a GeForce graphics card. The only optimizations done was in algorithm design.

Table 7. Time (in ms) to calculate different sections of a tile

Total	4.50 ms
Elevation and relative elevation	0.16 ms
Ecosystem placement and slope	3.60 ms
Calculate plants	0.20 ms
Build textures	0.35 ms

8 Conclusion

Programs like WCS have shown that very realistic images can be obtained from the modeling of ecotopes. I have presented a framework in which these ideas can be simplified to allow the real-time modeling of ecotopes. The advantages of this algorithm is as follows:

1. Compression of natural landscapes. All of the plants are calculated at runtime from a very small description. All the ecosystem information for a complete scene can be described in less than a Meg. This allows the algorithm to run efficiently on machines with limited RAM, freeing up memory resources for other processes.
2. Near real-time execution. Currently the algorithm needs about 15ms per frame to model ecosystems. With optimizations it would be possible to reduce this to as little as 6ms per frame. This is adequate for 30 fps screen updates, and will in the near future (due to faster computers) be fast enough to deliver 60 fps update rates.

References

1. 3D Nature (2000). World Construction Set 5 Users Manual. www.3dnature.com
2. Planetside Software (2000). Terragen Documentation. www.planetside.co.uk
3. Geomantics Ltd (1998). Genesis II Documentation. www.geomantics.com