# Simulating Desert Scenery

Bedřich Beneš
Dep. of Computer Science
ITESM CCM México D.F.
Bedrich.Benes@itesm.mx

Toney Roa
Dep. of Computer Science
ITESM CEM México D.F.
tjroa@itesm.mx

## ABSTRACT

An algorithm for simulating wind-ripples and moving sand is extended by the detection of fixed objects. This permits us simulation and animation of sand interacting with objects like houses, highways, cactuses, etc. Sand is accumulated on the windward side of an obstacle and the sand relocation and wind-ripples formation is diminished on the leeward side. The wind shadow depends on the object's geometry and the wind speed and direction. Sand tongues are formed as the result of the sand motion.

## Keywords

Desert scenery, visual simulation, erosion, procedural modeling, regular height fields

## 1. Introduction

Physically based simulation and procedural modeling of sand and desert sceneries have a significant importance in computer graphics. Desert scenes that could be possibly modeled by hand for quite a long time with problematic results depending on the human experience, can be obtained fast and elegantly by defining scene geometry, physical properties, and running corresponding algorithms that simulate the sand relocation. The results of the simulations are geometric models of three-dimensional virtual scenes that resembles reality.

Deserts are areas with a total rainfall less than 25 cm/year. Deserts are usually surrounded by areas having rainfall between 25–50 cm/year. Surprisingly, deserts usually do not contain much sand. The majority of deserts are rocky. The biggest desert in the world, the Sahara, has only about 10% sand.

Deserts are mostly located about $30^o$ latitude on the North and South, mainly due to the direction and intensities of winds that do not bring water to these areas. Wind, in this indirect way, presents the most important factor forming deserts.

The principal cause of desert erosion is water. Short and intensive rainfalls cause important changes in the face of the surface. At the same time the thermal shocks, high changes of temperature in the day and the night, cause dissolution of the rocks and stones that continuously change into sand.

The main factor forming the shape of sandy deserts is wind [Pye90]. Wind tends to move sand in its direction and this dynamic process is slowed down by the inner sand friction.

In this paper, we describe a procedural algorithm that significantly improves the previously described algorithm of Onoue and Nishita [Onoue00] by a simulation of sand interacting with objects. Sand grains are accumulated on windward sides of the obstacles and the sand motion and wind-ripples formation is diminished on the leeward side. The presence of obstacles causes formation of a wind shadow that depends on the object's geometry, the wind speed, and direction. This influences the wind-ripples formation.

## 2. Previous Work

The computer graphics community has focused on artificial terrains for more that twenty years.

In the fundamental book [Mande82] Mandelbrot describes algorithms for generating fractal terrains using the random midpoint displacement method and random faults. So-called noise functions, such as fractional Brownian motion, the Perlin function, etc., are suitable for texture, as well as for terrain generation. Since this paper deals with terrain erosions, we refer the reader to the tutorial [Deuss03] for an overview of the fractal techniques.

One difficulty with fractals techniques is that they



***WSCG SHORT Communication papers proceedings***
*WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.*

are not able to simulate eroded terrains that are naturally presented in our environment. Two approaches whose goal is to provide visual models of eroded terrains exist. The first class generates the eroded terrains directly [Kelle88, Nagas97, Prusi93, Sumne99], whereas the others [Beneš02, Chiba97, Gaill97, Li93, Musgr89] simulate an erosion process that can be applied either to real data or to a generated one.

Musgrave *et al* [Musgr89] introduces two *ad-hoc* erosion algorithms. The first works a low-pass filter simulating material deposition caused by thermal shocks. A part of the material melts and falls down. This forms typical slopes and smoothes the shape of a terrain. More complicated terrains can be obtained by the second algorithm - from the same paper - that simulates hydraulic erosion. Water running on the surface captures some part of the material that is moved to another place and deposited.

Beneš and Forsbach [Beneš01] show a Run Length Encoding-like layered data structure that is designed for erosion simulation algorithms and allows for significant data compression without slowing down the efficiency of the erosion algorithms.

The same authors describe an hydraulic erosion algorithm in [Beneš02]. The material that is captured by running water also travels inside the water. As the water evaporates the material settles down that allows a simulation of such phenomena as drying pools of water.

Chiba *et al* [Chiba97] use a physically inspired model to simulate river channels forming terrains. The algorithm is user-assisted. The user first defines the river channels interactively and the algorithm simulates the corresponding eroded terrain.

Simulation of sand and deserts was the focus of the paper of Onoue and Nishita [Onoue00]; in it they describe the formation of wind-ripples and dunes caused by wind. We extend this algorithm in this paper and its detailed description follows in the Section 3.

A technique simulating interaction of sand and snow with virtual objects is described in [Sumne99]. The amount of material that is moved when a terrain surface is penetrated by an object is determined. The material is deposited in the near position and eroded using a relaxation algorithm. This gives us visually plausible results of footprints in the sand, mud, and snow.

Li and Moshell [Li93] show a physically based volume preserving an algorithm that attempts to simulate virtual objects digging holes in virtual terrains. A somewhat complicated algorithm provides fast and realistic results of bulldozers and scoop loaders. The main contribution of this paper is the part of the algorithm that describes slippage of a part of terrain that is applied to interactive manipulations.

This paper continues with the description of the algorithm of Onoue and Nishita [Onoue00]. Extension of the algorithm by wind obstacles and the interaction with wind is described in Section 4. Implementation and results are then shown and Section 6 concludes the paper.

Recently Onoue and Nishita [Onoue03] presented a virtual sandbox; an interactive technique for an interactive manipulation of sand.

## 3. Wind-Ripples Simulation Algorithm

The algorithm of Onoue and Nishita [Onoue00] describes the formation of wind-ripples and dunes caused by wind. The justification of this technique is based on the fact that light and subtle sand grains can be easily captured by wind, and moved to another location. When deposited, they fall down the hill, until the so-called talus angle is reached. An example of a simulation of the wind-ripples formation is shown in Figure 1

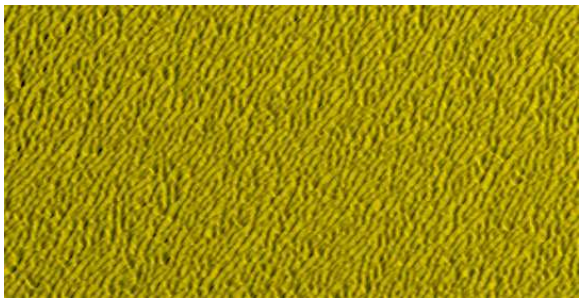

Figure 1: Wind-ripples formation

### 3.1. Data Structures

The algorithm works on a regular height field that can be represented as a two dimensional matrix where each element represents the height of the material. This is a very common representation and it is frequently used for erosion simulation. There are also algorithms working with compressed layered data [Beneš01], or with sets of triangles, typically used in Geographic Information Systems.

Regular height fields are not very well suited for real-time rendering. Planar areas are as equally tessellated as wrinkled ones, and no level of detail technique is native to them. There are many algorithms solving these rendering flaws, but from the viewpoint of the simulation, the algorithm must run for each element of the height field.

### 3.2. Basic Algorithm

Let $\mathbf{p} = (i, j)$ and $0 \leq i, j \leq n - 1$ denote the 2D discrete vector - a position in a height field. The regular height field is represented as a two-dimensional matrix denoted by $hf$ of dimension $n \times n$. Elements of a $hf$

are denoted by $hf(\mathbf{p})$. If this is clear from the context, we will omit the position vector $\mathbf{p}$ and we will write $hf$.

The simulation runs in discrete time steps $\Delta t$. Let $hf$ denote the height filed in the current time $t$ and $hf'$ the same height field in the next step $t + \Delta t$.

Each particle of sand – the smallest amount of sand that can be captured by wind – can be the subject of three processes, *saltation*, *suspension*, and *creep*.

Saltation occurs when the wind captures particles of sand, moves them in the air, and relocates to a different position (see in Figure 2).

Suspension is capturing dust particles and moving them indefinitely.

Creep means rolling the suspended particles down the hill reaching the energetically best position. This process can be thought of as a low-pass filter smoothing the terrain.

### 3.3. Saltation

We describe the process of saltation as shown in Figure 2.

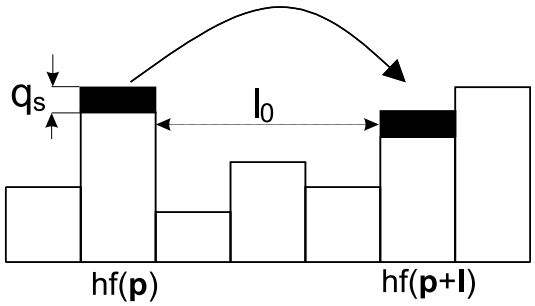


Figure 2: Schematic description of the process of saltation

An amount of sand at the position $\mathbf{p}$, denoted by $q_s(\mathbf{p})$, is captured by wind as described by the following equation

$$hf'(\mathbf{p}) = hf(\mathbf{p}) - q_s(\mathbf{p}) \quad (1)$$

This amount is deposited in the position

$$hf'(\mathbf{p} + \mathbf{l}) = hf(\mathbf{p} + \mathbf{l}) + q_s(\mathbf{p}) \quad (2)$$

Where $\mathbf{l} = (l_i, l_j)$ is so-called horizontal displacement vector (the "hop" vector) with components:

$$l_i(\mathbf{p}) = (l_{0i} + w_i hf)(1 - \tanh \tfrac{\partial hf}{\partial i}) \quad (3)$$

$$l_j(\mathbf{p}) = (l_{0j} + w_j hf)(1 - \tanh \tfrac{\partial hf}{\partial j}) \quad (4)$$

The vector $w(\mathbf{p}) = (w_i, w_j)$ represents the wind. Its size corresponds to the wind intensity and its direction to the wind direction. The vector $(l_{0i}, l_{0j})$ is the user defined value that corresponds to the average hop.

The function $q_s(\mathbf{p})$, from the equations (1) and (2), reflects the amount of transferred material. This depends on the average amount of material, denoted by $q_0$, and the height of the terrain element, reflected by the gradient. This is given by:

$$q_s(\mathbf{p}) = q_0(1 + \tanh(\nabla hf)) \quad (5)$$

There are many parameters that might seem strange. The $q_0$ is the average amount of the transferred sand. We work with values in the span $[0.1, 1.0]$. Values higher that one cause the system to become unstable and no wind-ripples are formed. Values close to one cause wilder terrain appearance, values close to zero make the terrain smoother as can be seen in Figure 3.

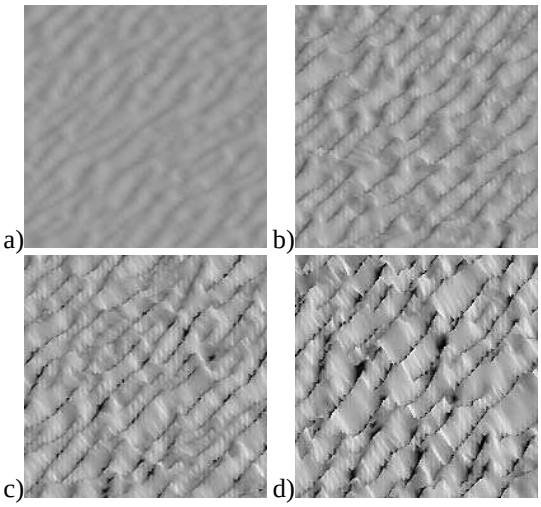


Figure 3: Higher coefficient $q_0$ causes wilder wind-ripples formation. a) $q_0$=0.1 b) $q_0$=0.5 c) $q_0$=0.7 d) $q_0$=0.9

The parameters $l_0$ and $w$ influence the frequency of the wind-ripples. Values out of the range $[0, 1]$ cause system instability and no wind-ripples formation.

### 3.4. Creep

The last material transport process - *creep* - smoothes the artificial surface. Particle of sand can be located on an unstable position and will fall down. The simulation of the process of creep takes each element of the $hf$ and compares it with its neighbors. If it is too high it is smoothed. This is done as weighted average of the heights of the closest neighbors of a cell that is subtracted from the cell height. Special care must be taken not to add or subtract the total volume of material in this process.

Physically, this concept corresponds to rolling the grains of sand down the hill. This is also known as the particle deposition process and it is commonly used for artificial terrain generation.

## 4. Wind Obstacles

The above described algorithm works for terrains that are homogeneous, without obstacles, and the wind blows in the same direction and with the same intensity all the time. It is easy to assume that the wind depends on the time, so the $w(\mathbf{p}) = w(t)(\mathbf{p})$. It is more difficult to put some obstacles there.

We have extended the previously described algorithm by interaction with obstacles that cause two fundamental effects. The first is the *accumulation* of the material on the windward side of the object. The second factor caused by the wind obstacles is the wind *intensity decrease* behind the object on the leeward side. This diminishes or stops the wind-ripples formation in the wind shadow.

### 4.1. Material Accumulation

The material accumulation is simulated in the following way: in the saltation step (see equations (1) and (2)) the material is moved from its position $\mathbf{p}$ and put onto the new position $\mathbf{p} + \mathbf{l}$. If the new height $hf'$ is located inside an object, we move it outside, to the border as depicted in Figure 4. Here the $w$ is the wind direction, $hf$ is the actual position, and $hf'_{updt}$ is the updated new position. The material that should be transferred into the object (left) is deposited outside (right).

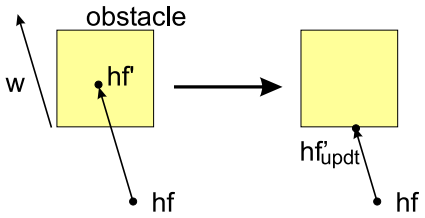


Figure 4: Material cannot be transferred inside an object (left) and is deposited outside

### 4.2. Creep

The change of material location causes accumulation of high volumes of material on the object boundaries. In the original algorithm, the creep is applied $n\times$ over the surface which assures visual plausibility of the results.

It would be possible to implement an adaptive algorithm that would search for the obstacles and smooth the peaks located close to them. This is not conceptually correct nor even necessary. We modified the algorithm of the thermal erosion [Musgr89].

To achieve fast and efficient smoothing of the terrain, we store the local gradient of each element. This is calculated once with the $O(m)$ time complexity ($m = n \times n$ is the number of the elements of the $hf$). The algorithm tests the gradient of each element and if the gradient is greater than the corresponding critical angle, called *the talus angle*, we move the maximum possible amount of material to the neighbors. The amount is distributed proportionally to the size of the gradient. This step is applied few times, until there is no material to move. Apparently, no material can be moved inside the obstacle. The example in Figure 5 shows the result of a simulation.

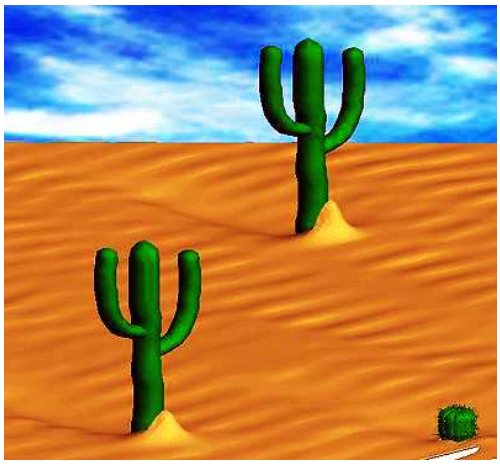

Figure 5: Material is accumulated on the windward side of cactuses

### 4.3. Wind Shadows

Another aspect that we simulate is the wind shadows caused by the presence of obstacles. Behind an obstacle the intensity of the wind is diminished and increases with the increasing distance. The distance depends on the size of the obstacle.

We present a purely *ad hoc* technique attempting to produce visually plausible results at interactive framerates. A physically correct approach would involve a study of the wind-obstacle interaction, obstacle surface viscosity, wind vortex, etc. We present a geometric solution that gives visually plausible results fast.

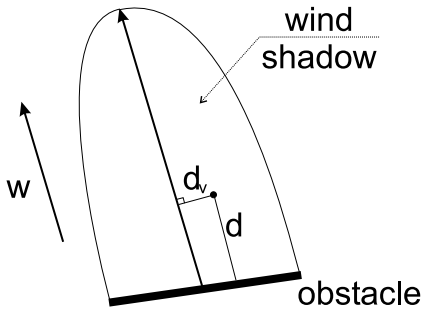


Figure 6: The wind intensity behind an obstacle depends on the distance from the obstacle and from the medial axis defined by the obstacle and the wind direction

A wind shadow depends on the height of the object and on its projected width. We suppose the wind intensity to be zero exactly behind an obstacle, and increase linearly.

What plays an important role is the shape of the wind shadow. We do not modify the wind direction in the wind shadow (see equation (4)); what is modified is only its intensity. This can be justified by the wind-ripple formation algorithm described in Section 3. Small wind intensity does not cause the wind-ripple formation that produces the shadow area correctly.

The intensity of the wind inside the wind shadow depends on the distance from the object (denoted by $v$), its height $h$, and the distance from the medial axis. This is defined by the projected center of the obstacle to the plane perpendicular to the direction of wind passing through the middle of the object (denoted by $d_v$). The external boundary of the shadow $s$ is parabola and the corresponding equation is

$$s = \frac{w}{2}(1 - d^2).$$

To define the wind shadow, we first project the entire object onto the plane that is perpendicular to the wind direction and passes throught the object's center. The object's center and the wind direction also define the main axis of the shadow. Each point of $hf$, that should be the subject of saltation, is tested whether or not it lies inside the shadow. In the positive case the wind intensity is decreased linearly as a function of the distance $v$ and the object height $h$. Points outside the shadow boundary are not influenced by the object shadow. The length of the wind shadow depends on the wind intensity and the height of the object. The width of the shadow depends on the projected area only.

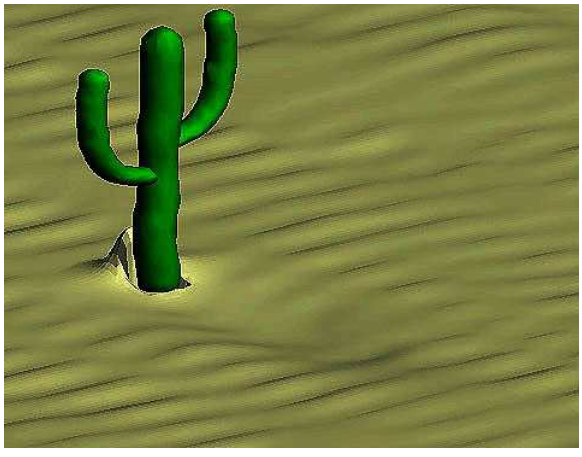

Figure 7: A wind shadow formed by a *Stenocactus coptonogonus*

Figure 7 shows the wind shadow formed behind a cactus in a desert scene and the accumulated material. The cactus was created in MAYA and exported to OpenGL.

## 5. Implementation and Results

The algorithm is simple, works fast, and we were able to run our scenes interactively on an IBM PC running on 1.5GHz up to the height field resolution $1000 \times 1000$. The entire implementation is done in C++ and uses OpenGL for displaying. All examples shown in this paper run interactively with the frame-rate between 4-10 fps.

Figure 9 shows an example of the sand accumulation due to the wind and obstacle interaction. The wind-ripples are formed correctly. As clearly visible there is a layer of accumulated sand in the front of them.

Another example in Figure 8 shows sand tongues that are formed on the highway due to blowing wind. This is the natural result of the wind and obstacle interactions.

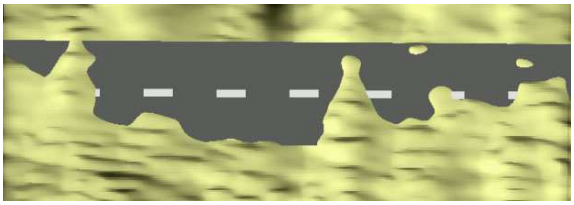

Figure 8: Sand tongues formed on a highway

## 6. Conclusions and Future Work

An extension of the algorithm of Onoue and Nishita [Onoue00] is presented. The original algorithm simulates wind-ripples formed by blowing wind on sandy deserts. The main contribution of our extension is the simulation of wind obstacles interacting with sand. The intensity of the wind is modified in the wind shadows according to the simplified geometry of the obstacle. The intensity of the wind is decreased which diminishes the wind-ripples formation. Another contribution is the accumulation of material in front of an obstacle.

The algorithm is fast, simple, and runs efficiently on an average IBM PC without any special requirements to hardware acceleration.

Future work should definitively focus on the physics-based wind and material transport simulation. There are many recently published algorithms in the area of computer graphics that could be applied here.

On the other hand, we believe that these simple procedural algorithm providing visually plausible results, can be used in computer games, the movie industry, etc.

## 7. REFERENCES


[Beneš01] B. Beneš and R. Forsbach. Layered Data Structure for Visual Simulation of Terrain Erosion. In Tonisao Kunii, editor, *IEEE Proceedings of the Spring Conference on Computer Graphics*, pages 80–86. IEEE Computer Society, 2001.

[Beneš02] B. Beneš and R. Forsbach. Visual Simulation of Hydraulic Erosion. *Journal of WSCG*, 1(Special Issue)pages 79–86, 2002.

[Chiba97] N. Chiba, K. Muraoka, and K. Fujita. An Erosion Model Based on Velocity Fields for the Visual Simulation of Mountain Scenery. *The Journal of Visualization and Computer Animation*,9 pages 185–194, 1997.

[Deuss03] O. Deussen, R. Fedkiw, F.K. Musgrave, P. Prusinkiewicz, and J. Stam. D.D. Ebert, organizer, Course 41: Simulating Nature: Realistic and Interactive Techniques. *Siggraph 2003 Course Notes*, pages 1–301, 2003.

[Gaill97] C. Gaillard, F. Zagolski, and F. Bonn. Modelling of Human Dimension on Soil Erosion Processes for Remote Sensing Applications. In *IEEE Geoescience and Remote Sensing Symposium*, volume 4, pages 2137–2139, 1997.

[Kelle88] A.D. Kelley, M.C. Malin, and G.M. Nielson. Terrain Simulation Using a Model of Stream Erosion. *Computer Graphics*, 22(4) pages 263–268, 1988.

[Li93] X. Li and M. Moshell. Modeling Soil: Realtime Dynamic Models for Soil Slippage and Manipulation. In *Proceedings of SIGGRAPH'93*, volume 27(4) of *Annual Conference Series*, pages 361–368, 1993.

[Mande82] B.B. Mandelbrot. *The Fractal Geometry of Nature*. W.H.Freeman, San Francisco, 1982.

[Musgr89] F.K.Musgrave, and C.E.Kolb, and R.S.Mace, . The Synthesis and Rendering of Eroded Fractal Terrains. In *Proceedings of Siggraph'89*, volume 23(3) of *Annual Conference Series*, pages 44–50, 1989.

[Nagas97] K. Nagashima. Computer Generation of Eroded Valley and Mountain Terrains. *The Visual Computer*, 13 pages 456–464, 1997.

[Onoue00] K. Onoue and T. Nishita. A Method for Modeling and Rendering Dunes with Wind-ripples. In *Proceedings of Pacific Graphics'00*, pages 427–428, 2000.

[Onoue03] K. Onoue and T. Nishita. Virtual Sandbox. In *Proceedings of Pacific Graphics'03*, in printing, 2003.

[Prusi93] P. Prusinkiewicz and M. Hammel. A Fractal Model of Mountains with Rives. In *Proceedings of Graphics Interface'93*, volume 30(4), pages 174–180, 1993.

[Pye90] K. Pye and H. Tsoar. *Aeolian Sand and Sand Dunes*. Unwin Hyman, London, 1990.

[Sumne99] R.W. Sumner, J.F. O'Brien, and JK. Hodgins. Animating Sand, Mud, and Snow. *Computer Graphics Forum*, 18(1) pages 22–31, 1999.


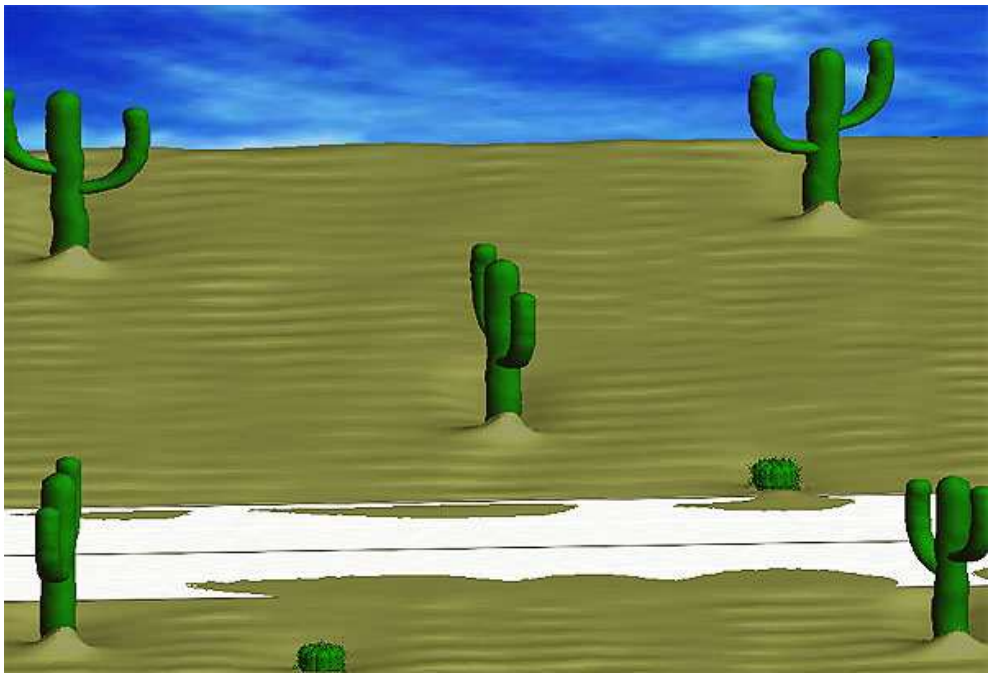

Figure 9: Down to Los Cabos