Master's Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Comparison of Sand Simulation Methods and Their Application to Snow Simulation

Bc. Vojtěch Cimbura

Supervisor: Ing. Jaroslav Sloup
Field of study: Open informatics
Subfield: Computer graphics
May 2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Cimbura**	Jméno: **Vojtěch**	Osobní číslo: **474637**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Zadávající katedra/ústav:	**Katedra počítačové grafiky a interakce**		
Studijní program:	**Otevřená informatika**		
Specializace:	**Počítačová grafika**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Porovnání metod simulace písku a jejich aplikace na simulaci sněhu

Název diplomové práce anglicky:

Comparison of Sand Simulation Methods and Their Application to Snow Simulation

Pokyny pro vypracování:

Seznamte se s mechanismy vzniku písečných dun a metodami jejich simulace [1-5]. Vybrané metody naimplementujte a porovnejte jejich schopnost generovat různé typy dun (parabolické, barchany, transverzální, podélné, hvězdicovité). Vytvořte interaktivní aplikaci, která bude zobrazovat vygenerované písečné útvary a umožní měnit simulační parametry porovnávaných metod. Vygenerované útvary porovnejte s fotografiemi odpovídajících typů dun a zhodnoďte jejich věrohodnost.
Prostudujte fyzikální princip přetváření povrchu zasněžené krajiny vlivem větrné eroze a následným transportem a sedimentací sněhu. Zaměřte se na odlišnosti chování písku a sněhu a jejich vliv na tvar vznikajících útvarů. Pokuste se rozšířit vybranou metodu simulace písečných dun o časově proměnné parametry sněhu (např. přilnavost) tak, jak jsou popsané v článku [6] a prostuduje jejich vliv na výsledný vzhled krajiny.
Implementaci proveďte v C/C++ s využitím OpenGL a CUDA.

Seznam doporučené literatury:

[1] B. T. Werner: Eolian Dunes: Computer Simulations and Attractor Interpretation. Geology, vol.23, no.12, p.1107-1110, 1995.
[2] Joanna M. Nield, and Andreas C.W. Baas: Investigating Parabolic and Nebkha Dune Formation Using a Cellular Automaton Modelling Approach. Earth Surface Processes and Landforms, vol.33, no.5, p.724-740, 2008.
[3] Ning Wang, Bao-Gang Hu: Real-Time Simulation of Aeolian Sand Movement and Sand Ripple Evolution: A Method Based on the Physics of Blown Sand. Journal of Computer Science and Technology, vol.27, no.1, p.135-146, 2012.
[4] Ricardo Carretero-González, Steven Bishop, Hiroshi Momiji, Andrew Warren: Modelling Desert Dune Fields Based on Discrete Dynamics. Discrete Dynamics in Nature and Society, vol. 7, issue 1, p.7-17, 2002.
[5] Axel Paris, Adrien Peytavie, Eric Guerin, Oscar Argudo, Eric Galin: Desertscape Simulation. Computer Graphics Forum, vol.38, no.7, p.47-55, 2019.
[6] S. Filhol, M. Sturm: Snow Bedforms: A Review, New Data, and a Formation Model. Journal of Geophysical Research: Earth Surface, vol.120, no.9, p.1645-1669, 2015.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jaroslav Sloup **Katedra počítačové grafiky a interakce**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **01.02.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

Ing. Jaroslav Sloup podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.

Datum převzetí zadání	Podpis studenta

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Ing. Jaroslav Sloup, for all his help, advice and suggestions. I also appreciate all the support, encouragement and understanding I received from my family. To my friends, thank you for your support through my studies.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, 20. May 2022

....

Abstract

This Master's thesis is focused on sand simulation methods analysis in the field of computer graphics and aims to compare these methods and discuss their disadvantages. The thesis also reviews sand and snow simulation similarities and compares their behavior. An application is designed and implemented to analyze existing simulation approaches within this work. Such application may be used for natural phenomena prediction or in the study of geological principles.

Keywords: Computer graphics, Simulation, Sand, Snow

Supervisor: Ing. Jaroslav Sloup
Karlovo náměstí 13
121 35 Prague 2
Czech Republic

Abstrakt

Tato diplomová práce je zaměřena na analýzu existujících metod simulace písku v rámci počítačové grafiky a cílí na porovnání těchto metod zároveň s analýzou jejich nedostatků. Práce také zkoumá podobnost simulace písku a sněhu a porovnává jejich chování. V rámci práce je navrhnuta a vytvořena aplikace umožňující analýzu existujících simulačních metod. Taková aplikace může najít uplatnění v predikci přírodních jevů nebo při zkoumání geologických principů.

Klíčová slova: Počítačová grafika, Simulace, Písek, Sníh

Překlad názvu: Porovnání metod simulace písku a jejich aplikace na simulaci sněhu

Contents

Figures

Tables

Chapter 1

Introduction

Natural phenomena have captured human attention since time immemorial. People study the processes in nature in great depth to understand them, yet there are plenty of unknown areas which are not explained or are somewhat inexplicable, despite modern technology. Those imperfections prevent full knowledge of natural phenomena. However, it does not stop scientists from replicating those processes the best they can.

The growth of the computer industry made it possible to put theoretical models of material movement in nature to a test. This way, scientists were able to verify the functionality of simulation models and develop new simulation approaches. Such a simulation model can reconstruct events based on known data to improve the understanding of processes and interactions in nature or predict future landscape development.

Figure 1.1: Desert dunes near Cadiz Dry Lake, California (from [27]).

The focus of this thesis is given on sand simulation techniques currently known in computer graphics. Also, snow simulation approaches will be discussed because there are some similarities between sand and snow transport principles. We will review both sand and snow properties from the geological point of view in Chapter 2, Geological Principles, since understanding geological principles is crucial for realistic computer simulations. Then we will examine already known approaches to sand and snow simulations in Chapter 3, Related Work. We aim to analyze discussed simulation approaches, and therefore the following Chapter 4 will introduce the approach to designing an application that meets the requirements, how the application should work and what should be achieved. The following Chapter 5 provides specific details of the used simulation algorithms. Finally, the results will be presented in Chapter 6, and the thesis will be concluded in Chapter 7.

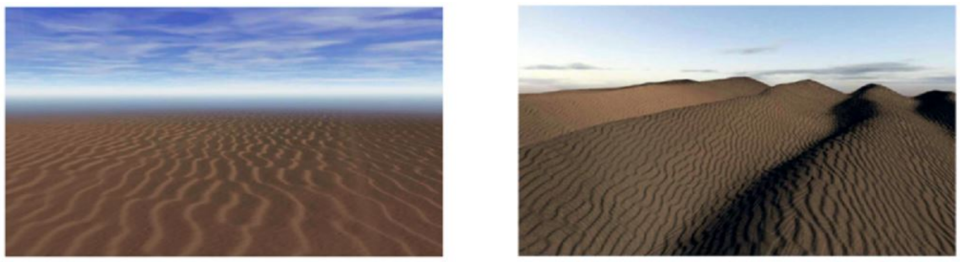


Figure 1.2: Simulated desert scenes (from [25]).

Chapter 2

Geological Principles

Sand and snow properties will be reviewed in this chapter. Considering sand, mainly factors influencing the sand transport will be investigated. Snow properties will be reviewed afterward. The crucial observation is the transport of those materials.

2.1 Sand

According to F.J. Pettijohn et al. [21], sand is a type of non-cohesive granular material which forms a network of grains. This grain network differs from other crystalline rock structures by the contact with individual grains. Crystalline rock grains are in continuous contact, whereas sand grains are in tangential contact, which means there is much space between the grains filled by air compared with other rocks. As a result, the sand has high porosity. This property is key for forming desertscapes because any fluid is quickly absorbed in the sand grain network. The amount of air between grains is proportional to the size of sand grains, defined by sand diameter, which is tightly connected with the sand definition itself. The sand grain diameter is from 62.5 µm to 2.0 mm. Particles smaller than 62.5 µm are commonly referred to as dust particles (J. F. Kok, [12]). This range is generally accepted and mostly used by geologists. Furthermore, according to ISO 14688 [4], there are three sand types: fine sand, medium sand, and coarse sand, with ranges in intervals 62.5 µm to 0.2 mm to 0.63 mm to 2.0 mm. Different sand grain types have different abilities to be transported, which is a crucial observation.

2.1.1 Sand Transport

The processes shaping the surface of Earth are called aeolian processes, from which we will be mainly focused on sand erosion, sand transport by wind, and sand deposition. Those processes are typical for forming deserts and dunes. The transport of sand particles by wind can occur in three modes, depending on wind speed and sand grain size. Sand grains of different sizes tend to be transported differently, as Jasper F. Kok [12] suggests:

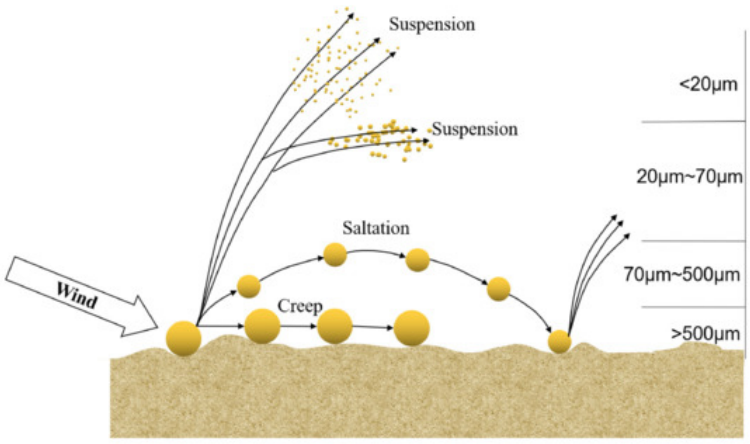


Figure 2.1: Different grain size particle movements under the effects of wind (from [16]).

Saltation: Sand particles of 100 μm diameter are the first to be moved by the increasing wind speed. As soon as a particle lifts, it 'hops' along the surface as the wind carries it. As the particle is carried, it will eventually reach a stable position after one of its 'hops.' However, as the saltating particle is bouncing along the surface, it interacts with other sand particles on the surface, which might lead to mobilizing other particles influenced by the bounce impact of the saltating particle. Interestingly, the most commonly lifted particles are dust particles, which have higher cohesive forces than sand particles carried by saltation. Here, depending on the size of the lifted dust particle, we distinguish between short-term and long-term suspension.

Suspension: Suspension is a typical transport mode for sand and dust particles of size lower than 70 μm. Moreover, we distinguish between long-term and short-term suspension. Long-term suspension is typical for dust particles of size lower than 20 μm. Those particles can be carried several days or even weeks from their place of origin and travel in the atmosphere thousands of kilometers. On the other hand, particles between 20-70 μm usually undergo short-term suspension and are not carried for great distances, unlike long-term suspended particles.

Reptation and Creep: Particles of diameter bigger than 500 µm undergo a process called reptation. Those sand particles are too heavy to be carried by the wind; they instead settle back to the soil after a short 'hop,' after less than a centimeter. Alternatively, those particles can roll or slide along the surface as they are driven by wind or undergo impacts of saltating particles in an alternative transport mode called Creep. Both Reptation and Creep are similar transport modes and are sometimes defined together as Creep.

Note that previously mentioned transport modes are not discrete. They blend instead, depending on both particle and wind properties.

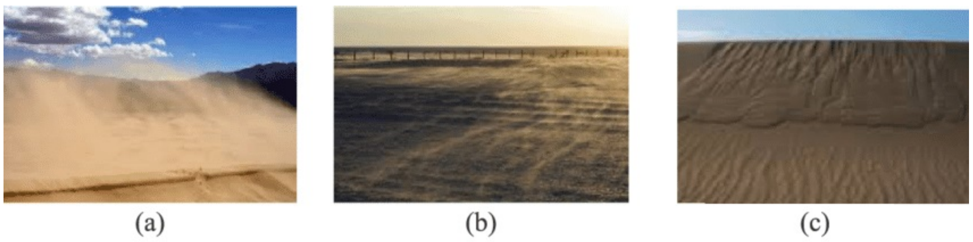


Figure 2.2: Sand transport modes. (a) Suspension, (b) Saltation, (c) Avalanching (from [17]).

2.1.2 Wind Erosion and Deposition Details

N. Lancaster [14] defines the dune initiation process as poorly understood and little studied. Nonetheless, the commonly accepted theory involves the reduction of near-surface wind speed and transport rates due to changes in aerodynamic roughness or microtopography of the sand surface. In other words, particles carried by wind will decrease their speed, and then they will be deposited. As soon as some particles are deposited, the dune has a foundation created. As the dune grows, it projects into the atmospheric boundary layer, i.e., it affects the airflow around it. The theory concerning the wind behavior around the dune is as follows.

The windward side of the dune undergoes erosion as the wind 'hits' the dune slope. Because the wind continues to change direction according to the dune shape, it accelerates up the slope as the streamlines are compressed, and the maximum speedup can be observed around the dune crest. After the wind reaches the crest and enters the leeward side of the dune, the wind speed decreases rapidly with the transport rate, resulting in sand particle deposition on the leeward side of the dune. The wind direction after it reaches the dune leeward side is more complex due to different angles between the wind and the dune crest. However, H. Tsoar [24] suggests that sand eroded on the windward side is deposited on the upper lee side, causing oversteepening.

As particles accumulate, they eventually reach the critical angle of internal friction, approximately defined as 35°. At this angle, the upper lee slope is not steady, and particles will cause avalanches until they reach a stable position. A stable position is reached as soon as the sand particles reduce the slope angle to 32° - 33°. This stabilized angle is called 'repose angle' or 'angle of repose' and will be widely used throughout this thesis. The overall visualization of wind speedup over the dune is shown in Figure 2.3.

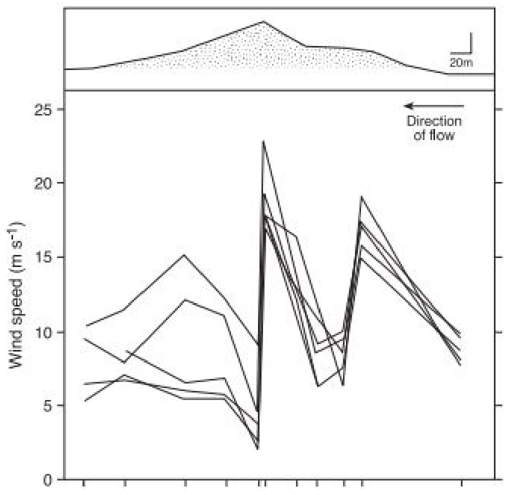


Figure 2.3: Five sets of wind velocity measurements over a linear dune in Namibia. Note the increase in velocity to the crest of the dune and the sudden reduction in the lee (from [28]).

Before we start to discuss the dune types themselves, it is necessary to introduce more wind properties. According to H. Tsoar [24], drift potential prevents vegetation from growing in the sand and stabilizing it. The accepted method of quantifying wind power is by referring to the drift potential of the wind, which is based on the sand transport equation:

$$q = Ku_*^2(u_* - u_*t) \tag{2.1}$$

where u_* is the shear velocity of the wind, u_*t is the threshold shear velocity, K depends on grain size, sand sorting, and air density, and finally, q represents the sand flux. Firstly, shear velocity, which is used to describe shear-related motion in moving fluids, wind, in our case, basically describes the stress caused by the wind applied to the surface. Threshold shear velocity intuitively is a limit when the wind speed applies enough force to initiate particle movement or deposition on the impact surface. The correlation between those variables with the sand grain diameter is shown in Figure 2.4.

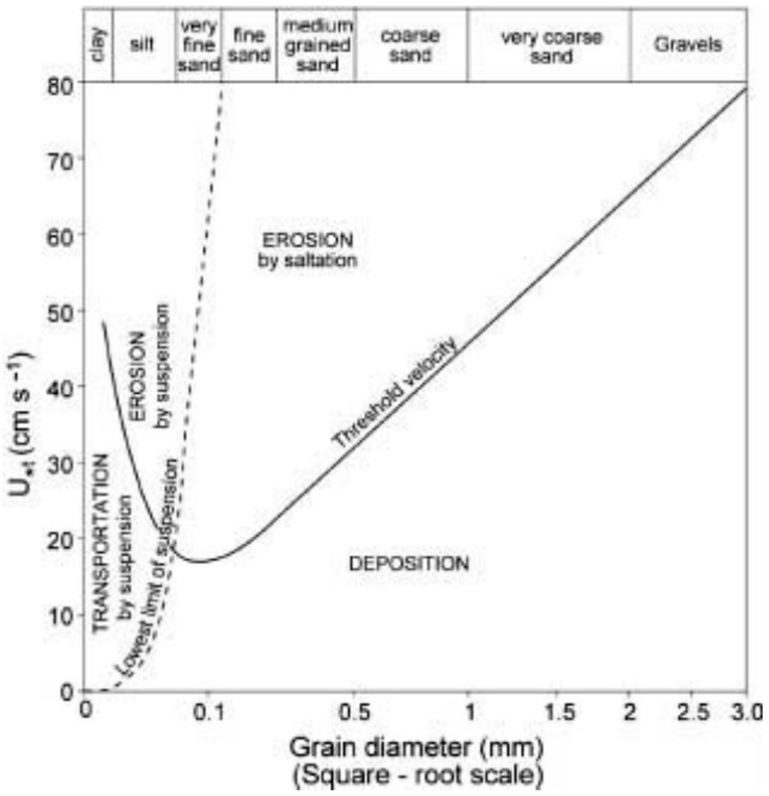


Figure 2.4: Threshold friction velocity (U_{*t}) curve for quartz grains of different diameters (solid line). The dashed line separates saltation from suspension (from [24]).

2.1.3 Dune Types

Wind speed, direction, and sand particle size are crucial for desertscape formation. This section will focus on introducing different types of dunes and describing the dune formation prerequisites. Afterward, we will focus on vegetation impact on the dune formation in Subsection 2.1.4.

Dunes are created by a series of interactions between sand particles and shearing flow [14]. Dunes can be observed at the bottom of rivers or oceans, where the flow of water forms the subaqueous dunes. We will mainly focus on flow caused by wind force for our purposes. Moreover, dunes are created and constantly developed over time as the surface responses to wind regime, such as wind speed and direction. All dune types discussed below can evolve from a variety of initial conditions. We will be mainly focused on some of the shapes shown in Figure 2.5 and on dunes in a vegetated environment.

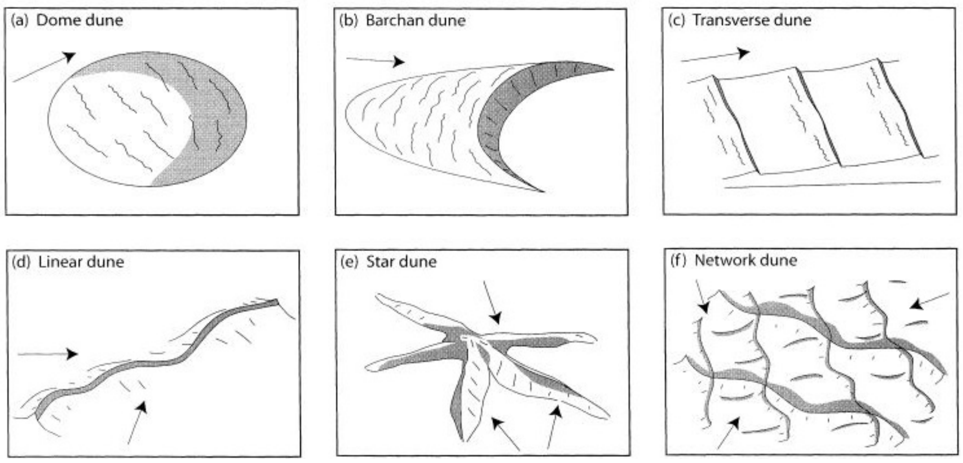


Figure 2.5: Schematic views of typical dunes, arrows show the predominant wind direction (from [1], modified).

Sand ripples: Sand ripples are typical for all sand-covered surfaces except those with rapid material deposition [14]. Their existence is closely linked with the sand transport modes saltation and reptation. Sand ripples are usually perpendicular to the wind direction. Furthermore, they can be considered an indicator of current wind direction and overall sediment transport since they are formed within minutes. Typically, sand ripples have a wavelength between 13 and 300 mm and an amplitude of 0.6 to 14 mm. Since ripples are omnipresent, as mentioned earlier, a ripple index is typically introduced, which corresponds to the ratio between ripple length and height. This way, one can compare ripples in different environments, which is particularly useful for comparing ripples created of differently sized sand grains and different strengths of wind. The ripple wavelength is a function of both sand particle size and sorting, and wind speed, i.e., ripples in coarse sands have a greater spacing than ripples in fine sands [14].

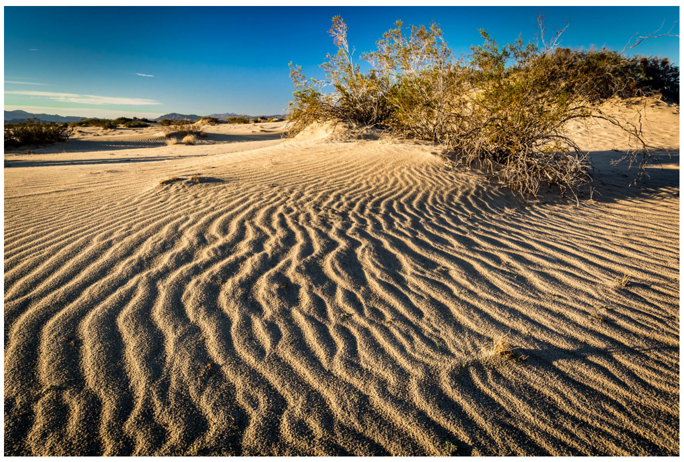


Figure 2.6: Sand ripples in vegetated sandy area (from [13]).

Barchan and Transverse Dunes: According to H. Tsoar [24], Barchan dunes and Transverse dunes are the same dune type with the difference in the amount of material available for aeolian transport. Both transverse and barchan dunes are formed by unidirectional wind direction, with a maximum angle variability of 15°. Transverse dunes are built of many barchans that have merged. Barchan dunes are characterized by an ellipsoidal shape with 'horns' extending downwind. The leeward side of a barchan is usually much steeper than the windward side [14]. The increasing wind speed causes to form the crescent shape of barchan as the wind 'hits' the windward side, diverging from the dune crest towards the flanks [24].

Linear dunes: Linear dunes, sometimes called linear seif dunes, are characterized by their length, which may exceed even 20 km, parallelism, and regular spacing. The slope of a linear dune is relatively small, usually similar on both sides, with the upper crestal area with more active sand movement [14]. As the wind changes its direction, both sides of linear dunes undergo erosion and deposition with respect to the current wind direction, as shown in Figure 2.7.

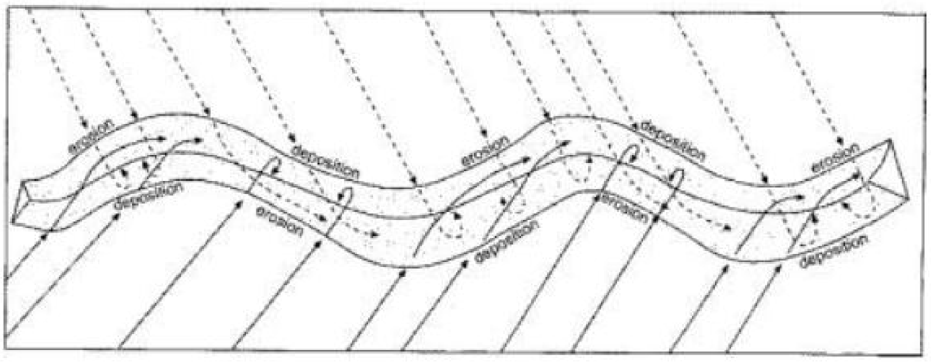


Figure 2.7: Schematic sketch of a linear seif dune that is under a bidirectional wind regime. One wind direction is shown by the solid line and the other by the dashed line (from [24]).

Star dunes: Star dunes are the largest dunes, which height may reach more than 300m. They contain a great volume of sand in comparison with other dune types [14]. Star dunes are created in areas of high wind direction variability. The shape of a star dune is mostly pyramidal with a central peak and commonly at 3 or 4 'arms' radiating from the central peak. Each arm has a sharp crest, resulting from changing wind direction. Usually, the 'arms' vary in sizes depending on the preferred orientation. Slope is greater along the crest, with an angle of 15-30°, whereas the bottom part has a broad and gentle slope of around 5-10°.

2.1.4 Dunes and Vegetation

Previously discussed dunes are typical for areas without rainfall, but some areas have enough rainfall and humidity to grow vegetation on the sandy surface. The vegetation is mainly limited by human impact or naturally limited by wind power. Vegetated surfaces cause steeper velocity gradients, which results in more significant shear stress, unlike unvegetated surfaces – vegetation causes greater friction than the unvegetated surface. Nonetheless, the more significant stress does not usually affect sand entrainment since it is not propagated from the plant to the ground itself. Interestingly, experiments proved that dunes formed by isolated plants are similar to those formed due to obstacles. Full vegetation cover prevents aeolian entrainment entirely, but partial vegetation cannot. There are mainly three dune types being formed in vegetated areas according to H. Tsoar [24]:

Nebkhas: Nebkhas, or coppice dunes, are isolated clumps of vegetation that act as sand traps. They can reach greater heights, up to 30m. The shape of a coppice dune depends on the vegetation coverage on the dune. Furthermore, nebkhas are considered static bedforms because they are usually changed as the vegetation changes only.

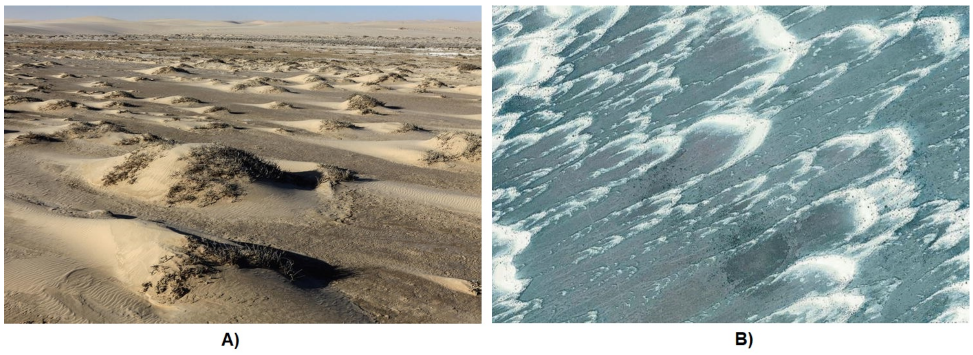


Figure 2.8: Sand dunes in vegetated areas. A) Nebkhas associated with plants in the coastal strip south of Walvis Bay, Namib Desert, Namibia (from [10]). B) Parabolic dunes in White Sands National Park, Tularosa Basin, Alamogordo, New Mexico. Wind direction is towards the upper-right corner (from [10]).

Parabolic dunes: Parabolic dunes are formed in humid or cold areas as well as in arid and semiarid areas with vegetation. The shape is parabolic, similar to the barchan dune, but the 'arms' are pointing upwind. They are usually present in coastal areas, where the vegetation is more easily established at the dune base. Vegetation in parabolic dune formation protects the less mobile 'arms' against the wind, which allows the central, wind-covered part to advance downwind, which eventually leaves the ridges behind and forms a U-shaped, parabolic dune. Some geologists see the parabolic dune as a further development of a spot blowout (wind-excavated sandy area).

Vegetated-linear Dunes: Those dunes are relatively lower, with rounded profiles. The vegetation covers them, sometimes entirely, sometimes abundantly on the lower slopes, but the vegetation is sparse or absent at the crest. When the vegetation covers the entire dune, the dune tends to be stabilized. Vegetated-linear dunes are usually parallel and can reach up to kilometers in width. A typical feature considering this dune type is the tendency of two adjacent dunes to co coverage and continue as a single ridge. This connection of dunes usually happens when the dunes form a Y-junction, which is typical for vegetated-linear dunes and is one of the distinguishing features between linear seif dunes and vegetated-linear dunes, together with the vegetation presence and straight shape.

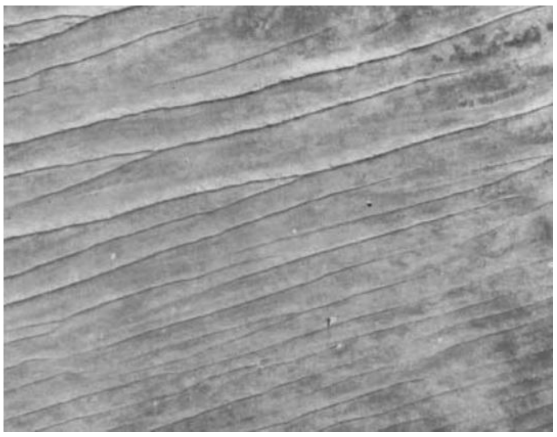


Figure 2.9: Aerial photograph of vegetated linear dunes. These linear dunes elongate in the direction of the dominant, strong wind. The dominant wind blows from left to right (from [24], modified).

2.2 Snow

According to Doesken et al. [5], the origin of snow formation lies in subfreezing clouds, where water exists in all three forms, liquid, solid, and vapor. Clouds are formed by water vapor transported in the atmosphere. When it comes to precipitation, only a fraction of the water contained in clouds is available for precipitation, and an even smaller fraction of precipitation consists of snow crystals. There are multiple snow crystal types, and each type is formed under different climate conditions. The main factors affecting the crystal are humidity, temperature, and wind. There may be even more crystal types created within one cloud, but most crystals tend to have hexagonal symmetry because of atom molecular arrangement in the crystal lattice. As the crystals are formed, they gain mass and size and start to fall downwards. The falling speed depends on the crystal properties and wind speed. Since snowfall does not directly influence our topic, we will focus on some fundamental snow properties and snow transport.

The humidity and temperature are crucial while observing fresh snow characteristics. Since snow crystals are subject to constant changes due to above-freezing temperatures, crystals tend to change their state even while falling to the ground. The most important snow property is snow density, which can be defined as a mass of water or ice per unit volume. Figure 2.10 shows different densities per unit volume. Snow density is constantly changing. As Doesken et al. [5] suggest, water density, considering the density definition above, is 1, ice is 0.917, and the upper density of fallen snow is around 0.40, while the lower density limit is near 0.01. Interestingly, fresh snow density does not correlate with location latitude or elevation. In Figure 2.11 we observe that most new densities are in range 0.04 to 0.10 despite altitude difference. Obviously, the lower the density, the higher the ability of snow to be transported by wind.

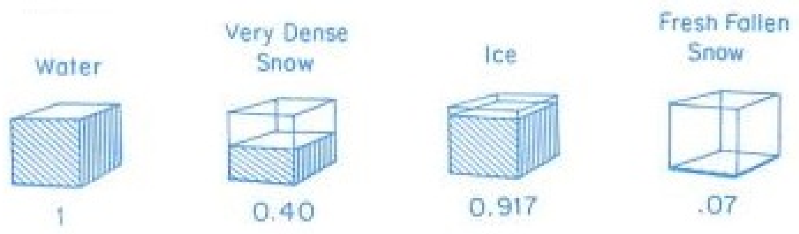


Figure 2.10: Relationship between the density of water states. The shaded parts show the volume that would be occupied by liquid water in case the cube content melted (from [5], modified).

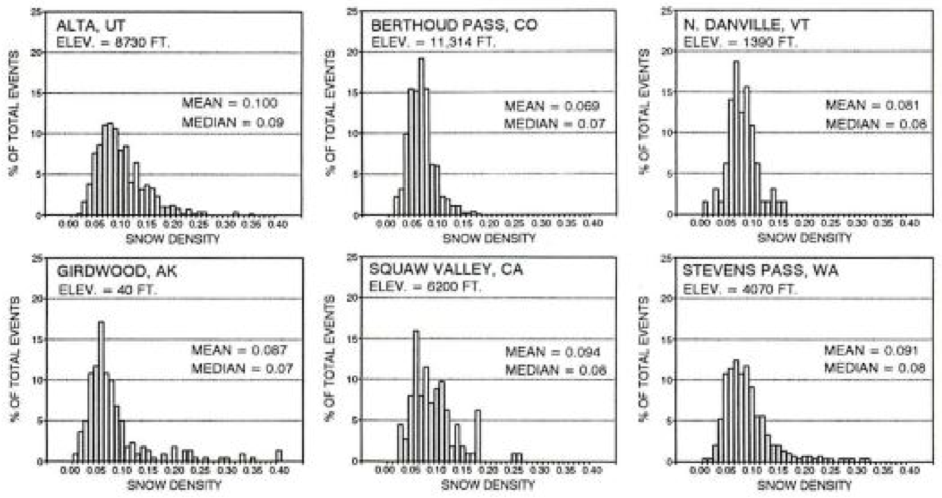


Figure 2.11: The distribution of density of 24-hour new snowfall totals at various U.S. locations (from [5]).

2.2.1 Snow Transport

Similarly to sand transport, it is still unclear how snow properties are responsible for snow bedform shapes and how the different shapes relate to each other. As stated by S. Filhol and M. Sturm [8], snow bedforms have analogs in sandy deserts or river bottoms, including dunes and ripples, whereas some features, such as snow pits, are snow specific. The process of snow transport is also almost identical; snow particles also undergo saltation, suspension, and creep. This observation of similarities is surprising if we consider snow particle density of $917 kg/m^3$ versus $2650 kg/m^3$ for sand. However, the critical difference is the ability of snow particles to bond to each other. This process, called sintering, significantly influences the snow bedform morphology. Sintering is absent in sand morphology because sand grains do not bond to each other in this way. Moreover, snow-pack is formed from snow precipitation under different conditions. Therefore the individual layers tend to have different properties. Snow also moves due to avalanches, which even more changes the snow-pack structure compared to sand avalanches.

2.2.2 Snow Dunes

Strong parallels between snow and sand bedforms have been recognized, and it was concluded that these forms develop through broadly similar transport processes, but the forms differ significantly in size, geometry, age, and particle properties [8]. Sand dunes can reach a hundred meters high, and their formation process might exceed centuries, whereas snow dunes are usually formed within hours, rarely last more than a few months, and hardly get higher than a meter.

There are seven typical snow bedforms, shown in Figure 2.12, that are widely recognized: transverse dunes, barchans, whalebacks, ripple marks, crag and tails, pits, and a broad group of erosional features called sastrugi. Since the shapes, properties, and conditions for some dune types formation are similar, we will not mention them explicitly. Also, we will be mainly focused on bedforms common for both sand and snow.

Snow barchans are usually $3 - 20$m long, between $5 - 12$m wide, but only $0.1 - 0.5$m tall, whereas sand barchans can be 10m high, 100m wide, and 150m long. The windward side of a barchan also often develops erosion features resembling flutes and scallops, whereas the windward side of sand barchans is usually smooth. Ripple marks look similar to sand ripples and are formed by

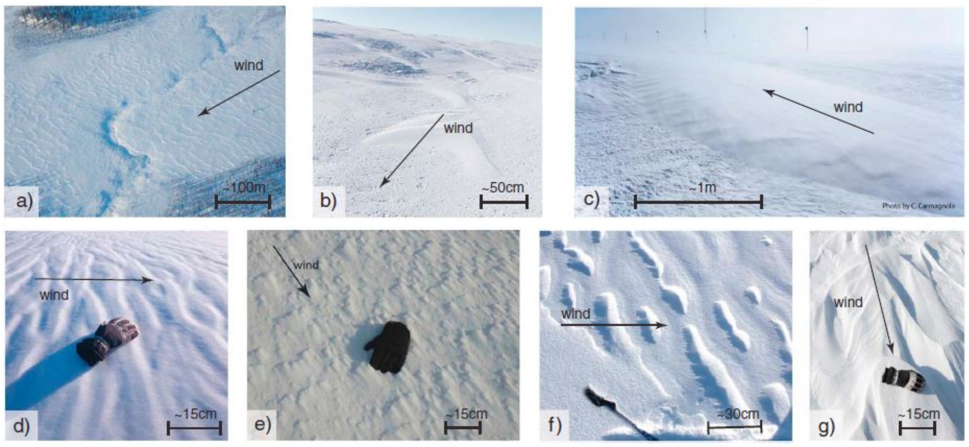


Figure 2.12: Snow bedforms classification. (a) Snow waves (transverse dunes), (b) barchan dunes surrounded by small pits, (c) a whaleback dune, (d) ripple marks, (e) crag and tail, (f) pits, with the handle of a ski pole for scale, and (g) sastrugi, with a glove for scale (from [8]).

saltation and creep. The difference is that creeping sand particles are located towards the ripple crest, whereas saltating particles tend to accumulate in the troughs.

2.3 Geological Principles Conclusion

In this Chapter, we discussed the main properties of both sand and snow. We focused on the properties of transport of those materials and analyzed common structures resulting from their transport. A brief description of the differences between snow and sand has been provided. With this knowledge background, we can consult the related work considering sand simulations in the upcoming Chapter 3 Related Work. For more detailed geological principles explanations, we recommend the reader to consult sand-related literature in [14], [24], [28], [12] or [10]. Snow-related literature can be consulted in [5] or [8].

Chapter 3

Related work

Computer graphics simulations can be divided into two groups. The first one copes with plausible results, usually at the expense of realistic phenomenon behavior, aiming at real-time visualization. The second approach is physically based. Such simulation is computationally demanding and usually uses more complicated formulas to describe the simulated phenomenons. Physically-based simulations tend to run offline and are more complicated to implement correctly, especially since setting the simulation parameters proved to be a challenge. We will discuss various approaches to sand and snow simulation in this Chapter and review their advantages, disadvantages, and other valuable insights. We will provide a more detailed explanation of individual implemented simulation algorithms and properties in the following Chapters 4 and 5.

3.1 The Base Model

One of the first sand simulation models was created by B. T. Werner [26]. In their model, dunes are formed by moving sand slabs, which stack on each other on a square lattice with a non-erodible substance representing the ground. They consider the number of slabs on a lattice grid cell as the elevation at this cell. Sand slabs are transported individually to roughly approximate the wind transport of sand grains (i.e., saltation). The slab is moved a specified number of lattice sites, L, in the transport direction and is deposited at the destination with a probability that depends upon sand slabs count at this location; therefore, it can bounce and be carried further. If it exceeds the lattice dimension, periodic boundary conditions are applied on this slab. As slabs deposit, height differences between neighboring lattice grid cell stacks occur, which is not acceptable due to the material stability. Therefore, an angle of repose is introduced and set to 30° to ensure that neighboring cells maintain the angle of repose. When the sand slabs reach a shadowed cell, the slab will be forced to deposit here. Shadowed cells are detected by setting a shadowed area with an angle equal to 15° from the highest peak located against the wind. The principle is visualized in Figure 3.1. The application simulates different types of dunes, such as barchans, linear dunes, and star dunes, shown in Figure 3.2.

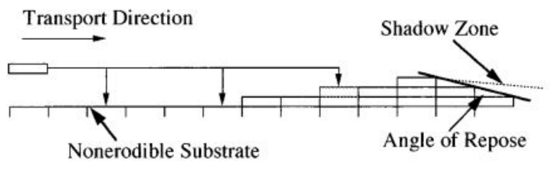


Figure 3.1: Side view illustrating dune-simulation transport algorithm (from [26]).

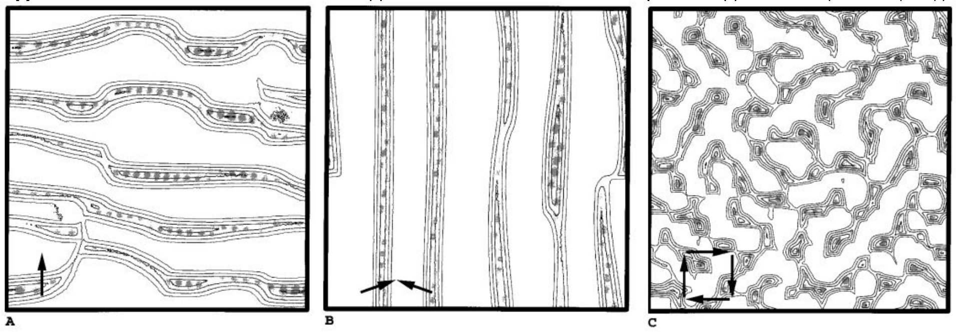


Figure 3.2: B. T. Werner's implementation results. A) Barchans, B) Linear dunes, C) Star dunes. The arrow indicates the simulated wind direction in the lattice throughout the simulation (from [26]).

The principle of their algorithm is rather simple while generating relatively plausible results. Other simulation algorithms use the idea presented by B. T. Werner further. However, this presented approach has some disadvantages. Firstly, the sand dunes might not be initiated, especially while setting different values for the wind speed. Secondly, the rules of material deposition according to the repose angle might be too simplified. Also, there is a problem with the algorithm functionality when the wind changes its direction so that it is not perpendicular to the square lattice. Then the transported slab would need to choose between multiple slabs while being transported by the wind above the grid, which would not be possible at all when $L = 1$. The question is whether this in-determinism is acceptable because such a situation does not occur while the wind is perpendicular to the lattice. The algorithm does not consider differently shaped non-erodible surfaces, such as hills or other phenomena. Finally, there is ambiguity in the time domain since the slab transport and the stabilization according to the angle of repose is performed per time step, whereas in reality, the sand avalanches occur faster, and a part of the sand mass is moved at the same time.

3.2 Modification to the Base Model

Momiji et al. [18] revised Werner's implementation and added two main improvements. The first revision incorporates more realistic wind behavior into the simulation, and the second considers the wind speedup and the equilibrium dune shape.

The base of the algorithm is similar to Werner's approach. The angle of repose of 33.7° is defined here instead of 30°. They removed the sand slabs' ability to erode on a cell in the dune's lee since the wind cannot cause erosion in the shadowed area. The shadow angle that defines the shadow zone slope is set to 15°. Momiji also reports that removing the ability of erosion in the shadow zone makes the slip face in the lee sustained, the dunes grow more quickly due to a substantial accumulation of sand slabs, and the dunes migrate downwind with constant speed.

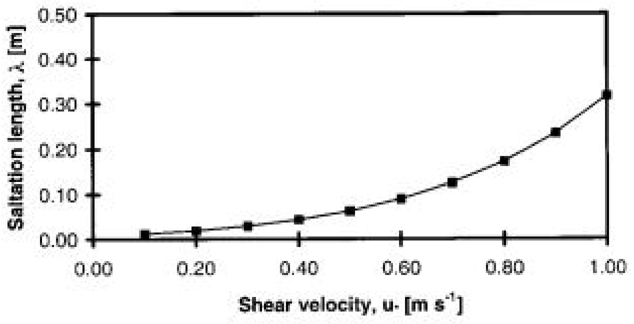


Figure 3.3: Relation between saltation length (λ) and shear velocity (u*) suggest non-linear relation between them (from [18]).

The wind speedup has not been considered in Werner's implementation, but it has been proved to be essential for dune transformations. They state that introducing wind speedup makes dunes lower and migrate more quickly. It also makes the dune windward slopes gentler, producing asymmetric dunes. Momiji uses a kinematic formulation for wind speedup, which counts with wind speed and the wind shear velocity or equivalently wind shear stress which causes the sand transport. They implemented both linear and non-linear wind speedup to achieve this behavior since the saltation length, which corresponds to the slab transport length in their model, non-linearly increases as shear velocity increases (Figure 3.3). Momiji reports that linear wind speedup controls windward dune slopes, whereas the non-linear increase of the transport length lowers dunes height and potentially increases the number of dunes. However, choosing better values for the shear velocity coefficients, both linear and non-linear, may result in better lee-side dynamics, as stated by Momiji.

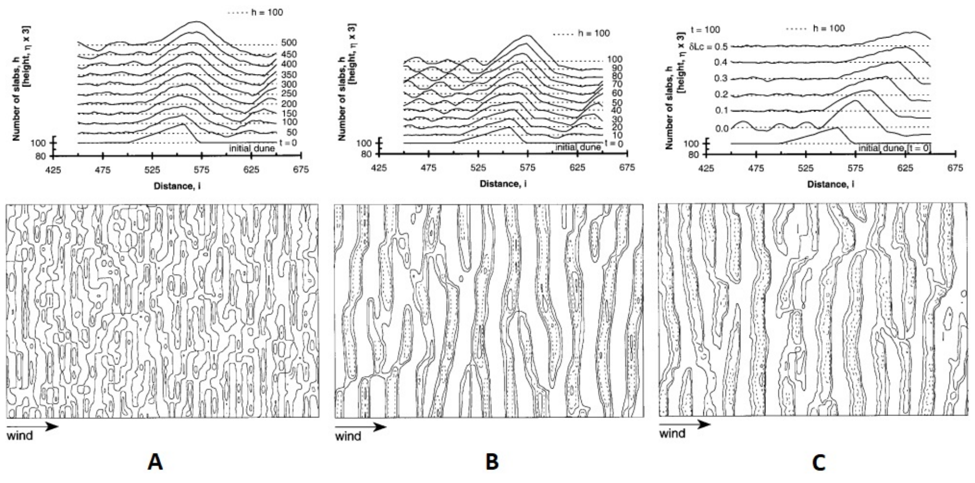


Figure 3.4: Comparison of Momiji et al. implementation improvements with Werner's implementation. A) Werner's implementation, B) Momiji's implementation with removed erosion from shadow zones, C) Momiji's implementation with added wind speedup and shear velocity (from [18], modified).

3.3 Modifying the Wind Speedup

S. R. Bishop et al. [1] further adjusted the wind speed-up implemented in previously discussed Momiji's approach. Momiji et al. defined the wind speed-up based on a constant representing the average number of slabs per cell. This value is assigned after the simulation grid is generated and remains the same throughout the simulation. Bishop et al. introduced a modified value for the wind speed-up computation. This value is dynamically adjusted according to the number of slabs at a given cell at each time step, and the results indicate a more realistic dune appearance, especially on barchan dunes.

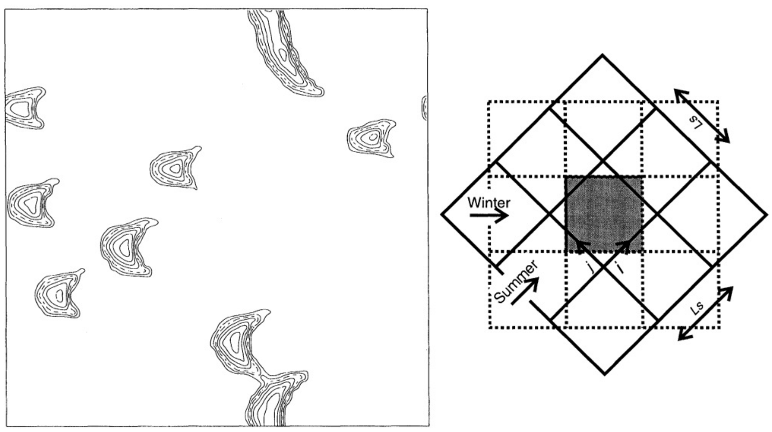


Figure 3.5: Bishop et al. simulation results. Left: Simulated barchans with gentler windward slope and steep leeward slope, Right: The lattice is rotated in response to the wind directional change (from [1], modified).

They also added a solution to apply wind in any direction. The wind essentially blows according to the same axis; it is the simulation grid that rotates by an angle, as shown in Figure 3.5. This way, the topography remains unchanged, and the angle of repose is checked after the rotation. However, the number of slabs is not conserved. If the average number of slabs is small, the situation is even worse, where as much as 10 % difference between the initial and the final number of slabs might occur. Bishop states that choosing an optimal pivot for the grid rotation might lower the sand slab reduction.

3.4 Vegetation Presence

Joanna M. Nield and Andreas C. W. Baas [19] focused on studying the impact of vegetation in the area of sand dunes simulation as well as new observations considering numerical slab-based models.

They used the original Werner's model (described in Section 3.1) to represent the simulation grid. They state that using other values of L, which represents the wind speed in the model, more precisely, the number of slabs skipped by the transported slabs per bounce, does not contribute to dune patterns development. They further mention different strategies for lattice cell polling (or equivalently sand slab polling) and define the difference in both methods. Original Werner's polling method allows repeatedly polled slabs in one time step, where one time step equals $Dim \times Dim$ slab polling events. Nield et al. report that this strategy smooths the landscape, preventing dune formation. While using the strategy of non-repeated polling, i.e., the slab is polled precisely one time per time step, this artificial effect is eliminated. Furthermore, changing the wind direction has been examined by setting a 60° transport direction by moving one-third of all polled slabs in the 'longitudinal' direction and two-thirds of the polled slabs in the 'transverse' direction. Their results show that barchan dunes modeled using this method are elongated perpendicular to the wind and tend to have unequal 'horns.'

Vegetation increases the threshold shear velocity required to initiate and sustain transport, and therefore Nield et al. imitate this by changing the probabilities of sand slab erosion and deposition and adjusting the angle of repose on vegetated cells. These modifications can be modeled as a local 'vegetation effectiveness' function on each cell. Note that this approach does not reflect vegetation's physical shape whatsoever; rather, it expresses the cell's ability to influence sediment transport. Since vegetation grows and dies throughout the years, an annual update of vegetation effectiveness seems reasonable. In order to represent a variety of vegetation types, a cell may contain multiple values of vegetation effectiveness for each vegetation type, and the overall influence is expressed as a sum of those values.

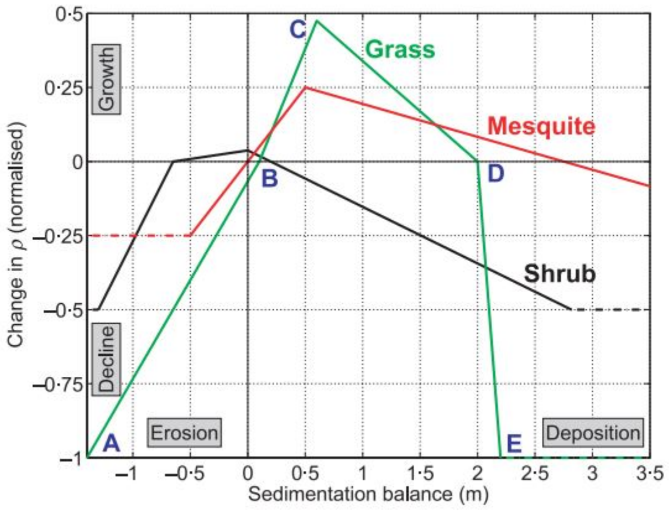


Figure 3.6: Growth functions for typical vegetation types used in model by Nield et al., representing annual growth response to deposition and erosion conditions (from [19]).

Nield et al. focused on implementing two phenomenons involving vegetation presence in a semi-arid environment: nebkha and parabolic dunes formed from blowouts. Both simulated scenes resemble their real-world correspondences. They also studied different initial conditions for the sand bedform formation. Results of their simulation are shown in Figure 3.7.

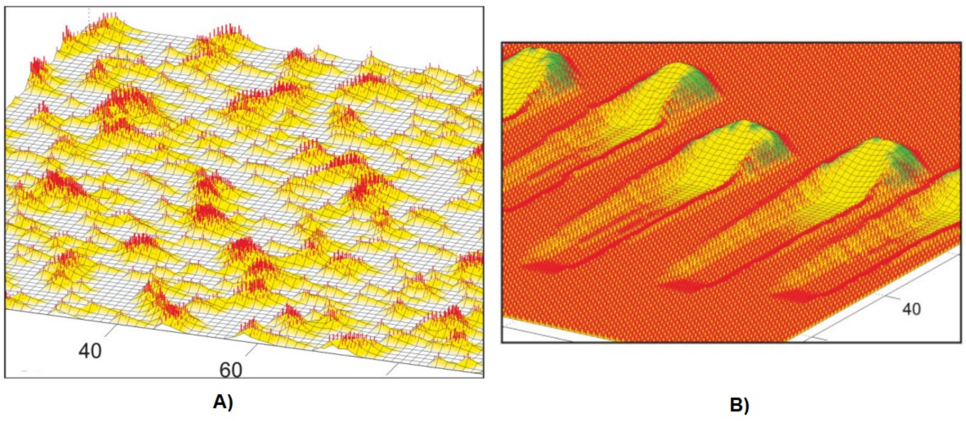


Figure 3.7: Nield et al. simulation results. A) Nebkha dune field, transport direction from upper left to lower right. Vegetation is represented by red bars. B) Parabolic dune development, vegetated surface with five identical 6-m-wide blowouts. Green shading indicates pioneer grass; the size and density of red bars indicate shrubs. Transport from lower left to upper right (from [19], modified).

3.5 Particle Based Simulation

The final sand simulation model that will be reviewed was published by Wang et al. [25]. Their approach is based on the same data representation, a uniform height map. However, it differs from other simulation models by moving particles instead of previously mentioned sand slabs and using a Eulerian form of ordinary differential equation of Newton's motion to compute the path of a moving particle instead of using probability-based sand material erosion or deposition. They also introduced a more complex wind field model, which, together with different particle motion computations, enabled the definition of any wind direction in the simulation grid. Vegetation was added to the simulation as a cylindrical shape. The wind field reacted to the vegetation, and on the leeward side of the cylinder, a wind shadow was formed. See Figure 3.8, where sand particles react accordingly to vegetation while forming the sand ripples. The main disadvantage of this model is the particle motion computation, which is slower than the probability-based movement of other simulation approaches. To counter that partially, they implemented their solution on GPU. Their model is used for sand ripple simulation with possible extensions to simulate transverse dunes since their geometry is similar to sand ripple geometry but on a larger scale. This combined simulation of both dune types was achieved by combining two simulation layers scaled differently together, with the effect resembling sand suspension, which is shown in Figure 3.9 below and Figure 1.2 at the beginning of this thesis.

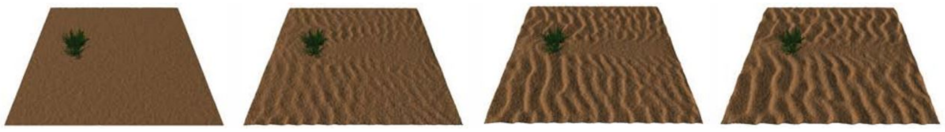


Figure 3.8: Particle based sand simulation with effect of wind shadowing behind vegetation. Captured after 100 s, 200 s and 400 s (from [25]).

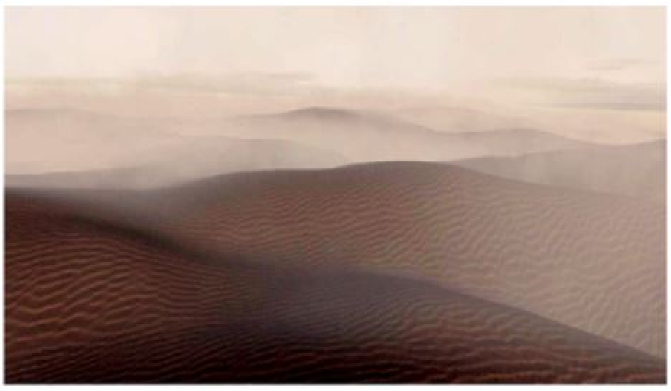


Figure 3.9: Sand dunes with the added effect of suspension (from [25]).

3.6 Snow Simulations

Due to the similarity of sand and snow particles, it seems possible to define the transported particle and other simulation parameters with different values, and one can observe the results, as Chopard et al. [2] showed in their model for material particle transport. They simulated both sand and snow transport and used it to simulate sand ripples. We will go through two snow simulation approaches, spanning most of our intended research areas.

Snow simulation consists of several steps. Typically snowfall is implemented, then snow needs to be stabilized as it reaches the ground and falls onto objects. Then a simulation of the thermodynamic process on the snow can be applied. The snow is also transported by wind, especially while falling to the ground, and avalanches occur when the snow becomes unstable. P. Fearing [7] in his thesis describes the simulation process of creating snow landscapes and provides efficient snow rendering insights. He uses a reverse approach for generating snow surfaces. Particles are emitted in the reverse direction, from the surface towards the sky, to determine if an object shadows a surface point. The surfaces of both ground or objects are further subdivided whenever a particle tracing indicates an interesting occlusion boundary or the surface is combined in case the area is wholly occluded or exposed. This computation generates a mass accumulation picture, making it possible to add an appropriate amount of snow to the scene. As soon as the snow is added, it is tested for stability. The snow stabilization process may affect the stability of both uphill and downhill neighbors, and therefore, it is implemented in several passes. The angle of repose is a crucial parameter for the stabilization process, and its value varies with the current snow type since it can range from near 90° in fresh dendritic snow to 15° in extreme slush conditions. A scene with added snowfall is shown in the following Figure 3.10.

E. Galin et al. [3] investigated the interactive generation of time-evolving, snow-covered landscapes with avalanches. They use a layered model to represent the surface, where the simulation grid can span from meters to kilometers. Each cell contains a stack of height values representing layers, where each layer represents either surface or several snow types, such as powdery snow or compacted snow. Their work aims to capture snow effects – snow cornices on the leeward side of mountain crests, snow buildup at the base of slopes scoured by avalanches, and also investigate ski-tracks simulation. They work with temperature influence in their simulation. The temperature depends primarily on altitude and sunlight exposure. As for sunlight exposure, they compute both direct and indirect. They also work with the effect of wind in the simulation resulting in snow transport. For avalanches, they define a rupture point at which the avalanche starts. If it is located in unstable snow,

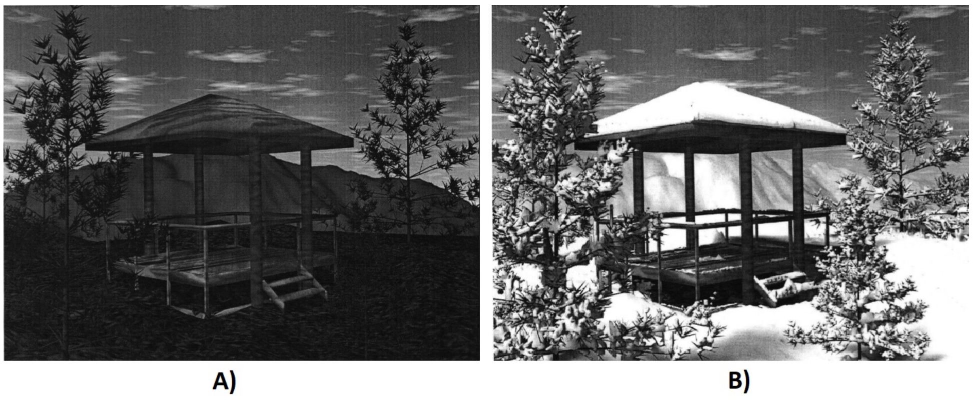


Figure 3.10: Showcase of P. Fearing snowfall and snow accumulation implementation. A) Initial scene, B) Scene with automatically added snowfall (from [7], modified).

the avalanche rapidly propagates to neighboring areas. All unstable snow in the rupture zone is immediately set into motion, as well as all unstable snow in the downslope reach of the avalanche. Some results of their work are shown in Figure 3.11.

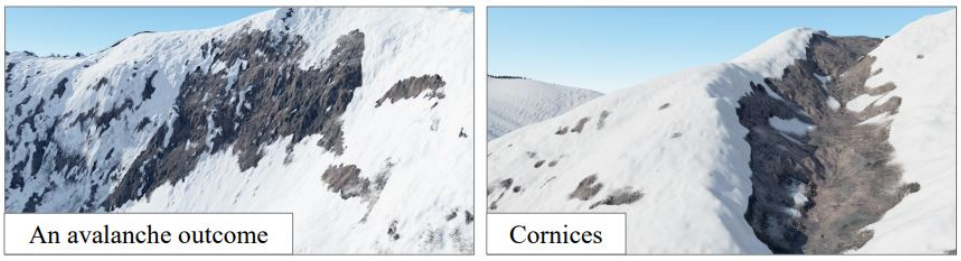


Figure 3.11: Showcase from Galin et al. snow forming application featuring some of their simulation results (from [3]).

Chapter 4

Solution Design and Analysis

There are numerous approaches suitable to implement an application with evolving dune field; nonetheless, they all have one thing in common – since the simulated phenomena are complex to understand, applications lack real-world realism, even with nowadays technologies. However, the plausibility of results is one of the critical properties of successful simulation in computer graphics. As mentioned earlier, all those combined processes are complicated so that it is currently not possible to implement them in a computationally acceptable way, and therefore, it is necessary to omit some features or implement them more straightforwardly. For example, implementing a wind field over the dune field is a difficult task, and many simulations try to simplify the wind behavior while maintaining the properties observed in the real world. Firstly we present various approaches to sand simulation and explain the thought process of designing our solution.

4.1 Simulation Approaches

Considering the data structure of simulated sand and the chosen approach to interact with it, there are more ways to approach the simulation. The first option is a numerical model, where the sand grains are generalized into sand slabs and moved as one unit. This representation is usually implemented on a uniform grid. Overview of this approach presented by B. T. Werner [26] and later extension of Werner's model by Momiji et al. [18], and Bishop et al. [1] was provided in 3.1, 3.2 and 3.3.

Another approach is based on cellular automaton (CA). There is a regular uniform lattice with discrete variables on each cell, and the values of variables specify the state of the automaton in each cell as it evolves in discrete time steps. Noritaka Endo [6] observed similar results as in Werner's simulation approach mentioned earlier using CA. CA is also useful for simulations of material movement, such as debris, as introduced by D. D'Ambrosio et al. [11], where they use a hexagonal grid (Figure 4.1). The hexagonal grid might be more suitable than the rectangular grid for material distribution since one cell has more neighboring cells, i.e., there is a better variability of directions where the material can be moved. The CA approach and numerical approach are sometimes considered to be equivalent.

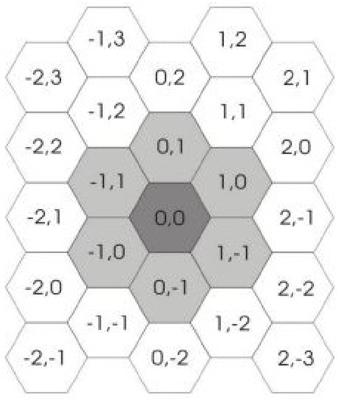


Figure 4.1: A portion of a 2-D-hexagonal cellular space. The central cell (0,0) and its six adjacent cells are marked in dark and light grey (from [11]).

As a third possibility, we may represent the sand grains as particles and use a particle-based simulation technique with computational fluid dynamics (CFD) to, for example, compute the wind field. CFD is beyond the scope of this thesis. We will instead refer to a particle-based simulation model by Wang et al. [25], which was discussed previously in 3.5. The idea is to use spherical grains of chosen diameter and iteratively compute their position within the grid based on forces applied to that particle. Since the number of sand grains limits all particle-based simulation approaches, a GPU acceleration is usually used to achieve better performance.

Those approaches have the potential to be combined. For instance, a hexagonal grid might also be helpful for higher wind direction variability in the case of a simulation with changing wind direction. For this case, an algorithm based on the principle of DDA algorithms may be used for moving sand slabs in any given wind direction.

4.2 Time Domain

Real-world simulation should consider the time domain with respect to the simulated phenomenon. It needs to be considered how fast dunes are formed and how fast they travel under certain conditions, and how those events will be measured. In all simulation approaches, the term time step t consists of the number of cell polls and sand slabs transporting initiations, equal to the number of grid cells. We refer to cell polls and sand slabs transporting initiations as a simulation step, i.e., one time step consists of $Dim \times Dim$ simulation steps. The connection with the realistic time domain might be made by specifying the physical dimension of a sand slab in correspondence with the time scale. By applying those parameters appropriately, one can achieve realistic sand flux, and dune migration speed [26]. Inspiration also might be a model by Nield et al. [19], where vegetation grows and dies in an annual cycle, which consists of several time steps t. For example, when they simulate 50 years of landscape evolution and define 80 time steps t per year, a total of 4000 time steps are performed.

In model by Wang et al. [25] they let the simulation run for a number of seconds and they observe the sand ripple formation. The time required to form sand ripples is therefore implementation dependent and we are unsure what the connection with realistic time domain is.

4.3 Simulation Model Analysis

The model by B. T. Werner [26] will be used as the base of the simulation. They introduced a model based on heightmap and sand slab movement, which impacted other simulation approaches; therefore, it enables adding other models based on this approach to the application easier. To add to previously mentioned principles in 3.1, they defined two probabilities for slab deposition on cell. The first one, denoted as p_s, represents the probability of sand slab deposition on a cell with at least one sand slab. This probability should be higher than the second variable, denoted p_{ns}, which represents the probability of sand slab deposition on a cell without any other slabs, essentially representing a bare surface. The slab will be deposited with probability set to 1 when the cell is shadowed. Sand slabs then move within the grid based on those conditions in the direction specified by the user and are stabilized to ensure the angle of repose within the grid is satisfied. Considering the grid scale, they used the cell width equal to 1 unit and the slab height equal to 1/3 unit, where one unit can be seen as one meter.

Momiji et al. [18] adds realism to Werner et al. model by removing the ability to erode slab in the shadowed area and by adding the wind speed-up across dunes because the wind speed increases as it goes over an obstacle. It is worth noting that they mention the effect of those modifications on transverse dunes only. Therefore, investigating the impact on other dune types, such as barchans and star dunes, is worth investigating. Nonetheless, removing the ability to slab erosion in shadowed areas should allow the dunes to be created sooner. The wind speed-up is added to the base wind speed based on the following equation:

$$L_{(i,j)} = \begin{cases} L_0 + C_1(h_{(i,j)} - h_{avg}) + C_2(h_{(i,j)} - h_{avg})^2 & (h_{(i,j)} \geq h_{avg}) \\ L_0 + C_1(h_{(i,j)} - h_{avg}) & (h_{(i,j)} < h_{avg}), \end{cases} \tag{4.1}$$

where $L_{(i,j)}$ denotes the wind speed at cell (i,j), L_0 denotes the base wind speed, C_1 denotes the shear velocity linear coefficient, C_2 the the shear velocity non linear coefficient, $h_{(i,j)}$ denotes the height at cell (i,j) and finally h_{avg} denotes the average number of slabs per cell in the simulation grid. C_1 influences the windward dune slope and should produce an asymmetric dune profile with a longer windward side, whereas the non-linear term C_2 controls dune height.

Bishop et al. [1] further modifies the wind speed-up and explains the impact on forming both barchans and transverse dunes. The modification is done by defining the reference number of slabs on each cell in a given time step t, denoted as $h_{ref}(t)$. This value replaces h_{avg} in Equation 4.1 and is defined as

$$h_{ref}(t) = h_{avg} - \frac{1}{2Dim^2} \sum_{(i,j)=(0,0)}^{(Dim-1,Dim-1)} |h_{(i,j;t)} - h_{avg}|, \tag{4.2}$$

where h_{avg} denotes the average number of slabs per cell in the simulation grid, Dim denotes the grid dimension and $h_{(i,j;t)}$ denotes the height at cell (i,j) at time t. The second term in this equation expresses the average deviation from h_{avg}. Let S denote the sum in Equation 4.2. The value of S can be updated without the need to iterate through the whole grid at each time step before the sand erosion or deposition process at cell (i,j), as M. Fousek [9] suggests:

For sand slab erosion at cell (i,j):

$$S = \begin{cases} S-1 & (h_{(i,j)} > h_{avg}) \\ S+1 & (h_{(i,j)} \leq h_{avg}). \end{cases} \tag{4.3}$$

For sand slab deposition at cell (i,j):

$$S = \begin{cases} S-1 & (h_{(i,j)} < h_{avg}) \\ S+1 & (h_{(i,j)} \geq h_{avg}). \end{cases} \tag{4.4}$$

Considering grid rotation to achieve variety in wind directions, this approach will not be implemented because the number of slabs is not conserved while rotating the grid, which might cause undesirable behavior.

Nield et al. model [19] adds vegetation to the model, which enables simulation of semi-arid environments with vegetation presence with other dune types. The vegetation coverage on a cell is expressed as a 'vegetation coefficient' ρ, which is used to adjust the erosion or deposition probability on a given cell as follows:

$$\begin{aligned} p_{e(veg)} &= p_{e(bare)} - \rho, \\ p_{d(veg)} &= p_{d(bare)} - \rho(1 - p_{d(bare)}), \end{aligned} \tag{4.5}$$

where $p_{e(veg)}$ is the probability of slab erosion affected by vegetation, $p_{e(bare)}$ is the base erosion probability without vegetation (set to 1), $p_{d(veg)})$ is the probability of slab deposition affected by vegetation and $p_{d(bare)}$ is the base deposition probability without vegetation (either p_s or p_{ns}). All probabilities are forced to be in $[0,1]$. However, the vegetation coefficient is allowed to exceed those bounds. $\rho > 1$ suggests that the vegetation has grown above the value required to shut down transport, and $\rho < 0$ represents nutrient depletion or a hydrologically deficient environment. When $\rho \geq 0.3$, the angle of repose is set to 40° allowing steeper slopes to form on vegetated areas. Normalized change in ρ on each cell is then obtained from grow functions shown in 3.6. Furthermore, three vegetation coefficients for three vegetation types were introduced – Grass, Mesquite, and Shrub. Their vegetation coefficient ranges are shown in Equation 4.6, the influence on a cell is based on the sum of their values:

$$\begin{aligned} \rho_{grass} &\in [0,1], \\ \rho_{mesquite} &\in [-1,3], \\ \rho_{shrub} &\in [-0.5,1.5]. \end{aligned} \tag{4.6}$$

They also proposed different strategies for cell polling, which are mentioned in the following Section 4.6. Another change they added was a more precise explanation of the time domain and different sand slab heights. Unlike previous simulation approaches, they defined the sand slab height as $1/10$ unit length instead of $1/3$ unit length. They also locked the wind speed L equal to 1 and stated that $L \geq 1$ does not contribute anything additional to the development of dune patterns.

The simulation approach by Wang et al. [25] is particle-based, on contrary to other models (all following equations taken from [25]). They focus on particles capable of saltation; therefore, they have chosen sand grain diameter $d = 15mm$. Sand particle at position $\boldsymbol{P} = (P_x, P_y, P_z)$ erodes and is lift-off to the air by initial lift-off velocity $\boldsymbol{V}$. They set the ascend initial velocity V_y as

$$V_y = Bu_*, \tag{4.7}$$

where B is an impact coefficient with a value in $[0.8, 2]$ and u_* is a friction velocity corresponding to wind speed. Grain is being lift-off in a random direction from 21° to near 90° with respect to the horizontal axis. Furthermore, spherical coordinate (ϕ, ψ) is used to express the lift-off angle, denoted as ϕ, and the wind direction denoted as ψ. Equation 4.8 shows the process of lift-off velocity $\boldsymbol{V} = (V_x, V_y, V_z)$ computation:

$$\begin{aligned}
&V_x = \|\boldsymbol{V}\| \cos\phi \cos\psi, \\
&V_y = Bu_* = \|\boldsymbol{V}\| \sin\phi, \\
&V_z = \|\boldsymbol{V}\| \cos\phi \sin\psi, \\
&\text{where:} \\
&\|\boldsymbol{V}\| = \sqrt{V_x^2 + V_y^2 + V_z^2}, \\
&0 \leq \phi \leq \pi/2, \\
&0 \leq \psi \leq 2\pi.
\end{aligned} \tag{4.8}$$

When the sand grain is lifted, several forces are applied to this grain in the real world, but the two most influential forces are gravitation force $\boldsymbol{F_g}$ and drag force $\boldsymbol{F_d}$, which can be computed as

$$\begin{aligned}
\boldsymbol{F_g} &= (0, -\frac{1}{6}\pi d^3 \rho_s g, 0), \\
&= (0, -mg, 0), \\
\boldsymbol{F_d} &= \frac{1}{8}\pi d^2 C_{ds} \rho_a \|\boldsymbol{\Delta V}\| \boldsymbol{\Delta V}, \\
&= A \|\boldsymbol{\Delta V}\| \boldsymbol{\Delta V},
\end{aligned} \tag{4.9}$$

where for $\boldsymbol{F_g}$, m denotes the sand particle mass and can be considered a constant (given the grain diameter d is constant), $\rho_s = 2655 kg \cdot m^{-3}$ denotes sand particle density and $g = 9.81 m \cdot s^{-2}$ denotes gravity acceleration. For $\boldsymbol{F_d}$, A denotes the sand particle drag in the air and can be considered a constant, $C_{ds} = 0.5$ denotes drag coefficient of the sand grain (value corresponds to a rough sphere), $\rho_a = 1.1644 kg \cdot m^{-3}$ denotes air density at 30°C and finally $\boldsymbol{\Delta V} = \boldsymbol{U} - \boldsymbol{V}$ denotes the relative velocity between wind and flying sand particle. The direction of $\boldsymbol{F_d}$ is the same as $\boldsymbol{\Delta V}$. Since the initial particle position $\boldsymbol{P}$ lift off velocity $\boldsymbol{V}$ is known, both $\boldsymbol{F_g}$ and $\boldsymbol{F_d}$ are known, sand grain position and velocity after a specified time step Δt can be computed iteratively by the Eulerian form of ordinary differential equation of Newton's motion:

$$\begin{aligned}
&\boldsymbol{V_{curr}} = \boldsymbol{V_{prev}} + \frac{\boldsymbol{F_d} + \boldsymbol{F_g}}{m}\Delta t, \\
&\boldsymbol{P_{curr}} = \boldsymbol{P_{prev}} + \boldsymbol{V_{curr}}\Delta t.
\end{aligned} \tag{4.10}$$

When a sand grain reaches the ground in the next time step Δt, it has hit the ground with impact velocity $\boldsymbol{V_1}$ and has outgoing velocity $\boldsymbol{V_2}$. Impact velocity is then divided into a normal component $\boldsymbol{V_{1n}}$ and a tangential component $\boldsymbol{V_{1t}}$. Moreover, two damping parameters are used to simulate the friction force f_{frac} on the tangential and the momentum attenuation f_{attenu} on the normal direction:

$$\begin{aligned}
&\boldsymbol{V_{1n}} = \boldsymbol{V_1} \cdot \boldsymbol{N} \times \boldsymbol{N}, \\
&\boldsymbol{V_{1t}} = \boldsymbol{V_1} - \boldsymbol{V_{1n}}, \\
&\boldsymbol{V_2} = -\boldsymbol{V_{1n}} \times f_{attenu} + \boldsymbol{V_{1t}} \times f_{frac},
\end{aligned} \tag{4.11}$$

where $\boldsymbol{N}$ denotes the terrain normal at the impact location, $\cdot$ denotes the dot product of two vectors, and $\times$ denotes vector – scalar multiplication. If the outgoing velocity $\boldsymbol{V_2}$ is less than a given threshold, it is set to zero to avoid infinite bouncing. When the sand particle is eroded or deposited, the cell containing the particle position changes its height $h(i, j)$ by δ. The simulation grid is stabilized similarly to previous simulation models.

Lastly, we will discuss the wind field model used in Wang et al. [25] simulation. They formalize logarithmic wind profile:

$$U(z) = \frac{u_*}{k} \ln\left(\frac{z - z_d}{z_0}\right) + \Phi, \tag{4.12}$$

where z denotes the height above the surface, $U(z)$ denotes the mean velocity at the height z, $k = 0.4$ denotes dimensionless Von Karman's constant, $\Phi = 0$ denotes thermal stability term, which is not taken into account by setting the value to zero. $z_d = 0$ denotes zero-plane displacement, ignored by their model, and finally, z_0 denotes the height at which wind speed reaches zero. They set $z_0 = d/30$ for surface without vegetation.

4.4 Initial Grid

The user should be able to define the grid dimension he requires, limited by memory availability only. Nonetheless, we capped the dimension at 4096×4096 cells because larger grids would take long to compute. Based on the simulation presented in the previous chapter, we do not expect a detailed grid with a higher dimension. We instead expect a larger area with more dunes. Having a simulation of a detailed dune being formed would require an alternative approach to simulation since all simulation algorithms and parameters would need to be scaled appropriately. We should provide the possibility to choose the initial grid characteristics and observe the behavior of implemented models on different initial grid types – for example, investigating if any dunes will be formed when the initial grid already has dunes generated or is entirely flat.

Furthermore, researched models consider flat non-erodible material layer. They do not consider the impact of non-erodible material layer variability, or they do not mention they addressed this explicitly. The problem is in the height difference between neighboring cells, where there is a more significant non-erodible material difference. If this fact is not taken into consideration, a non-realistic behavior might occur, as shown in Figure 4.2. Here, if the wind speed (denoted as L) is represented as ten slab movements per one slab bounce, the slab will not even consider the height difference it needs to overcome as the wind carries it. This issue should be further examined and addressed by checking the slab's collision with the surface between bounces.

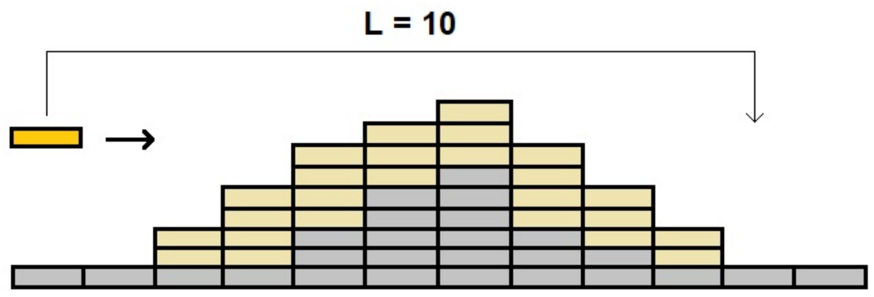


Figure 4.2: Slab is transported with wind speed equal to 10 slabs per 'hop'. The slab will completely ignore the small dune between the origin and the destination, even though it should hit the windward slope of the dune. Non-erodible material is grey, erodible material is yellow.

4.5 Wind Properties

The wind speed and direction have a crucial role in desertscapes formation. Identifying the simulation behavior under the variability of those parameters might yield crucial observations. Firstly we need to analyze possible approaches to wind properties representation.

The simulation approaches presented in related work usually simulate constant wind speed, only modified by some models according to the elevation. Modifying the wind speed throughout the simulation even further might be worth investigating. For example, star dunes tend to be formed under variable wind direction, speed, and various time windows for each wind direction, with one dominant direction. Those more advanced conditions were not considered in any model we found. In the Werner et al. approach, they defined equal time windows for each direction to simulate star dunes. Moreover, the problem is shown in Figure 4.2 also points out the problem of setting the wind speed too high, which will allow the transported slab to bypass the dune pattern formation mechanisms, as Nield et al. [19] noticed in their research. Computing the wind field as precisely as possible usually requires to use computational fluid dynamics (CFD) based on the Reynolds-averaged Navier-Stokes equations, as shown by Liu et al. [15], which is not suitable in our case. Therefore we will instead focus on a simpler approximation.

4.6 Slab Polling

According to mentioned simulation models based on slab movement, choosing a cell with slabs for erosion seems straightforward: choosing a suitable random cell within the grid, transporting it with respect to the wind direction and speed, and eventually depositing it later. We refer to a suitable cell for erosion as a cell that contains at least one slab and meets other slab erosion requirements. Those requirements might be different for each implemented model. For example, Werner's model requires at least one slab present at a cell only, the Momiji et al. model requires at least one slab at a given cell, which is not in the wind shadow, and Nield et al. modify the slab erosion probability based on vegetation present. However, there are more ways to choose a random cell. Considering the randomness of slab polling, one can use several random number generators available, but because this option of implementing various random number generators is not that significant, we will instead focus on different slab polling approaches.

As Nield et al. [19] mentioned, they found differences in dune formation while using repeated and non-repeated polling. While using repeated polling, one cell can be polled multiple times, and some cells could be omitted entirely. While using non-repeated polling, each cell is guaranteed to be polled exactly once per one time step. Non-repeated polling can be achieved by creating an array of indices with a size equal to the number of cells in the simulation and assigning a sequence of numbers from one through the number of cells to this array. Then during the simulation, it is sufficient to shuffle the indices prior to each simulation time step, go through this array one by one, and poll slabs according to an index, which is incremented after each poll. This way, we ensure every slab is polled exactly once per one time step.

There are also two ways to approach the polling process of sand slab erosion. One can choose one cell suitable for sand slab erosion, and when the polled cell is not suitable for erosion, the process of cell polling is repeated until a suitable cell is found, all within one attempt to initiate sand slab transport. Alternatively, when a cell is not suitable for erosion, one can abandon the sand slab transport attempt and continue to poll cells using random or non-random polling. The process of cell polling will be further discussed in the following Chapter 5.

4.7 Probability Aspect in Simulation

In discussed simulation models, one should have the opportunity to modify the probabilities crucial for sand slab behavior. Modifying those parameters will likely lead to different simulation behavior and results. Furthermore, alternating those parameters might correspond to different sand properties or weather conditions. As discussed in the previous Chapter in Nield et al. model [19], which introduced vegetation, by alternating those parameters they achieved to simulate vegetation presence in the simulation. This concept can be further applied to snow simulation as well.

4.8 Snow

Snow simulation in a scale presented by Chopard et al. [2], P. Fearing [7] or Galin et al. [3] is beyond the scope of this work. However, the base principle of sand behavior can be addressed by modifying the probabilities used for slab deposition or erosion. This way, one can achieve behavior corresponding to the snow packing process, where the snow layer becomes more stable and immune to wind erosion. The snow packing process also depends on temperature and snow density. A strategy for snow simulation based on previously introduced sand models should be based on dealing with snow density and the impact of temperature on the snow. We suggest omitting snowfall simulation and studying the impact of those parameters on freshly fallen snow.

The initial snow density can be obtained from the function describing the relationship between temperature and snowfall density shown in Figure 4.3, left. As Doesken et al. [5] explain, the density of freshly fallen snow is not strictly a function of temperature. Snow density generally decreases with temperature, but for temperatures below -5°C, the density increases again because sand crystals are generally small and pack in high-density layers as they reach the ground. After snowflakes reach the ground, they undergo metamorphism, which describes the melting process of snowflakes based on temperature, sublimation, and vapor pressure. This process can be expressed as shown in Figure 4.3, right, where functions describe the process of snow density increase over time at various locations.

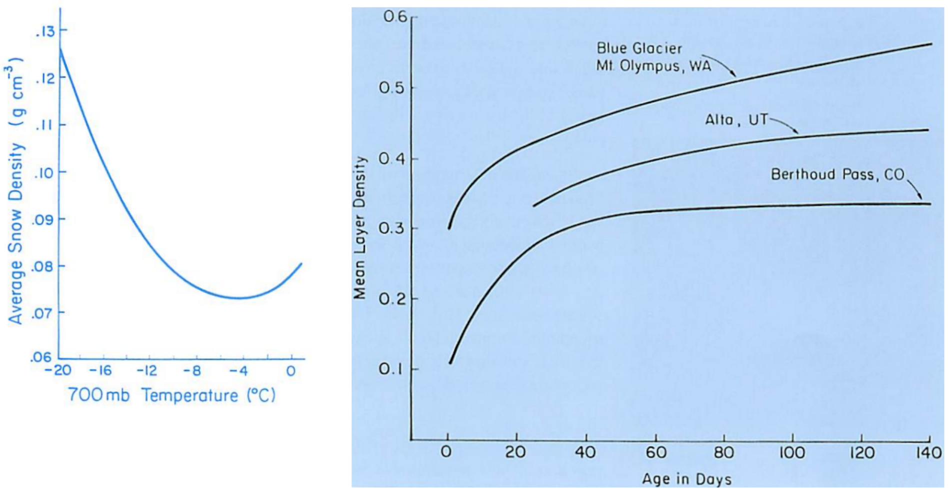


Figure 4.3: Impact of temperature on snow density and snowpack over time. Left: Relationship between temperature and snowfall density at High Altitude Observatory at Climax, Colorado, US (from [5]). Right: Changes of snowpack density over time in various US locations (from [5]).

4.9 Analysis Conclusion

In this chapter, we analyzed available simulation approaches, provided details on existing simulation models that we chose to implement, and went through the crucial properties of each model. Then we identified critical simulation parameters and suggested possible solutions and remarks. We also stated possible drawbacks, which should be further considered and possibly resolved.

Chapter 5

Implementation

Our simulation is written in C++ and visualized using OpenGL. We implemented the simulation model by B. T. Werner [26] described in 3.1 as the application base and then added other functionalities from discussed simulation models. In Figure 5.1, the simulation workflow is shown. This diagram describes the base model. We will go through the simulation loop and give detailed insights into the implemented procedures. Before we describe the diagram, we will discuss the data representation in Section 5.1 and the visualization in Section 5.2.

5.1 Data Representation

Similar to sand simulation models presented in related work, we use a regular uniform lattice to represent our simulation space, represented by a right-handed Cartesian coordinate system with a Y-axis pointing towards the sky. Each cell represents a stack (or column) of sand slabs, i.e., we do not consider individual slabs as objects accumulating on a cell. This way, we do not unnecessarily create a high number of slabs, which would be both memory and computationally demanding. Each column is still considered to be formed by sand slabs internally – the height of the stack is proportional to the number of stored sand slabs on this lattice cell. As a sand slab erodes at a cell, we only decrease the cell's stack height by δ, which usually corresponds to the slab height. For further extension, we define the height of non-erodible material at each cell. The non-erodible material is always under the erodible material.

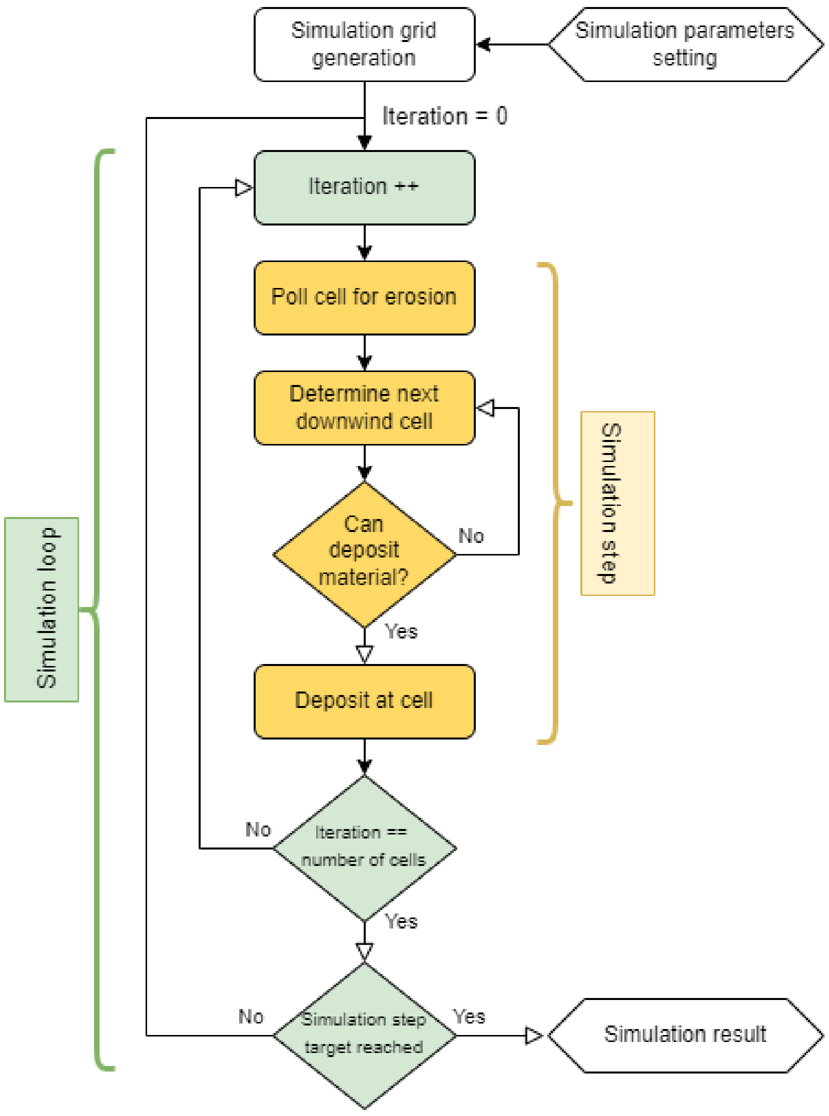


Figure 5.1: Workflow diagram of the application.

5.2 Visualization

We chose a suitable and straightforward data visualization for our case, a heightmap. The heightmap is visualized as uniformly shaped triangle strips (Figure 5.2). Each vertex has its (x, z) coordinates within the grid and its height y, which corresponds with the number of sand slabs on a cell plus the height of the non-erodible material, respectively. The y value expresses the elevation at a cell. Since the coordinates within the grid do not change during the simulation, we only need to update the height of vertices as the simulation runs. We apply color according to the chosen color palette based on the vertex's height.

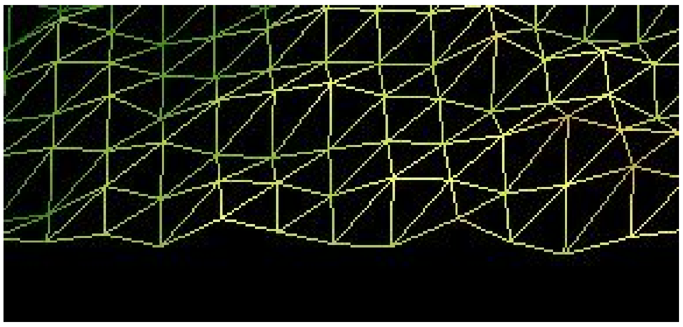


Figure 5.2: Detailed wireframe view of the simulation grid.

We chose two color palettes to represent the elevation in the simulation grid (Figure 5.3). The first palette is *RdYlGr*, commonly used for elevation visualization in most maps. The red color expresses the maximum stack height (or elevation) currently on the grid, and the green color expresses the minimum height. The color between the minimum and maximum height is interpolated. As the slabs are moved and the elevation changes, the maximum height is adjusted accordingly, which ensures there is always the maximal color difference. The second color palette is used for vegetation visualization. The color depends upon the vegetation coefficient value in a given cell. The dark green color expresses the vegetation presence, and the light green color expresses no vegetation. The user can choose between those color palettes while running a simulation based on Nield et al. model.

The simulation result can be exported into ParaView to show a more detailed view or other visualization techniques. This third-party visualization program allows to smooth the surface and show contours, which are essential for comparison and correct elevation perception of the exported scene. This modified scene can be further exported to Blender or any other modeling software, where additional modifications can be applied, such as textures or mesh decimation. This way, our results can be used, for example, as terrain for video games or similar purposes.

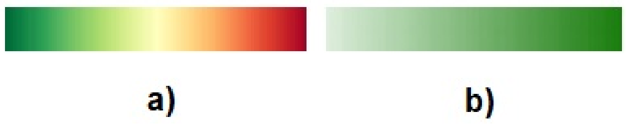


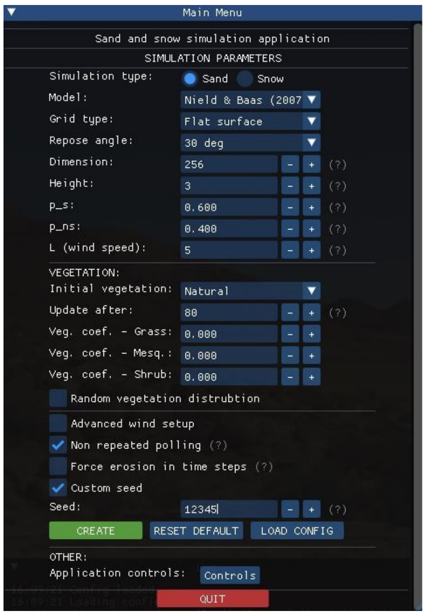
Figure 5.3: Used color palettes. a) *RdYlGr* color palette, b) Color palette used for vegetation presence visualization.

5.3 Simulation Parameters

The simulation requires various parameter settings in order to generate various dune types. Figure 5.4 shows the Main Menu window, where parameters are customized. Firstly user chooses between sand and snow simulation and then between simulation models. Based on the model selection, simulation model parameters are shown or hidden. In Figure 5.4 left, the Model by Nield et al. is chosen, which enables the modification of vegetation related parameters, such as the vegetation scene, annual vegetation update cycle, vegetation coefficients for each vegetation type, and the functionality to enable randomness of vegetation coefficient initial generation. Another standard simulation parameters are the grid type, where the user can choose between flat or randomly elevated surfaces or other possibilities such as transverse dunes or sand ripples.

An advanced wind regime setting will become accessible by ticking 'Advanced wind setup,' Figure 5.4 right. The user can add, remove, or move up or down a specific wind regime in the upper part. Each wind regime is defined by wind direction, initial and final wind speed, and the duration the wind will blow from a specified direction. Modification of the wind speed throughout this wind season is possible using the chart with the Beziér curve with interactable control points. The X-axis represents the normalized value of the wind regime duration; Y-axis represents normalized wind speed. By moving those control points by hold and drag, the user modifies the wind speed in this wind regime. Button 'RESET CP' will reset the control point's position. By ticking 'Show CP guidelines,' the control points will be visually connected by a line with the curve's beginning or ending point.

Furthermore, other parameters are the repose angle and the simulation grid dimension, which is restricted from 10×10 through 4096×4096 cells. The Height parameter defines generated number of slabs per cell. Probabilities p_s and $p_n s$ are restricted on interval $[0, 1]$ and are used to adjust the slab deposition behavior. Wind speed L can be set on interval $[1, dimension/2]$ to counter the problem of extra-long slab travels per iteration. Non-repeated polling ensures exactly one slab poll per cell in each time step, and forced erosion per time step ensures a cell suitable for sand slab erosion will be chosen per each simulation step. Ticking the checkbox 'Custom seed' enables the customization of a random number generator seed used for 'seeding' this generator. Any integer number can be submitted here. Once all simulation parameters are set or loaded from a configuration file, clicking on the 'CREATE' button launches the simulation, and the grid will be generated. Resetting default simulation parameters is possible after clicking on the button 'RESET DEFAULT'.


Main Menu

Sand and snow simulation application

SIMULATION PARAMETERS

Simulation type:	[● Sand] Snow
Model:	Nield & Baas (2007
Grid type:	Flat surface
Repose angle:	30 deg
Dimension:	256
Height:	3
p_s:	0.600
p_ns:	0.400
L (wind speed):	5

VEGETATION:

Initial vegetation:	Natural
Update after:	80
Veg. coef. - Grass:	0.000
Veg. coef. - Mesq.:	0.000
Veg. coef. - Shrub:	0.000

☐ Random vegetation distrubtion

☐ Advanced wind setup

✓ Non repeated polling (?)

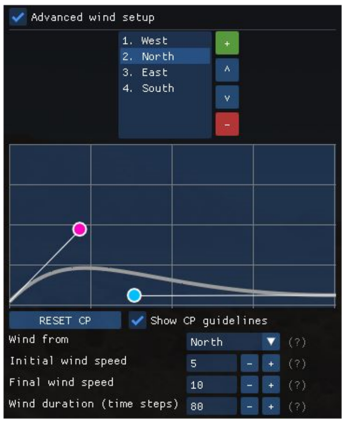
☐ Force erosion in time steps (?)

✓ Custom seed

Seed: 12345

OTHER:

Application controls: Controls


✓ Advanced wind setup

1. West
2. North
3. East
4. South

✓ Show CP guidelines

Wind from	North
Initial wind speed	5
Final wind speed	10
Wind duration (time steps)	80

Figure 5.4: Application UI for setting parameters. Left: Main Menu window with simulation parameters settings. Right: The user interface of advanced wind setup.

5.4 Simulation Loop

After the simulation parameters are defined and the simulation created, the user defines the desired number of time steps t, and the simulation loop can be started. Pseudocode of simulation loop is in Algorithm 1. In the pseudocode, $advancedWind$ is a boolean variable determining if the advanced wind option is enabled. L denotes the original value of wind speed chosen by the user in the simulation parameters setting. Function $BezierValue$ determines the current wind speed based on the y-coordinate value of the point on the Beziér curve, which x-axis coordinate is computed from the current time step i and current simulation step j.

5.5 Simulation Step

The pseudocode of one simulation step is shown in Algorithm 2 and Algorithm 3 respectively. We implemented those two approaches based on the previous discussion in 4.6. Nonetheless, each simulation step consists of three events, shown in the diagram 5.1: Slab erosion, slab transport, and slab deposition.

```
Function simulationLoop():
begin
    for $i$ = 0; $i < timeSteps$; ++$i$ do
        for $j$ = 0; $j < cellCount$; ++$j$ do
            if $advancedWind$ then
                $windSpeed \leftarrow$ {BezierValue((i * cellCount + j) /
                  (cellCount * timeSteps))};
            else
                $windSpeed \leftarrow L$
            end
            simulationStep($windSpeed$)
        end
    end
end
```

Algorithm 1: Pseudocode of simulation loop.

Slab erosion: Different cell polling strategies for slab erosion are why we present two pseudocodes for simulation steps. Slab erosion is the process of finding a suitable slab to erode. Requirements on the cell for erosion differ between each simulation model. Werner's model requires at least one slab on the cell, both Momiji's and Bishop's model additionally require the cell to be in the wind shadow, and Nield and Baas's model further modifies the probability of slab erosion based on the vegetation coefficient, shown previously in Equation 4.5. More detail on the slab erosion process is described in Section 5.6.

Slab transport: The slab is transported as long as the sand transport conditions are valid. Two conditions are required for Werner's, Momiji's, and Bishop's model to terminate the transport and deposit the slab. Firstly, deposition probabilities p_s and p_{ns} on the given cell influence the slab bounce probability, based on the cell's number of slabs. Secondly, the slab will be immediately deposited if the current slab location is in the wind shadow. Wind shadow detection is explained in 5.9. Nield and Baas's model further modifies the slab erosion and deposition by the value of vegetation coefficient on the given cell, as shown previously in Equation 4.5.

Slab deposition: Any condition for slab transport termination has to be fulfilled to end the slab transport and begin the slab deposition process. More detail on the slab erosion process is described in Section 5.7. Moreover, Nield's model also tracks the height change at each cell, which is used to compute the vegetation coefficient change after the annual cycle. Every slab erosion and deposition on the cell is tracked, and the deposition balance is updated accordingly.

```
Function simulationStepForceErosion(windSpeed):
begin
    while (Slab not eroded) do
        $CurrentCell \leftarrow$ {Randomly choose a lattice cell}
        if (CurrentCell.slabCount > 0) then
            StabilizeErodedSlab($CurrentCell$)
            break
        end
    end
    $shift \leftarrow \{windSpeed$ in the wind direction$\}$
    while Slab not deposited do
        $CurrentCell.position \leftarrow CurrentCell.position + shift$
        $generatedProb \leftarrow$ {Random probability on $[0, 1]$}
        $depositionProb \leftarrow CurrentCell.slabCount > 0\ ?\ p_s : p_{ns}$
        if (IsInShadow(CurrentCell) or (generatedProb < depositionProb)) then
            StabilizeDepositedSlab($CurrentCell$)
            break
        end
    end
end
```

Algorithm 2: Pseudocode of simulation step with erosion forcing per simulation step.

```
Function SimulationStep(windSpeed):
begin
    $CurrentCell \leftarrow$ {Randomly choose a lattice cell}
    if (CurrentCell.slabCount == 0) then
        return
    end
    StabilizeErodedSlab($CurrentCell$)
    $shift \leftarrow \{windSpeed$ in the wind direction$\}$
    while Slab not deposited do
        $CurrentCell.position \leftarrow CurrentCell.position + shift$
        $generatedProb \leftarrow$ {Random probability on $[0, 1]$}
        $depositionProb \leftarrow CurrentCell.slabCount > 0\ ?\ p_s : p_{ns}$
        if (IsInShadow(CurrentCell) or (generatedProb < depositionProb)) then
            StabilizeDepositedSlab($CurrentCell$)
            break
        end
    end
end
```

Algorithm 3: Pseudocode of simulation step without erosion forcing per simulation step.

5.6 Stabilization Process of Slab Erosion

Both stabilization processes work similarly. As for erosion, we lift the chosen slab and begin the slab transport process. However, we need to ensure the angle of repose is not violated on the cell where the slab eroded. Therefore we have to check if the angle of repose from the eroded cell's neighbors is violated, and if yes, we choose the steepest gradient, and from the cell where the steepest gradient was found, we continue the process of checking neighboring cells until we find a cell without the angle of repose violation. Here we remove the slab. The outcome is precisely the same as if we removed the slab from its original location and then moved a slab from a stabilized cell in the direction of the steepest gradient to take its place since the unstable slab in the location of the steepest gradient would be transported on the original place of sand slab erosion in any case. This approach can be justified because only one slab can erode simultaneously, and only one can move down the steepest gradient to take its place. The deterministic order in which the neighboring cells are processed is shown in Figure 5.5 B). The pseudocode of this process is shown in Algorithm 4. Note that it differs from Algorithm 5 only in the order in which the neighbors are processed, and the action performed on the stabilized slab.

```
Function StabilizeErodedSlab(Cell):
begin
    $CurrentCell \leftarrow Cell$
    $CurrCellIsNotStable \leftarrow true$
    while ($CurrCellIsNotStable$) do
        $NeighbouringCell \leftarrow null$
        for ($Neighbour : CurrentCell.neighbours.inOrder(erosion)$) do
            if (Repose angle violated and gradient is the steepest) then
                $NeighbouringCell \leftarrow Neighbour$
            end
        end
        if ($NeighbouringCell$ is null) then
            $CurrCellIsNotStable \leftarrow false$
        else
            $CurrentCell \leftarrow NeighbouringCell$
        end
    end
    $CurrentCell.slabCount -= 1$
end
```

Algorithm 4: Pseudocode of sand slab erosion stabilization process.

5.7 Stabilization Process of Slab Deposition

When the slab reaches its location where it should be deposited, we try to place it at its final destination and check the surrounding cells to see whether the angle of repose is violated. In case it is not violated, the slab is deposited. Otherwise, we choose the steepest gradient from neighboring stacks and move the slab down the steepest slope. This 'avalanche' process repeats until the slab reaches a cell, where it can be safely deposited without the repose angle violation. The order in which we process the angle of repose violation is shown in Figure 5.5 A). Further details considering the order of choosing the neighboring cells are described later in 5.8. The pseudocode of the stabilization process of sand slab erosion is shown in Algorithm 5.

```
Function StabilizeDepositedSlab(Cell):
begin
    $CurrentCell \leftarrow Cell$
    $CurrCellIsNotStable \leftarrow true$
    while ($CurrCellIsNotStable$) do
        $NeighbouringCell \leftarrow null$
        for ($Neighbour : CurrentCell.neighbours.inOrder(deposition)$)
         do
            if (Repose angle violated and gradient is the steepest) then
                $NeighbouringCell \leftarrow Neighbour$
            end
        end
        if ($NeighbouringCell$ is null) then
            $CurrCellIsNotStable \leftarrow false$
        else
            $CurrentCell \leftarrow NeighbouringCell$
        end
    end
    $CurrentCell.slabCount += 1$
end
```

Algorithm 5: Pseudocode of sand slab deposition stabilization process.

5.8 The Order of Choosing Neighbouring Cells

Here we shortly review the difference between the order of processing the neighbouring cells in the slab erosion and deposition stabilization process. The order matters only in case of equal gradients defining slope of neighbouring cells towards the tested cell, because the steepest gradient should be chosen in a deterministic manner. For the deposition process, we use the Von Neumann neighbourhood order shown in Figure 5.5 A), where we prefer the direction of wind to be processed first. For slab erosion process we use the order on the right, B), where the direction against the wind is processed first. This way the deposited slabs tend to move down the wind and eroded slabs tend to be replaced by those located against the wind. When changing the wind direction, the processing order is adjusted accordingly.

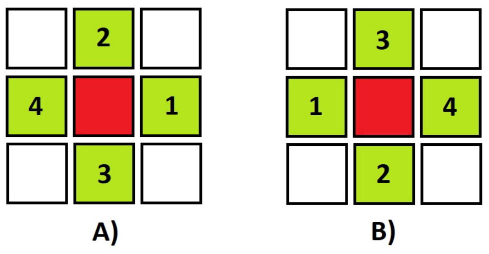


Figure 5.5: The order in which the neighbouring slabs are checked. Wind blows from left to right. A) Slab deposition order, B) Slab erosion order.

5.9 Wind Shadow

Wind shadow, in reality, is a complicated phenomenon. We took the idea of wind shadow computation from Werner's model. The wind shadow is simplified to obtain an area downwind from the highest peak under the angle of 15°. All cells in the area are shadowed. To determine if the cell is shadowed or not, we iterate several cells downwind and compute the shadow area. The idea is presented in Algorithm 6, where *UpperBound* is an experimentally obtained value based on the height difference between the maximum elevation in the grid and the elevation of the processed cell. If the procedure does not detect the shadow area on the given cell, the function returns *true*, *false* otherwise. *shadowZoneAngle* is tan of 15 degrees converted in radians.

```
Function IsInShadow(currCell):
begin
    for $dist = 0; dist < UpperBound; ++dist$ do
        $nextCell \leftarrow$ {Next Cell in direction against wind}
        $\Delta h \leftarrow$ {Height difference between $currCell$ and $nextCell$}
        if $\Delta h \,/\, dist > shadowZoneAngle$ then
            return true
        end
    end
    return false
end
```

Algorithm 6: Pseudocode of shadow area computation.

5.10 Dealing with Obstacles

In the previous chapter, we mentioned the problem when sand slab travels with higher wind speed and ignores the surface beneath. This problem extends when we consider a rock in the scene. Ignoring the fact that sand hits the windward side of the rock would result in unnatural behavior; however, none of the models based on sand slab movement explicitly discussed this problem. Here we introduce the approach to eliminate this artifact.

When the sand slab travels above cells, it should iteratively check whether it hit an obstacle along the way or not. Obstacles added to the scene do not contain erodible material. Obstacles can be added by setting the initial grid parameter to 'Three rocks' or 'Uneven surface.' For the 'Three rocks' scene, we added three rectangular-shaped areas to the simulation grid, and their height is dependent on the *Obstacleheight* parameter. Those areas will consist of the non-erodible substrate, which cannot be moved by erosion. To ensure the sand slab will react to the non-erodible surface, we implemented a function to detect this event, shown in Algorithm 7. The sand slab iteratively confirms that the angle of repose is not violated because of an obstacle made of non-erodible material. If the repose angle is violated, the slab will deposit on the cell, and the slab transport will be terminated. Otherwise, the slab will continue its travel to the target location specified by the wind speed. This approach ensures that the slabs will not get on top of the obstacle, completely ignoring it.

```
1  Function detectObstacle(currPos, targetPos):
2  begin
3      returns true when obstacle detected, false otherwise
4      $nextPos \leftarrow currPos$
5      while ($nextPos \neq targetPos$) do
6          $nextPos \leftarrow$ Next slab in the wind direction
7          if (Repose angle violated and nextPos has nonerodible
            material) then
8              depositeSlab($currPos$);
9              return $true$
10         end
11         $currPos \leftarrow nextPos$
12     end
13     return $false$
14 end
```

Algorithm 7: Pseudocode of obstacle detection.

5.11 Particle-based Model

Wang's model has a different approach than numerically-based models. We adjusted their implementation to fit our models. Instead of tracing many particles at once each time step, we trace individual particles from start to their final position. Each simulation step consists of tracing one particle. The number of traced particles is equal to the cell count. The particle is lifted off the ground by an initial velocity, and then the tracing begins. The tracing procedure in one simulation step is shown in Algorithm 8. Pseudocode functions, such as $generateParticleVelocity()$, are based on formulas presented in 4.3.

```
Function simulationStep(windSpeed):
begin
    $Vel_{prev} \leftarrow generateParticleVelocity()$
    $Pos_{prev} \leftarrow generateParticlePosition()$
    while (particle not deposited) do
        $Vel_{curr} \leftarrow updateParticleVelocity()$
        $Pos_{curr} \leftarrow updateParticlePosition()$
        if (is under surface) then
            $newVelocity \leftarrow computeParticleSpeedAfterBounce()$
            $Pos_{curr} \leftarrow$ Move particle above surface
            if ($newVelocity < thresholdVelocity$) then
                $depositParticle()$
                return
            end
            $Vel_{curr} \leftarrow newVelocity$
        end
        $Pos_{prev} \leftarrow Pos_{curr}$
        $Vel_{prev} \leftarrow Vel_{curr}$
    end
end
```

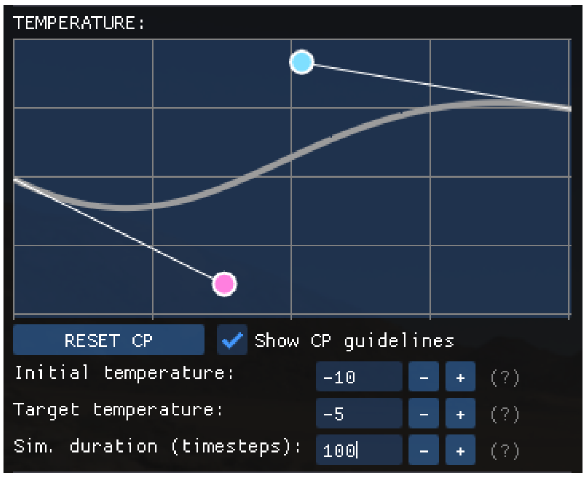
Algorithm 8: Pseudocode of particle tracing in one simulation step.

We simplified the implementation by setting the wind direction from the West only. We also adjusted the simulation scale, where the spacing between cells is 1 cm only, instead of 1 m for numerically-based models. The scale adjustment is because of the more realistic computation of sand grain movement by wind transport. Also, moving traced particles above the surface is necessary because particles will get under the surface to detect the ground-particle collision. The particle should continue to bounce when it is above the surface.

5.12 Snow Simulation

Since most of the focus of this thesis is given on sand simulation, the implementation of snow simulation in our application is experimental. We also implemented the snow simulation based on the model by B. T. Werner only. It uses the principles used for sand simulation methods, where discrete sand amounts are moved as sand slabs. In our implementation, snow is also moved in a probability-driven model, where snow slabs represent a transported material unit. However, to apply the snow properties to the previously introduced sand-based simulation model, we define new parameters.

Firstly, the initial snow density is determined based on initial temperature according to the function shown previously in Figure 4.3, left. The user then determines the temperature at the end of the simulation and adjusts the temperature development throughout the simulation. The simulation length is then based on those parameters – higher temperature causes the snow to pack faster. Moreover, the simulation length corresponds to individual time steps, corresponding to days in our simulation. The snow density is each time step (day) adjusted based on the lowest from shown functions shown previously in Figure 4.3, right. If the snow is packed before the simulation reaches the target step count, the simulation loop terminates since no other slabs can be moved. This state corresponds to the snow packing, which eventually causes the snow surface to become resistant to wind erosion. The user interface is shown in Figure 5.6. Temperatures can be adjusted from -20°C through 0°C, which is also the y-axis range of the curve setting window.



✓ Show CP guidelines
Initial temperature: -10
Target temperature: -5
Sim. duration (timesteps): 100

Figure 5.6: The user interface for temperature settings for snow simulation.

Throughout the simulation, we use the same principle of adjusting the erosion and deposition probability which was used in the model by Nield et al. [19], where they use a vegetation coefficient to adjust the probabilities. In this case, we use a coefficient representing the resistance of snow to be transported by wind. This coefficient is increased each time step according to the snow density development, which also depends on the current temperature. The snow density development is approximated by the lowest function shown previously in Figure 4.3, right. With higher temperatures, the coefficient is increased more. The current temperature at each time step is taken from the current value of the temperature curve in a given time step. This way, we approximately ensure that snow reacts to temperature. The density increase is computed as a multiplication of current snow density and temperature, scaled to interval [0,1] to correspond to the used temperature range. Pseudocode of snow density update is shown in Algorithm 9, where $func(t)$ represents obtaining the value of density change based on previously introduced density function.

```
1  Function updateSnowDensity():
2  begin
3      $currTempScaled \leftarrow getCurrentTemperature()$
4      $\Delta density \leftarrow currTempScaled \times func(t)$
5      for (all cells) do
6          $Cell.resistanceCoef \leftarrow Cell.resistanceCoef + \Delta density$
7          if ($Cell.resistanceCoef \geq 1$) then
8              $Cell.resistantToTransport \leftarrow true$
9          end
10     end
11 end
```

Algorithm 9: Pseudocode of snow resistance coefficient update in each time step. × is scalar – scalar multiplication.

5.13 Configuration Files

The user can create configuration files that hold the simulation parameters. The main reason to implement this functionality is to make setting specific simulation parameters faster and easier. After a simulation is set up and started, the user can save the simulation parameters of this specific simulation for later simulation re-run. When creating a configuration file, the user will be asked to specify where to save the file. When the user wishes to load a file, he will choose one from the system. Configuration files can be loaded when setting simulation parameters. After the file is loaded, simulation parameters will be set to values specified in the configuration file. The user might further open those files and modify any values here; however, setting invalid values might lead to file corruption. If the file is corrupted, the notification about which parameter was not loaded correctly will be shown in the Application Log window. Configuration files are in the *.json* format.

Chapter 6

Results

Before diving into the result presentation, we need to clarify some assumptions and conventions used in our simulation. Multiple simulation models have a variety of possible parameter combinations, and it is not possible to discuss all of them. We will present the most compelling results and results proving the implemented model's functionality. We always introduce the results of the given model and then review its functionality and behavior, possible with various parameters. Each model's results will be accompanied by a table of parameter settings for that simulation. Changed parameters in individual other results will always be mentioned. The result presentation will consist of the simulation result in the application view, accompanied by the view of contours on a smooth surface obtained after exporting to ParaView. All configuration files used for the result presentation are included as an attachment. Also, the comparison with real-world photographs of sand dunes with our simulation results will be analyzed. Lastly, we will review the results of the snow simulation.

6.1 Werner's model

The basic model should simulate barchans, transverse, linear, and star dunes. We confirmed the functionality of Werner's approach. According to our observation, non-repeated polling significantly impacts dune formation in Werner's model. The dunes are formed faster and have a more regular and sharper shape.

Figure 6.1 shows the results of the barchan simulation in Werner's model. The parameters are stated in Table 6.1. We observe the barchan-like structures being formed; however, the shape does not directly correspond to barchans found in nature. The most significant difference is the absence of a gentler windward slope, which makes the simulated barchans more 'moon-shaped.' This lack of realism is caused by not considering the wind speed-up on the windward slope, which is confirmed by the symmetrical windward and leeward dune slope and well visible in the contour visualization. We also observe not entirely flat surface between the dunes, which might suggest smaller sand bumps that are not entirely part of the dune morphology, where barchans are present. Nonetheless, we observe barchan 'horns' elongating in the wind direction. Throughout the simulation, we also observed that higher barchan dunes tend to have less visible 'horns.'

Parameter	**Value**
Grid type	Flat surface
Angle of Repose	30°
Dimension	256×256
Height	3
Slab Height	1/3
p_s	0.6
p_{ns}	0.4
Wind speed	5
Wind direction	West
Advanced wind setup	false
Non-repeated polling	false
Force erosion	false
Time steps	1000

Table 6.1: Parameters for barchan dunes simulation using Werner's model.

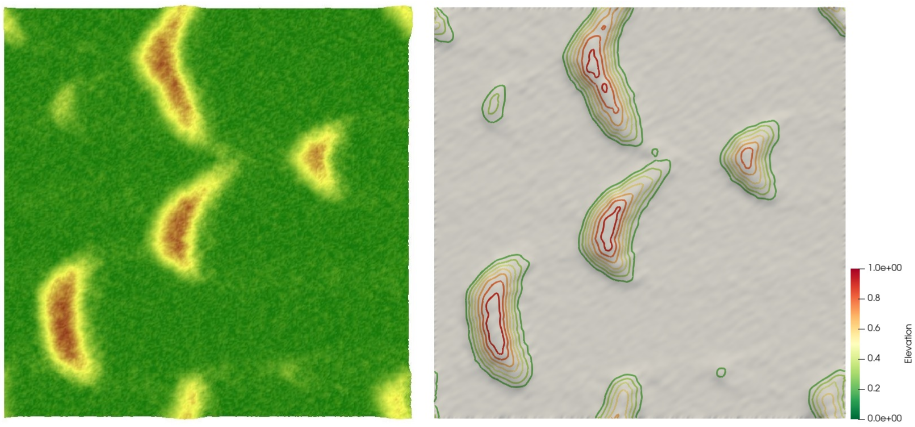


Figure 6.1: Barchan dunes simulated using Werner's model, maximum elevation reached 26 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Visualized contours in ParaView.

Transverse dunes in Werner's model more likely resemble a zoomed-in view of sand ripples (Figure 6.2). The regular shape is caused by the wind speed-up effect's absence on the windward dune slope, and the regularity is also visible thanks to the contour visualization in the Figure below on the right. We also see that the leeward dune side is not as steep as expected. Small bumps between dunes seem typical for Werner's model and are caused by the slab erosion in the shadowed area. Parameters to simulate this scene are identical to those presented in 6.1 except for the parameter Height. For transverse dunes, we set the parameter *Height* to 10 slabs per cell as the initial condition. The resulting height in the simulation result was 28 slabs per cell. We have simulated linear dunes as well with the same parameters as for transverse dunes, but the result looks very similar to the transverse dune result, even though we set two opposite wind directions altering after 100 times steps and run the simulation for a total of $t = 1000$ time steps.

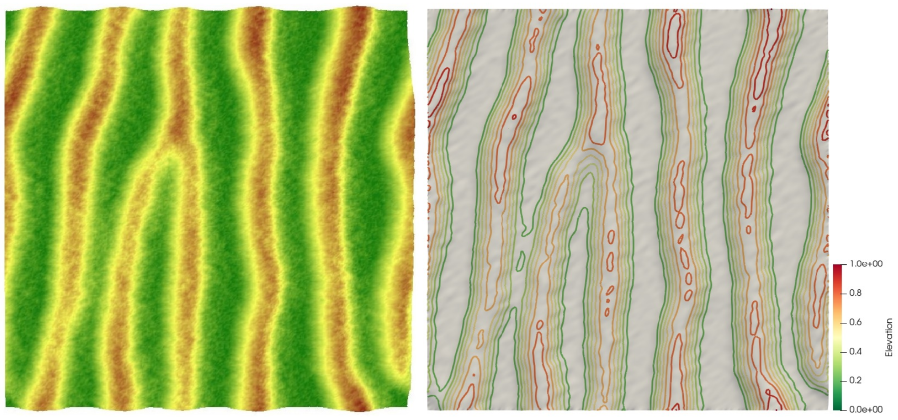


Figure 6.2: Transverse dunes simulated using Werner's model, maximum elevation reached 28 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Visualized contours in ParaView.

Star dunes form under varying wind directions. For star dunes simulation, we kept the same parameters for the simulation of transverse dunes except for the parameter *advancedwindsetup*. Here, we defined four wind directions in order: West, North, East, and South under constant wind speed $L = 5$. The wind blew 80 time steps per direction, and we performed a total of 640 time steps to achieve the result in Figure 6.3. In the result, one can observe some star-shaped dunes consisting of a central area and 'arm' resembling dunes emanating from the central peak, similar to those shown in 3.2. There are also dunes without a central area. We think those are formed as secondary dunes which were separated from a central peak or were not merged with other dunes yet. We used a grey-scale color palette and fewer contours to visualize the elevation to achieve better visibility in the contour visualization. Star dunes will be merged and spaced according to the wind direction and speed when we let the simulation run longer, but their shape will stop resembling star dunes. We assume that it is caused by constant wind speed and the periodicity of wind direction changes.

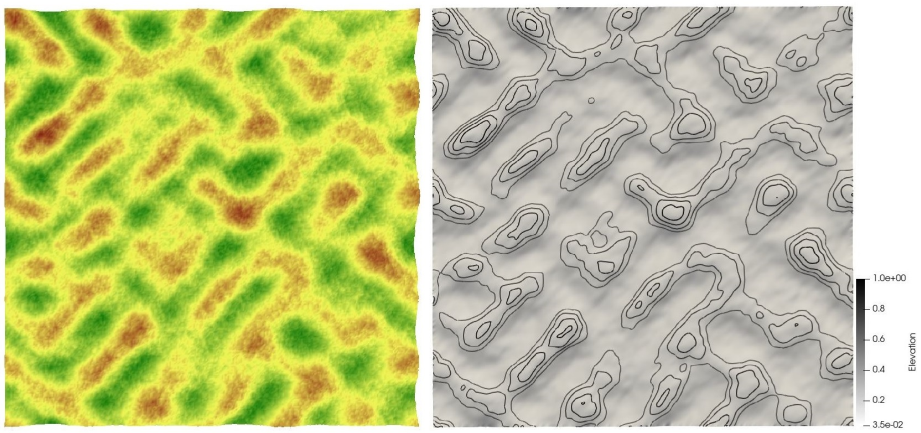


Figure 6.3: Star dunes simulated using Werner's model, maximum elevation reached 23 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Visualized contours in ParaView.

Next, we present the difference between repeated and non-repeated polling strategies applied to Werner's model for barchan simulation. Parameters for the simulation are the same as in 6.1 except the time step t, which was set to 200. In Figure 6.4 we observe a dramatic difference between the polling strategy. Dunes are formed faster and have sharper shapes using a non-repeated polling strategy. Barchans are sharper when we let the simulation with non-repeated polling run for $t = 1000$.

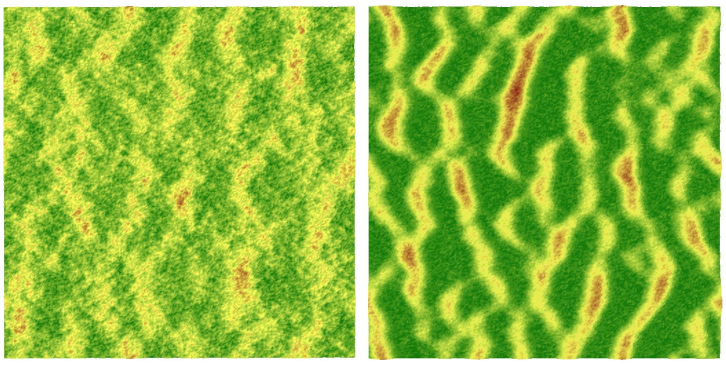


Figure 6.4: Difference between non-repeated and repeated polling after $t = 200$. Left: Repeated polling. Right: Non-repeated polling.

When we let the simulation run long enough, barchan dunes will merge and form one long dune. For transverse dunes, they will be merged and eventually reach an equilibrium state, where all dunes straighten, will be equally spaced, and will travel at a constant speed. This is caused by the wind speed being constant throughout the whole simulation. The amount of stabilized dunes depends on the amount of sand available for transport in the simulation grid, which is tightly connected with grid dimension and wind speed. This equilibrium state, in reality, does not occur since the wind speed varies in time and varies depending on the surface elevation.

6.2 Momiji's model

Momiji's model was primarily used to simulate transverse dunes, but we tested its functionality also by simulating barchans and star dunes. For the simulation of barchans in Figure 6.5, we used the parameters shown in 6.2. We observed that barchans are usually no longer as thin as in Werner's model, and the windward slope is usually gentler than the leeward slope. Also, we observed that the top parts of barchan dunes tend to be more flat, which is visible in the contour visualization. Star dunes look similar to Werner's model with more flat top areas – those results are not included here.

Parameter	Value
Grid type	Flat surface
Angle of Repose	33.7°
Dimension	256×256
Height	3
Slab Height	1/3
p_s	0.6
p_{ns}	0.4
Wind speed	5
Wind direction	West
Linear coef.	0.4
Quadratic coef.	0.002
Advanced wind setup	false
Non-repeated polling	false
Force erosion	false
Time steps	1000

Table 6.2: Parameters for barchan dunes simulation using Momiji's model.

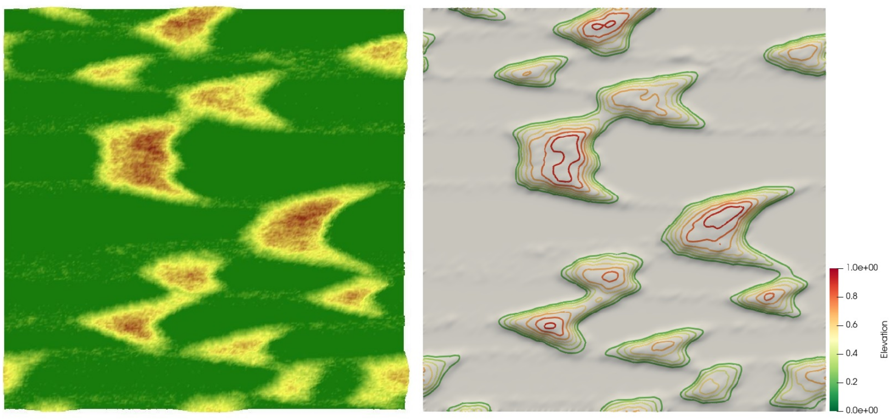


Figure 6.5: Barchan dunes simulated using Momiji's model, maximum elevation reached 26 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Visualized contours in ParaView.

Transverse dunes in Momiji's model have a gentle windward slope and steep leeward slope, as shown in Figure 6.6. Parameters declared in Table 6.2 are used except the Height parameter, which was set to 10, and we simulate $t = 10000$ time steps. Some dunes are more curved than expected during the simulation; however, they will straighten when the simulation runs longer. Here transverse dunes have already reached an equilibrium state and travel at a constant speed while maintaining the same spacing. According to the contour visualization, the top dune parts are flatter than expected.

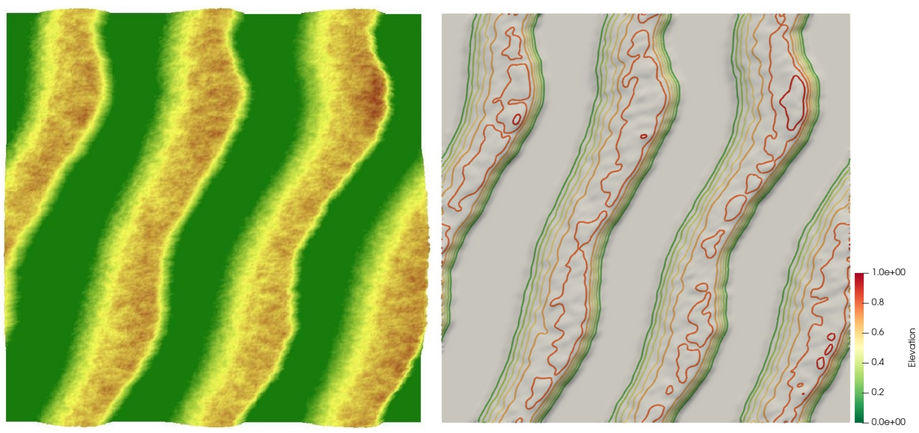


Figure 6.6: Transverse dunes simulated using Momiji's model, maximum elevation reached 34 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Visualized contours in ParaView.

Star dunes shown in Figure 6.7 are simulated using Momiji's method and are wider and slightly lower with flatter top areas. Wind directions were set in order: West, North, East, and South under constant wind speed $L = 5$, and we set 40 time steps per wind season, total of $t = 320$. Initial height was set to $Height = 10$. Other parameters remain the same as in Table 6.2.

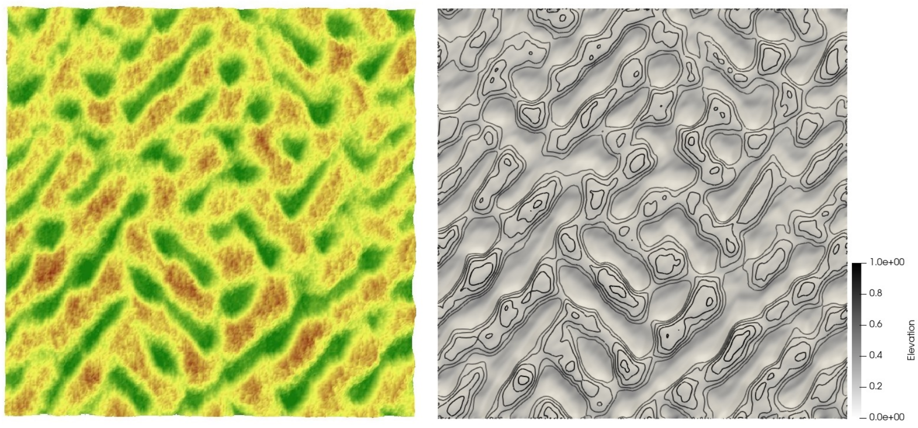


Figure 6.7: Star dunes simulated using Momiji's model, maximum elevation reached 22 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Visualized contours in ParaView.

6.3 Bishop's model

Bishop's model adjusts the wind speed-up computation, ensuring a gentler windward slope for barchans. We confirmed that the dynamic update of wind speed-up computation changes the dune shape, as shown in 6.8. However, some barchans might have the top dune part flatter or even shift slightly against the wind. This artifact is probably due to the quadratic wind speed-up coefficient, which influences the dune's height. The parameters used for the simulation are listed in Table 6.3. Transverse dunes have identical structures to those formed using Momiji's model, and we do not include the result here.

Parameter	Value
Grid type	Flat surface
Angle of Repose	33.7°
Dimension	256×256
Height	3
Slab Height	1/3
p_s	0.6
p_{ns}	0.4
Wind speed	3
Wind direction	West
Linear coef.	0.4
Quadratic coef.	0.002
Advanced wind setup	false
Non-repeated polling	false
Force erosion	false
Time steps	1500

Table 6.3: Parameters for barchan dunes simulation using Bishop's model.

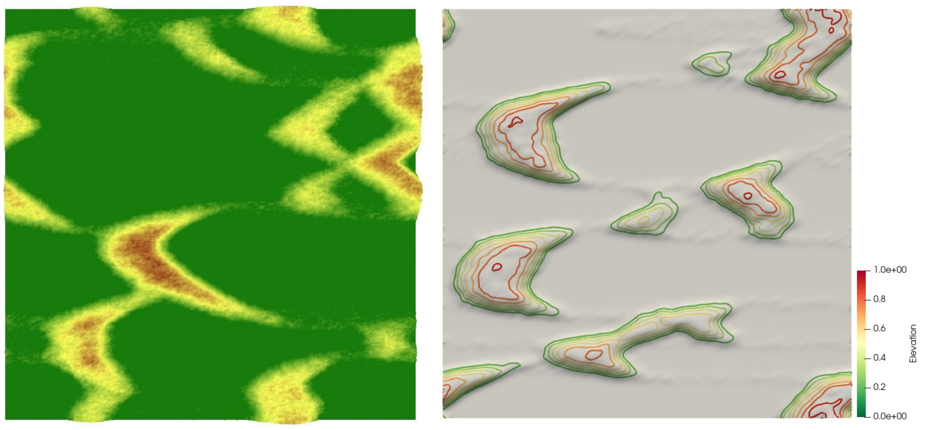


Figure 6.8: Barchan dunes simulated using Bishop's model, maximum elevation reached 28 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Visualized contours in ParaView.

6.4 Dunes In Vegetated Environment

Nield and Baas's model introduced vegetation. We created three simulation scenarios, and the results are presented below. First simulation scene in 6.9 shows Nebkha dune formation, parameters for this simulation are in Table 6.4. 'Natural development' option lets the vegetation grow naturally, based on the vegetation growth functions. By adjusting the wind speed, nebkha dunes have various shapes. When $L = 1$, dunes tend to be wider and shorter, whereas dunes are longer and thinner when $L = 5$. The shorter the annual update of vegetation coefficients, the faster the simulation reaches an equilibrium state. An equilibrium state with vegetation presence occurs when the vegetation coefficients for each cell reach value 1; therefore, erosion and transport will stop. This situation corresponds to a state where the vegetation covers the whole scene, and no sand movement happens. In Nield's model presented in [19], they analyse the equilibrium state in their model by setting different growth function values, which leads to different equilibrium state scenarios.

Parameter	**Value**
Grid type	Flat surface
Angle of Repose	30°
Dimension	256×256
Height	3
Slab Height	1/10
p_s	0.6
p_{ns}	0.4
Wind speed	5
Wind direction	West
Advanced wind setup	false
Non-repeated polling	true
Force erosion	false
Time steps	500
Vegetation update per year	80
Initial vegetation distribution	Natural development
Veg. coef. Grass	0
Veg. coef. Mequite	0
Veg. coef. Shrub	0
Random vegetation distribution	false

Table 6.4: Parameters for nebkha dunes simulation using Nield's model.

We also observed parabolic dunes formation when we let the simulation run longer, with the same parameters as in 6.4 except the time step, which was $t = 1500$. The results resembles parabolic dunes shown in 2.8 B). As mentioned in the result discussion about nebkha dunes, the equilibrium state will change according to growth functions.

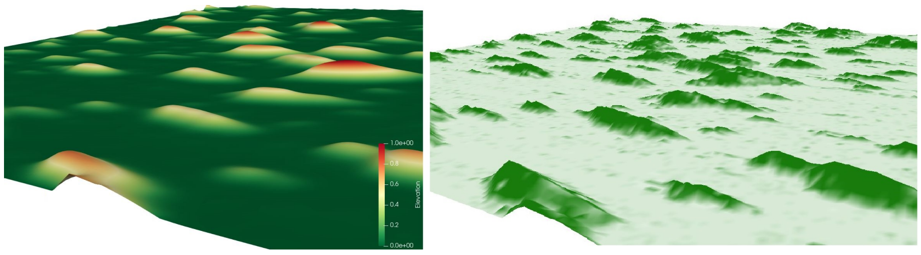


Figure 6.9: Nebkha dunes simulated using Nield's model, maximum elevation reached 58 slabs per cell in the final result. Left: Visualization with elevation and smooth surface in Paraview. Right: Vegetation presence visualized in the application.

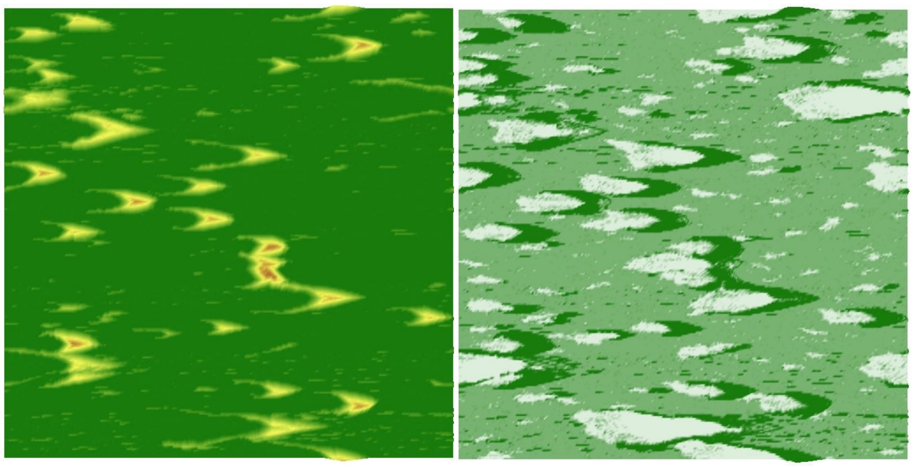


Figure 6.10: Parabolic dunes simulated using Nield's model, maximum elevation reached 85 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Vegetation presence visualized in the application.

The scene with simulated blowouts is similar to the parabolic simulation from blowouts shown in 3.7. We initially placed three 5×6 m (width×length) not vegetated areas. The parameter settings are shown in Table 6.5. Not vegetated area resulted in the material erosion and transport onto the vegetated area in the wind direction. Throughout the simulation, the parabolic dunes were getting larger. In the vegetation presence view, we observe areas with lower vegetation coefficients (light green color), which indicate the previous blowout area. This area is enlarging, and vegetation is present at a lower rate than in other surrounding areas. Parabolic dunes formation from blowouts is also possible because of the higher initial shrub presence because Shrub grows slowly, but it is harder to die out when it establishes. If we lower the shrub presence and make the presence of grass or mesquite higher, the surrounding vegetated area will eventually become extinct, and the whole sediment in the grid will move and form smaller parabolic dunes similar to those shown in Figure 6.10 but will reach equilibrium state faster. Generally, changes in coefficient parameters for each vegetation type have a considerable impact on the simulated scene for Nield's model.

Parameter	Value
Grid type	Flat surface
Angle of Repose	30°
Dimension	100×100
Height	20
Slab Height	1/10
p_s	0.6
p_{ns}	0.4
Wind speed	1
Wind direction	West
Advanced wind setup	false
Non-repeated polling	false
Force erosion	false
Time steps	4000
Vegetation update per year	80
Initial vegetation distribution	Blowouts
Veg. coef. Grass	0.2
Veg. coef. Mequite	0
Veg. coef. Shrub	0.9
Random vegetation distribution	false

Table 6.5: Parameters for parabolic dunes simulated from blowouts using Nield's model.

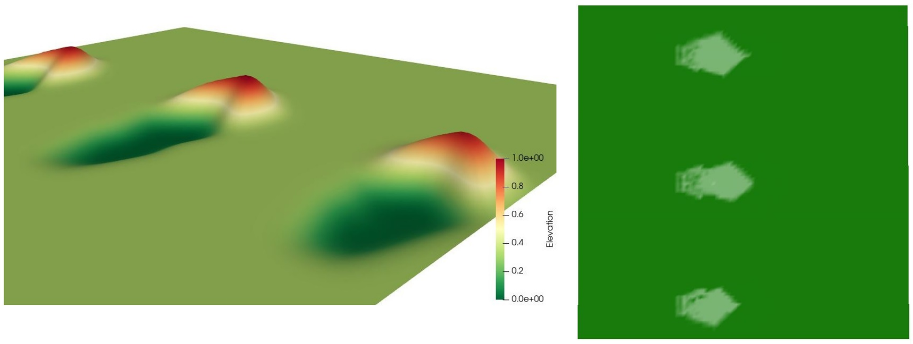


Figure 6.11: Parabolic dunes simulated from blowouts using Nield's model, maximum elevation reached 76 slabs per cell in the final result. Left: Visualization with elevation and smooth surface in Paraview. Right: Vegetation presence visualized in the application.

Finally, we created a scene with three isolated rectangular areas with vegetation presence, Figure 6.12. The parameters are shown in Table 6.6. Since Nields's model is based on Werner's model, dunes outside the vegetated areas developed into transverse dunes covered with vegetation mainly on the windward slope. In the vegetated areas, the sand is caught by the growing vegetation and is moving slower through this area. Letting the simulation run longer results in vegetation growth on all dunes, and the initially vegetated areas will disappear.

Parameter	Value
Grid type	Flat surface
Angle of Repose	30°
Dimension	256×256
Height	20
Slab Height	1/10
p_s	0.6
p_{ns}	0.4
Wind speed	1
Wind direction	West
Advanced wind setup	false
Non-repeated polling	true
Force erosion	false
Time steps	200
Vegetation update per year	80
Initial vegetation distribution	Vegetation islands
Veg. coef. Grass	0.2
Veg. coef. Mequite	0.4
Veg. coef. Shrub	0.6
Random vegetation distribution	true

Table 6.6: Parameters for scene with isolated vegetation islands using Nield's model.

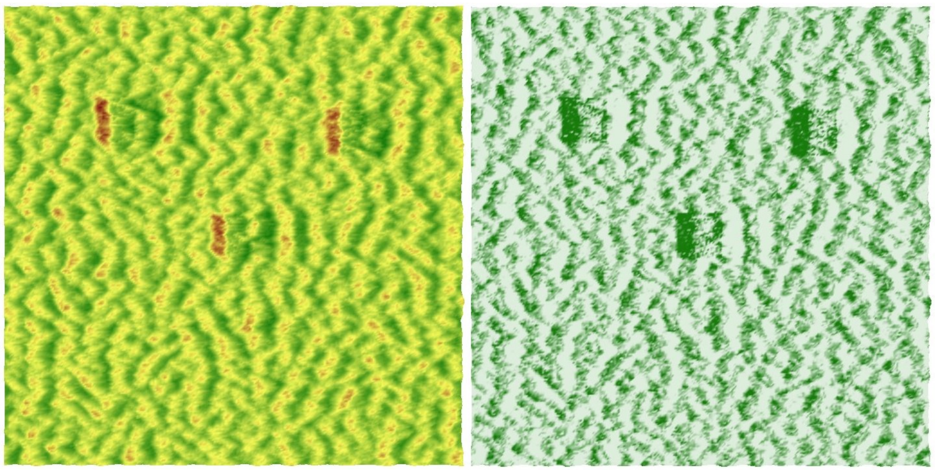


Figure 6.12: Dunes simulated from isolated vegetation islands using Nield's model, maximum elevation reached 57 slabs per cell in the final result. Left: Visualization with elevation in the application. Right: Vegetation presence visualized in the application.

6.5 Various Parameter Settings

A variety of parameters defines implemented simulation models. The combination of those parameters and altering their values leads to different results. This section presents some interesting observations of parameter influence on the simulation result.

6.5.1 Grid Dimension

Firstly, the dimension makes the grid larger, so more dunes can be formed. The dune size tends to be similar to those dunes formed on grids with lower dimensions. We did not observe any marginal differences while using larger grids, except that the simulation runs slower when the dimension is bigger. It also naturally adds more sand into the simulation. In Figure 6.13 we present the result of the barchan dune simulation using Momiji's model, with parameters from Table 6.2 but with dimension set to 1024×1024.

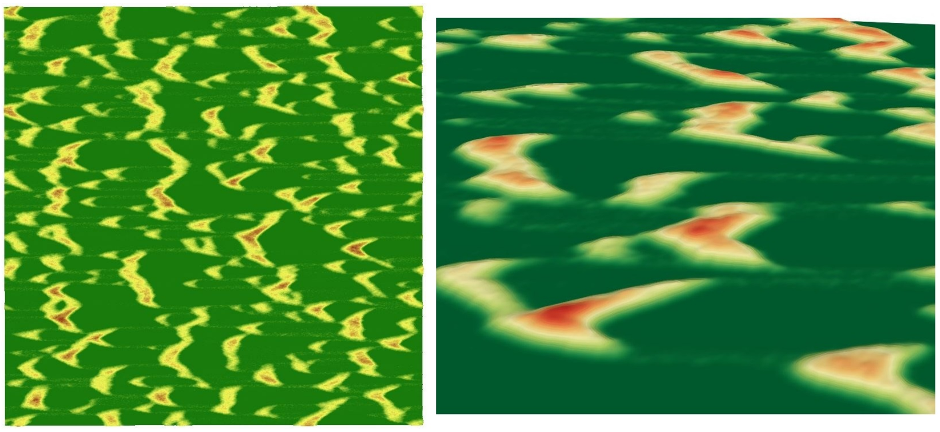


Figure 6.13: Barchan dunes on larger grid, using Momiji's model, maximum elevation reached 29 slabs per cell in the final result. Left: Elevation visualization in the application. Right: Close-up view on smooth surface with elevation in ParaView.

6.5.2 Uneven Initial Grid and Obstacles

The initial grid type might also influence the simulation. In our simulation, we may choose between 'Flat surface' and 'Random height,' which adjust the number of slabs by +− one from the original value specified in the

Height parameter. The difference between those does not significantly impact the simulation; the simulated dune type will be formed. However, the layout and shape of simulated dunes will be different. 'Sand ripples' and 'Transverse dunes' initial grid settings have a surprisingly negligible impact on the simulation. We observed that any simulation algorithm would eventually remove the artificially placed sand slabs, and the dunes will evolve naturally. Even the simulation of transverse dunes ignores the 'Transverse dune' initial grid setting. However, setting the different heights of the non-erodible surface underneath the sand significantly impacts the simulation. We will discuss the case when we do not adjust the model to react to the uneven non-erodible surface. For this purpose, we created the 'Three rocks' scene. The problem is shown in Figure 6.14, where the dunes ignore the rocks and cross them. The sand ignores the height difference and is freely deposited on top of the rock. Only the stabilization algorithm adjusts the slopes thanks to the repose angle. On the other hand, in Figure 6.15, we observe that the sand mass reacts to the obstacles accordingly, and sand is accumulating at the frontal area of the obstacle, and no sand is on the top of the rock. We observe that transverse dunes are formed in areas where rocks are absent. Similar results are for other models and simulated dunes. Adding the impact of non-erodible surface on dune simulation added more realism, which was not considered in existing numerically based models.

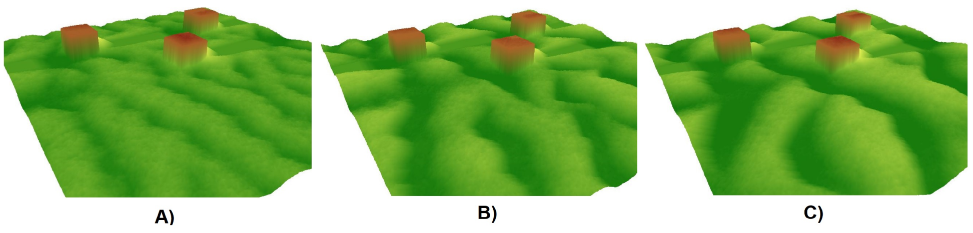


Figure 6.14: Transverse dunes with rocks in the scene. The sand is not accumulating in the frontal rock area. Simulated using Momiji's model, wind from left to right. A) $t = 100$, B) $t = 300$, C) $t = 500$.

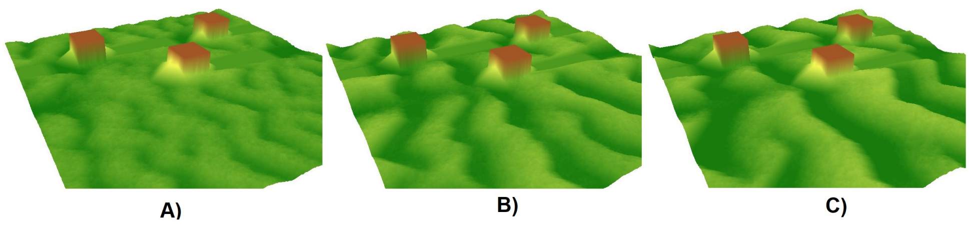


Figure 6.15: Transverse dunes with rocks in the scene. The sand is correctly accumulating in the frontal rock area. Simulated using Momiji's model, wind from left to right. A) $t = 100$, B) $t = 300$, C) $t = 500$.

6.5.3 Angle of Repose

Repose angle has a considerable effect on simulated dunes. The most visible impact can be observed in Nield's model, where the angle of repose is set to 40° when $\rho \geq 0.3$, which allows steeper slopes to be formed. In Figure 6.16 we show the difference between the angle of repose set to 30° and 33.7° using Bishop's model to simulate barchans. Figures taken after $t = 500$ time steps. We observe that contours are closer together for a higher repose angle, representing steeper slopes. Both results are simulated using the same parameters, except the angle of repose.

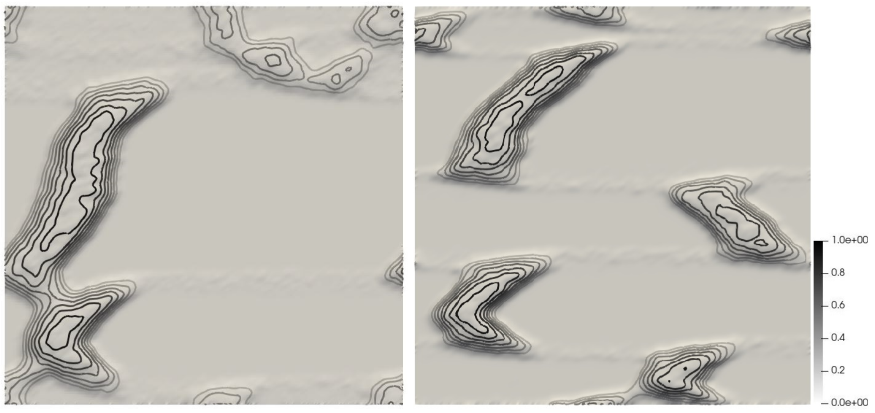


Figure 6.16: Barchan dunes simulated using Bishop's model. Left: Contours, Angle of repose is 30°. Right: Contours, Angle of repose is 33.7°.

6.5.4 Forced Erosion in Simulation Step

The difference between forced erosion and default behavior usually impacts the simulation result. For Werner's approach to barchan dunes simulation, we observed that forced erosion causes the dune surface to be more smooth than in the default simulation step. The simulation of barchan and transverse dunes while using the force erosion option leads to faster dune development than using the default option. The forced erosion option does not significantly impact Star dunes. Having this option disabled in Nield's model leads to slower time step iterations. The most significant difference in simulation speed can be observed while simulating parabolic dunes from blowouts.

Nonetheless, in Figure 6.17 the difference between the forced erosion and default erosion is shown. The barchans are beginning to be formed on the left, whereas, on the right, the barchans are already formed. For other simulation models and simulated dunes, the impact is similar.

Considering the impact of non-repeated polling, we already showed the difference in Figure 6.4. For other simulated dunes, the difference has similar nature; the dunes are formed slower when using repeated polling, and it is usually necessary to simulate with a higher number of time steps t.

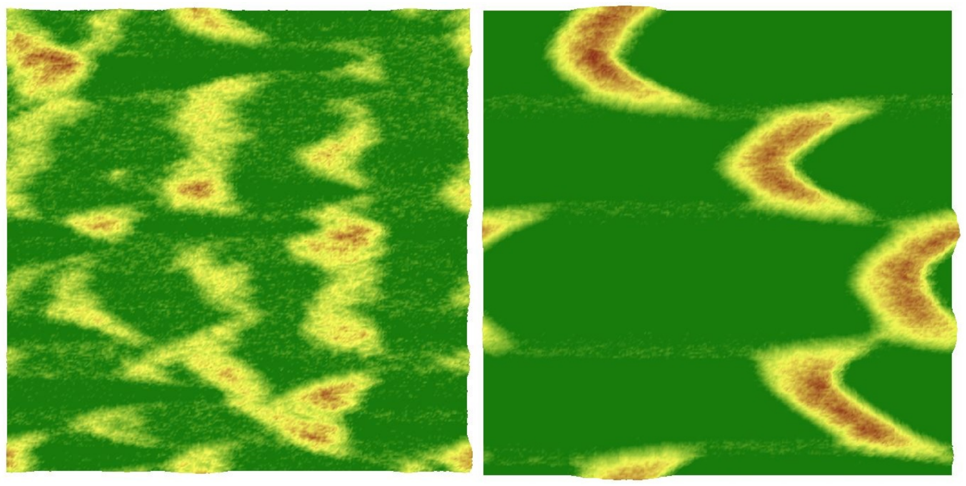


Figure 6.17: Impact of forced erosion in simulation steps on the development of barchans, Bishop's model. Both figures captured after $t = 700$ time steps. Left: Default simulation step. Right: Forced erosion per time step.

6.5.5 Wind Speed

The most significant impact of all other parameters on the simulation has wind speed. Variable wind speed also enables customization options, but setting a correct wind speed for the simulation is crucial. Dunes will not be formed on some occasions even though the wind speed is constant. On the contrary, variable wind speed may also prevent the dunes from forming, even though it may stop the sand from entering the equilibrium state. We will go through our conclusions considering the wind speed.

Firstly, we will discuss Transverse dunes. For Werner's model, transverse dunes were formed on a 100×100 grid from wind speed $L = 2$ to $L = 42$. Less and above this interval, no transverse dunes were formed. The equilibrium state usually consisted of only one transverse dune. However, for wind speed around $L = 40$ the spacing between the dunes was significantly smaller, and the equilibrium state consisted of several transverse dunes instead of only one. The same holds for Momiji's model, except that transverse dunes were formed at $L = 1$. Bishop's model formed transverse dunes on interval $L = [2, 39]$. It is worth mentioning that this interval gets bigger by increasing the grid dimension. We think the periodic boundary conditions cause this behavior.

Setting the correct wind speed for barchans is tougher than for transverse dunes. For Werner's model on grid dimension of 256×256, barchans were formed on interval $L = [3, 10]$. For values of $L \leq 10$, barchans were not formed. Dunes resembling mounds of sand were formed instead – typical barchan 'horns' were not visible. We also observed dunes resembling sand ripples around $L = 50$. For Momiji's model, the interval for wind speed required to form barchans was larger. For $L = 1$ unspecified dunes were formed. Surprisingly, on $L = 2$ the dunes resembled sand ripples. Bishop's model behaved similarly to Momiji's model under the changes of L.

Dunes resembling star dunes were formed under lower or higher wind speeds. For wind speed $L = 1$ and variable wind direction, each after $t = 40$ time steps, the star dunes formed slower than for higher wind speeds. In Figure 6.18, left, sand ripples using Momiji's model were formed for $L = 2$. Star dunes formed using Momiji's method are shown on the right.

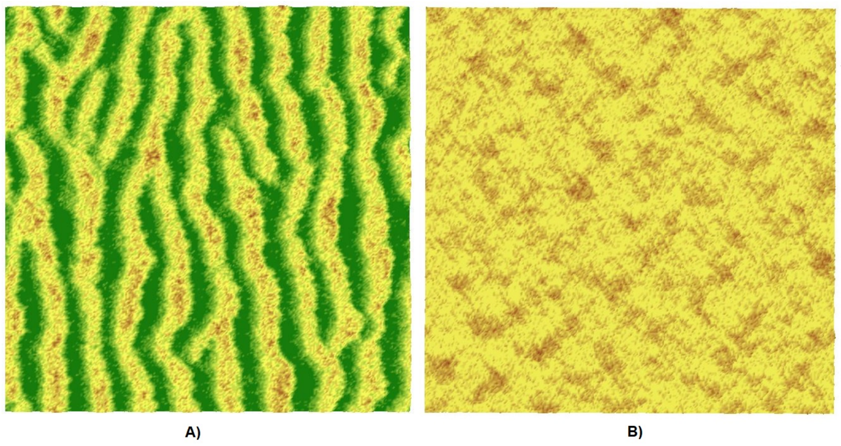
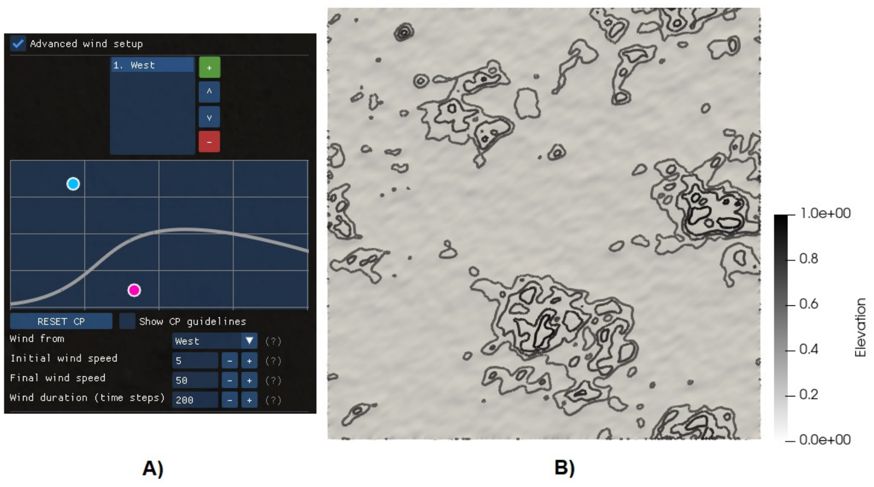


Figure 6.18: Impact of different values of constant wind speed L on Momiji's model. A) $L = 2$, barchans should have formed. B) $L = 1$ for all directions, subtle star dunes are formed after multiple wind regime iterations.

Modifying the wind speed throughout the simulation using the Beziér curve caused the dunes not to reach the equilibrium state. Unspecified dunes were usually formed throughout the simulation. When a specified dune shape began to be formed, the changing wind speed prevented it from further initiation, and the dunes changed into unspecified mounds of sand. In Figure 6.19, we present one result of settings varying wind speed L throughout the simulation using Bishop's model, which resulted in unspecified dune formation. Even though changing wind speed comes naturally into mind, using models based on sand slab movements, which use probability for material deposition, does not lead to any sensible simulation results. Those models are susceptible to changing wind speed and usually require constant wind speed throughout the simulation to form some dunes. Nonetheless, careful setting of wind speed in the simulated wind regime might eventually result in dunes formation while avoiding the equilibrium state.



[✓] Advanced wind setup
1. West
Show CP guidelines
Wind from: West
Initial wind speed: 5
Final wind speed: 50
Wind duration (time steps): 200

Figure 6.19: Impact of different values of variable wind speed L on Momiji's model. A) Settings used for the simulation. B) Simulation result after total of $t = 1000$ time steps, contours applied in ParaView.

6.5.6 Deposition Probability

Adjusting the deposition probabilities p_s and p_{ns} influenced the simulation results. However, those values should be adjusted sensibly, following the expected sand behavior in nature. Figure 6.20 shows the result of adjusting deposition probabilities on barchan simulation using Werner's model. The parameters are taken from 6.1 except we turned on Forced erosion and set probabilities $p_s = 0.4$ and $p_{ns} = 0.9$. An increase of p_{ns} resulted in faster barchan formation because the sand was more likely to bounce on a surface without sand. Lowering p_s ensured sand was more likely to be deposited on a surface containing sand. This eventually resulted in forming of barchans using $t = 500$ time steps only. Barchans also have a more natural shape; probably the best-looking barchans formed using Werner's model.

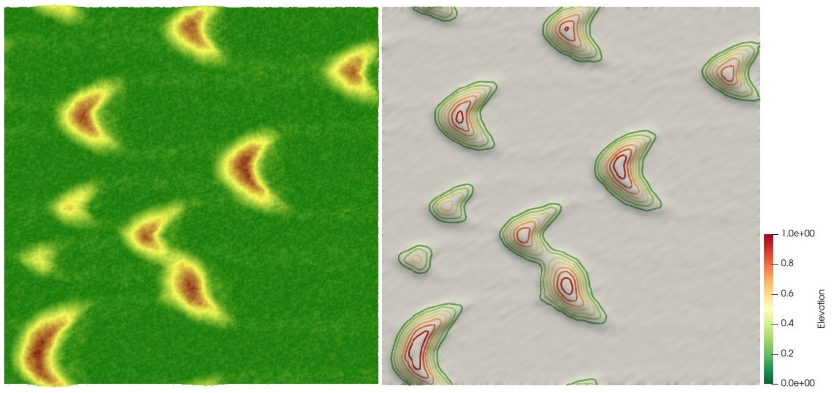


Figure 6.20: Barchans formed using Werner's model. Left: Elevation visualized in the application. Right: Contours applied in ParaView.

6.6 Particle-based Model

We encountered complications implementing Wang's particle-based model. Wind ripples were forming only under certain conditions. The best-looking sand ripples created by Wang's model are shown in Figure 6.21, where the wind speed was set to $10m/s$. Total of $t = 100$ time steps were performed on 100×100 simulation grid. Next, we will review the encountered problems in our implementation.

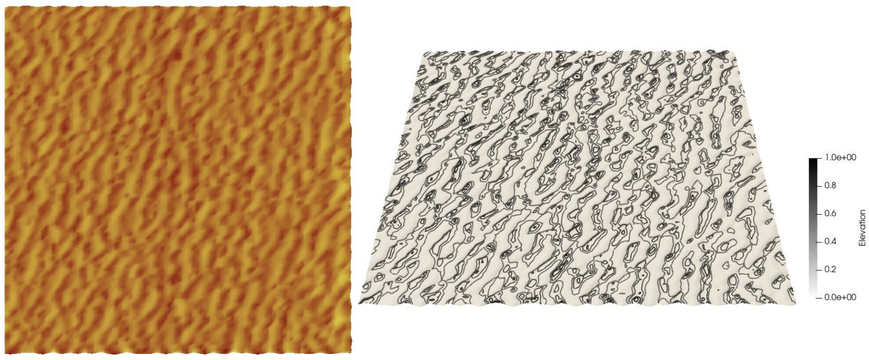


Figure 6.21: Sand ripples formed by Wang's model. Left: Visualized elevation in the application. Right: Contours added in ParaView.

6.6.1 Implementation Problems

The first problem is the ripple movement in the direction against the wind. When the particle trajectories are visualized, most particles do not bounce and are deposited immediately after contact with the ground. Also, the particle's usual deposition position is on the windward side instead of on the leeward side of the specific sand ripple, which might make the sand ripples evolve against the wind. This can be caused mainly by two factors, dependent on parameter settings. Firstly, the dampening factors control the particle bounce for tangential and normal bounce velocity. Setting those parameters based on wind speed might ensure the particle will bounce correctly. Secondly, the problem of wind field computation seems to be the key to the correct implementation of this model. The wind should reach a lower speed while the particle is closer to the ground, and therefore the dampening factors should be scaled differently. However, our implementation of a wind field does not ensure the wind slows down enough while closer to the ground. This artifact then causes the particles to bounce indefinitely because at each 'hop,' they will reach relatively high speed, and on the next bounce, they will not slow down enough. We ensured that the effect of indefinite bounces would not happen by restricting the number of allowed bounces.

Furthermore, correct wind behavior in numerically-based models requires detecting the wind shadow. Transported sand slabs then react to wind shadow accordingly. In implementation by Wang et al. [25] they do now mention the wind shadow explicitly, but when we consider a particle traveling across a sand ripple peak, the wind speed should decrease based on the real-world measurements presented in 2.3. Consequently, the implemented wind field might cause the traveling particle to skip the leeward ripple slope instead of depositing, and the particle will reach the lower windward area of the next sand ripple where it deposits, which might eventually cause the sand ripple movement against the wind.

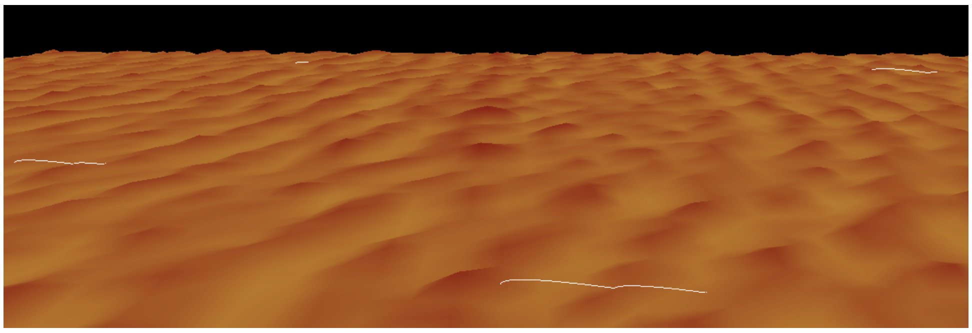


Figure 6.22: Trajectories of particle transportation visualized in our application (white lines).

In Figure 6.22, we show the traced particle trajectory visualized in our application. When the particles are moved according to the periodic boundary conditions on the other side of the grid, a line across the simulation grid shows this event. We assume the correct particle trajectory would differ from those captured in our application. Since the wind slows down near the surface, the particle at the end should fall more. Note that the particle's acceleration after its lift-off is correct because the wind is faster than the initial particle speed, and while the wind carries the particle, it should accelerate. In Figure 6.23 the captured and expected trajectory is visualized. The trajectory should have the expected shape by implementing the correct wind field, where particles would slow down closer to the ground. This way, the infinite bouncing should also be eliminated.

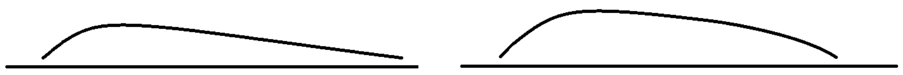


Figure 6.23: Sand grain trajectories. Left: Traced trajectory. Right: Expected trajectory.

6.7 Comparison with Real-world Dunes

This section reviews the authenticity of our results with real-world dunes. Firstly, there are more types of barchans formed in various locations. In Figure 6.24, barchans created by Werner's approach are compared to real-world barchans formed in Paracas National Park, Peru. Simulated barchans are relatively thinner than the real-world example. The shape difference is mainly caused by the lack of wind speed-up in Werner's model, which results in almost equal windward and leeward dune slopes. In both simulated barchans and real-world barchans, smaller barchan dunes are present. We also observed that when simulated barchans get bigger, they usually lack the sharp-angled middle part of the dune at the base of the leeward slope, whereas the real world barchans are usually more 'V-shaped' in this area. The difference is also caused by the lack of more realistic computation of wind speed-up. Apart from the almost equal windward and leeward slopes causing sharper shape, the insufficient wind speed-up computation also causes the simulated barchan dunes to be higher than expected. Based on the comparison with a real-world example, we can confirm that the wind speed-up factor and the ability of sand to be eroded in the wind shadow significantly impact barchan shape correctness.

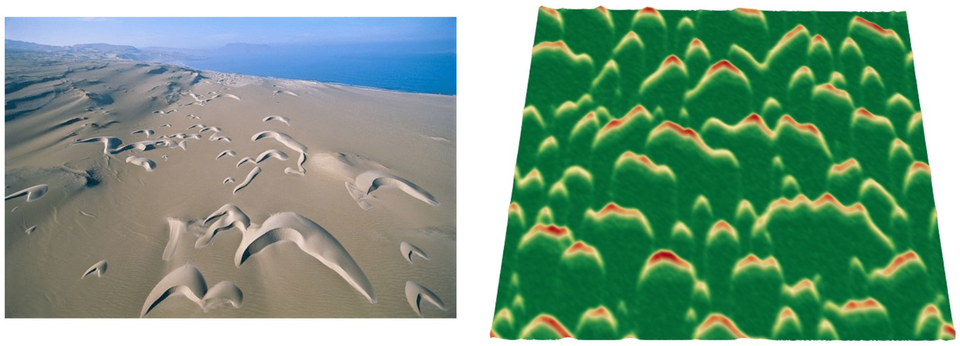


Figure 6.24: Comparison of simulated barchans with real-world dunes. Left: Real-world barchans (from [23]). Right: Barchans generated using Werner's model.

On the contrary, Figure 6.25 shows barchans with a gentler windward slope, simulated using Bishop's model, which resemble the real-world example better. In the lower-left part of real-world photography, we observe that multiple barchan dunes connect, forming one larger dune. This process is typical for simulated barchan dunes as well. Also, barchans simulated using Bishop's model are lower and sharper at the base of the middle leeward slope than the result generated by Werner's model, which gives the simulated result better authenticity. Both models generate dunes with elongating 'horns' in a unidirectional wind regime, typical for real-world barchans.

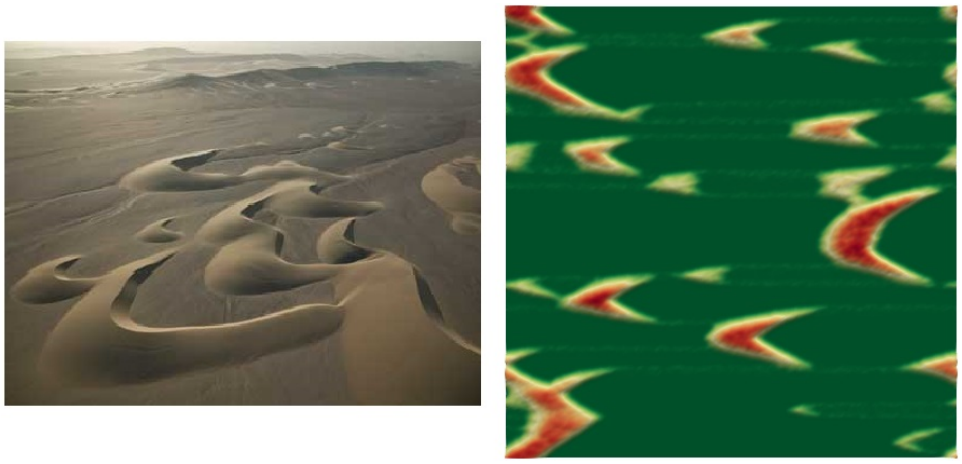


Figure 6.25: Second comparison of simulated barchans with real-world dunes. Left: Real-world barchans (from [20]). Right: Barchans generated using Bishop's model.

Linear dunes are formed mainly in two directions, and they usually evolve in wave patterns, as shown previously in schematic 2.7. In Figure 6.26 we present the comparison of real-world linear dunes and linear dunes formed using Werner's model. Contrary to the simulated result, the real-world linear dunes are more 'wavy' and have a sharper crest. The difference in crest sharpness is caused primarily by two factors. Firstly, the sharper crest shape cannot be represented in the simulation grid due to repose angle constraints and material stabilization algorithms. Even if those factors were addressed, the simulation grid visualization does not support sharp edges, which a different visualization approach may resolve. Secondly, since we cannot choose arbitrary wind directions required for more realistic linear dune development, the simulated result is formed using opposing wind directions, which eventually causes the dunes to be less 'wavy.'

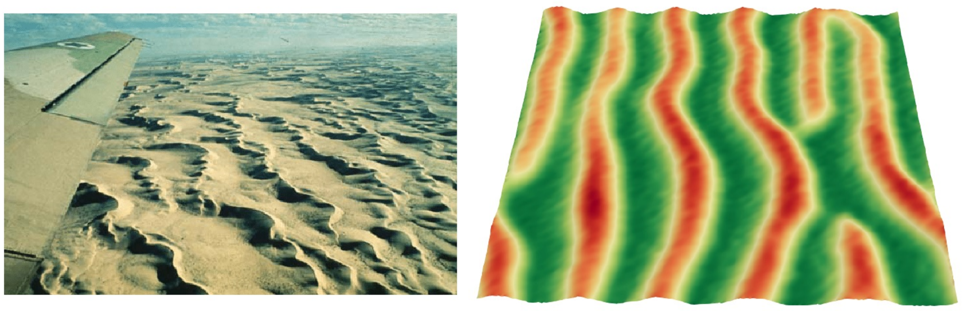


Figure 6.26: Comparison of simulated linear dunes with real-world dunes. Left: Real-world linear dunes (from [24]). Right: Linear generated using Werner's model.

Transverse dunes are formed under the same conditions as barchans, but more sand available for transport is required to form them. The comparison of real-world examples with our results is in Figure 6.27. Real-world transverse dunes are close together, and right after one transverse dune's leeward slope ends, the windward slope of the next dune begins. The real-world photo confirms the previously shown transverse dune sketch in Figure 2.5. Momiji's model generated a transverse dune with higher spacing than spacing in the real-world example. This difference is caused by the wind speed concept in the probability-driven simulation model, which does not allow the dunes to be formed closer together. Transverse dunes are closer together during the early stages of the simulation, but their shape is not straight – they straighten later when more time steps are simulated. However, apart from the non-realistic spacing, we observe the sharper leeward slope and gentler windward slope typical for transverse dunes. The results resemble transverse dunes simulation results presented by Momiji et al. in Figure 3.3.

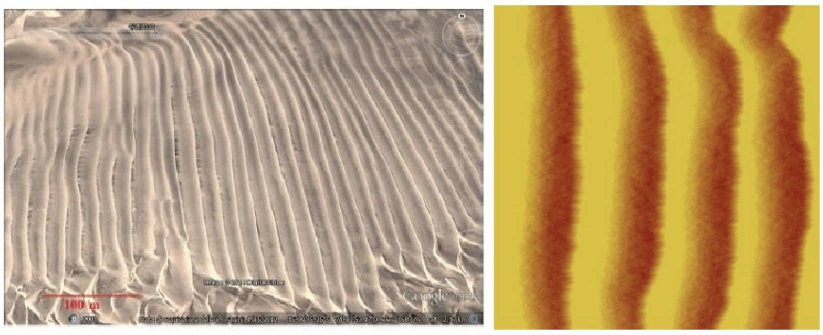


Figure 6.27: Comparison of simulated transverse dunes with real-world dunes. Left: Real-world transverse dunes (from [22], modified). Right: Transverse dunes generated using Momiji's model.

Star dunes have a central part and radiating 'arms' originating here. Simulation of star dunes in our implementation in comparison with real-world aerial photography is shown in 6.28. Simulated dunes tend to have wider 'arms' closer together than real-world star dunes. The crest sharpness absence in our results was explained previously in the linear dunes comparison. We also observe that simulated star dunes are more connected than the real-world example, which might be caused by the absence of arbitrary wind direction settings. The small dune spacing is caused by the wind speed concept in the numerically-based sand simulation model. We conclude that Momiji's numerically based model could not reproduce star dunes perfectly as can be found in nature. On the other hand, the simulation results resemble the star dunes simulation presented by B. T. Werner in their simulation results in Figure 3.2.

Nebkha dunes are sand dunes created in vegetated areas. The shape of nebkha dunes generated by our application resembles a real-world example. The comparison is shown in Figure 6.29. We observe that sand accumulates

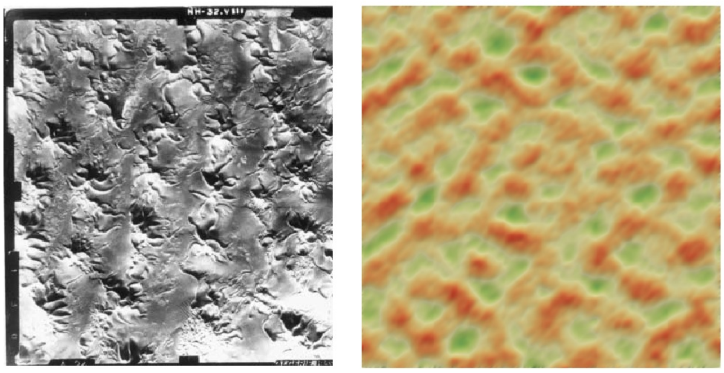


Figure 6.28: Comparison of simulated star dunes with real-world dunes. Left: Real-world star dunes (from [24]). Right: Star dunes generated using Momiji's model.

on the leeward dune side, elongating the dune shape. However, in the results shown in Section 6.4, we observed that the vegetation covers almost the entire dune, which does not precisely correspond to the real-world principles discussed in Chapter 2. On the other hand, the vegetation coverage in our simulation result resembles the simulated nebkha dunes presented by Nield et al. quite well. Adjustments to vegetation presence evolution and distribution might be made by adjusting the vegetation growth functions to fit real-world examples better.

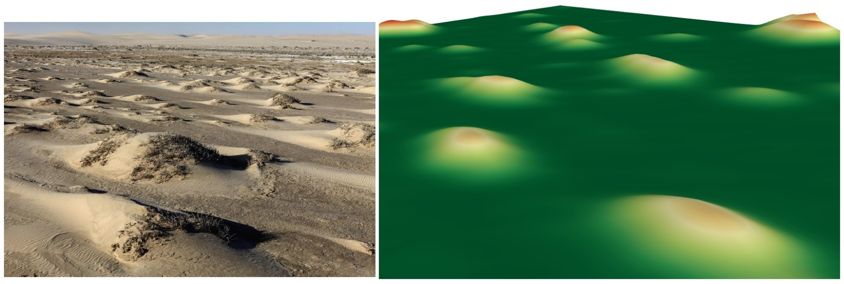


Figure 6.29: Comparison of simulated nebkha dunes with real-world dunes. Left: Real-world star dunes (from [10]). Right: Nebkha dunes generated using model by Nield et al.

Parabolic dunes are vegetated dunes, shaped similarly to barchans but with 'horns' elongating against the wind direction. The dunes are also covered with vegetation, especially covering the 'arms,' so the middle part can travel faster. The comparison is in Figure 6.30. Firstly, the shape resembles the real-world example quite well, especially the extending middle section of parabolic dunes. We cannot see the vegetation coverage in the real-world example, and therefore we cannot precisely determine the vegetation coverage correctness of our simulation. The vegetation coverage imprecision was already discussed while comparing nebkha dunes with real-world examples, which applies to parabolic dunes, too – the vegetation should not cover almost the entire dune; it should be rather on the base of the windward slope.

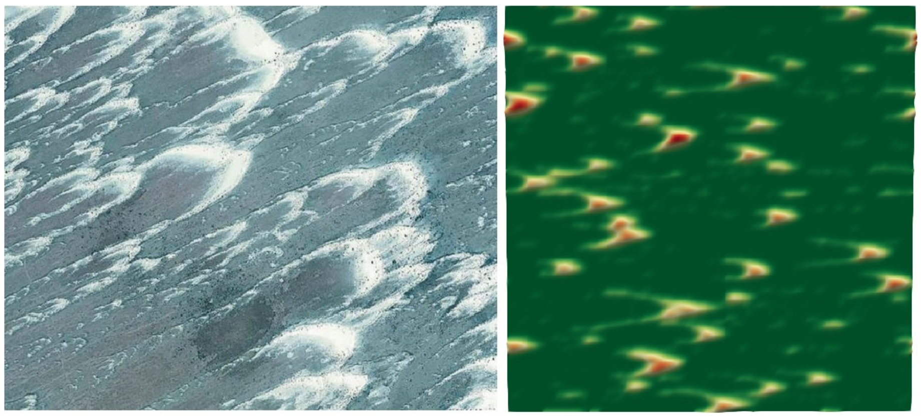


Figure 6.30: Comparison of simulated parabolic dunes with real-world dunes. Left: Real-world parabolic dunes (from [24], modified). Right: Parabolic dunes generated using model by Nield et al.

6.8 Snow Simulation Results

Finally, we present the results of our snow simulation implementation. Our approach is experimental, based on a previously introduced snow behavior theory, and not based on any known existing snow simulation model. However, we present the result of snow ripples development over time in Figure 6.31. The simulation parameters are shown in Table 6.7. Throughout the simulation, the temperature was constant. The target time step count for the simulation was set to $t = 100$, but the simulation ended after $t = 98$ time steps. Essentially, this corresponds to a situation where snow was located in an area without any precipitation for 98 days, and eventually, the snowflakes 'decayed' to a state where they cannot be transported by wind. Our model does not consider the temperature difference between day and night or the impact of sun shafts and the melting process itself. However, the snow simulation added the development of snow based on parameters influencing the snow development.

We observe that the snow formations in our simulation results do not precisely represent the real-world snow ripples shown previously in Figure 2.12. The shape is not consistent, but their height might correspond to the real-world observation that snow dunes are generally smaller than sand dunes. In order to improve the snow simulation, more realistic thermodynamic computations should be accounted for, and more attention should be given to analyzing the snow behavior.

Parameter	Value
Grid type	Flat surface
Angle of Repose	30°
Dimension	256×256
Height	3
Slab Height	1/3
p_s	0.6
p_{ns}	0.4
Wind speed	20
Wind direction	West
Advanced wind setup	false
Initial temperature	-20°C
Target temperature	-20°C
Target time steps	100

Table 6.7: Parameters for scene with snow ripples.

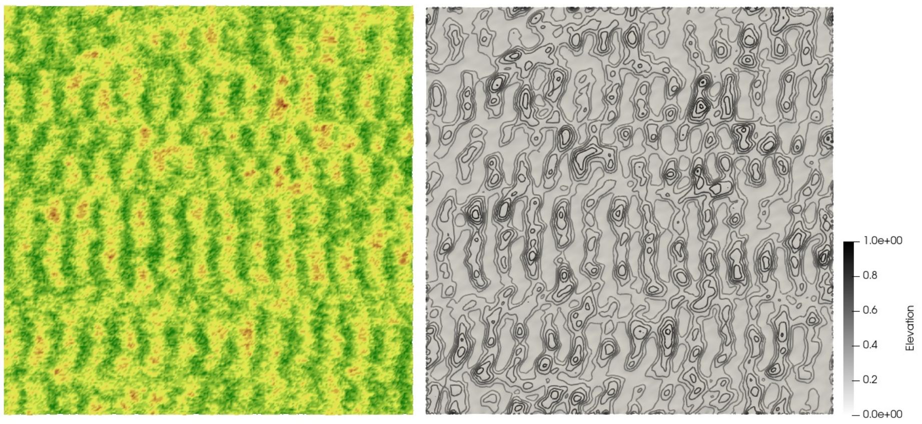
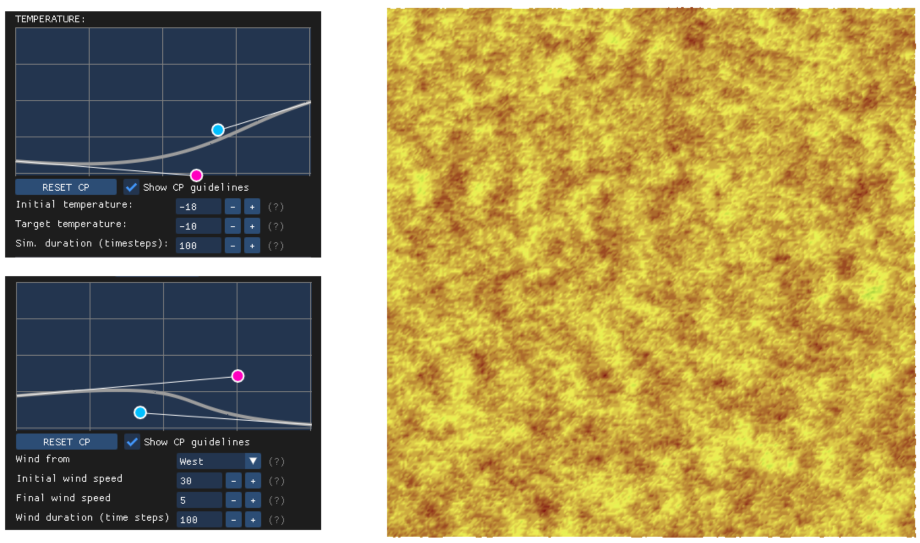


Figure 6.31: Snow ripples simulated using a snow model, initially based on B. T. Werner's sand model. Left: Simulation result with elevation visualized in the application. Right: Contours added in ParaView.

Our final result shows our attempt to simulate snow barchans or sastrugi. In Figure 6.32 we observe that no apparent snow dunes are formed; however, in the left part, we may observe barchan-shaped snow formations resembling snow barchans starting to be formed. Results were simulated with changing temperatures throughout the simulation and changing wind speed. The wind blows from the west. The snow was immune to erosion after $t = 51$ time steps. Temperature and wind regime settings are also shown in Figure 6.32.



```
Initial temperature:          -18
Target temperature:           -10
Sim. duration (timesteps):    100
```

```
Wind from                     West
Initial wind speed            30
Final wind speed              5
Wind duration (time steps)    100
```

Figure 6.32: Attempt to simulate snow barchans or sastrugi using a snow model initially based on B. T. Werner. Left: Temperature and wind regime settings. Right: Simulation result with elevation visualized in the application.

Chapter 7

Conclusion

This thesis aimed to investigate several sand simulation approaches and create an interactive application used for changing the simulation parameters. Simulation models were identified and implemented to analyze their behavior. In Chapter 6, we reported and discussed the correctness of the simulation results on barchans, transverse dunes, linear dunes, star dunes, parabolic dunes, and other dune types using those models. We also concluded that dunes tend to enter an equilibrium state after the simulation runs long enough, depending on the parameters settings. Theoretically, this state can be prevented by the careful setting of the wind regime. Based on drawbacks of some implemented simulation models, techniques to eliminate artifacts were proposed, such as the impact of uneven erodible surface on the sand dune formation. The most significant parameters influencing the simulation behavior were identified, and their influence was explained. The most crucial parameter to set was the wind speed. Also, because of the unclear nature of sand grain entrainment, implemented simulation procedures are limited in the erosion procedure and time domain. Finally, we discussed the difference between simulated dunes with real-world photos.

The wind-driven snow simulation was added to the application using some of the simulation parameters used for snow bedform development, such as temperature and snow density. Simulating snow is similar to the sand simulation approach while considering the differences between sand and snow. Because of the simplification of our snow simulation, we could not replicate a variety of snow dune types.

7.1 Future Work

The addition of other particle-based simulation models to simulate sand will enlarge the model collection. Some snow simulation models might be added, and sand and snow behavior similarities might be investigated. Models with hexagonal grids might improve the wind direction variability.

Implemented simulation algorithms are serial. Optimizing the current algorithms and enabling parallelism will result in faster computation; however, the behavior of parallel simulation algorithms has to be carefully examined because the computations should be independent, especially when using the GPU's massive parallel computation capability to a maximum.

Wang's particle-based model [25] in our implementation has problems concerning the wind field setup and parameters setting; we observed unnatural behavior of the implemented model. Those problems should be further resolved. The key to solving these problems seems to be understanding wind speed behavior concerning the wind shadowed areas, which might lead to correct wind field behavior. Modifying the sand grain diameter of each generated particle might result in more realistic results. Additionally, it might be worth considering simulating snow instead of sand using Wang's model and observing the results.

Further identification of more simulation parameters influencing the simulation might result in more natural results, especially considering the snow simulation. More snow simulation parameters might be introduced to extend our work, and current parameters may be more realistically based.

Visualization in our application can be improved by implementing contour visualization or other visualization modes. Also, techniques used to smooth the simulation grid might be applied. Different approaches to visualization should be taken to visualize sharp dune crests.

Bibliography

[1] Steven Bishop, Momiji Hiroshi, Ricardo Carretero-Gonzalez, and Warren Andrew. Modelling desert dune fields based on discrete dynamics. *Discrete Dynamics in Nature and Society*, 7, 02 2002. doi:10.1080/10260220290013462.

[2] Bastien Chopard, Re Masselot, and Alexandre Dupuis. A lattice gas model for erosion and particles transport in a fluid. *Computer Physics Communications*, 129, 01 2000. doi:10.1016/S0010-4655(00)00104-1.

[3] Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and Marie-Paule Cani. Interactive generation of time-evolving, snow-covered landscapes with avalanches. *Computer Graphics Forum*, 37, 04 2018. doi:10.1111/cgf.13379.

[4] Comité Européen de Normalisation. Iso 14688-1:2017 (e): Geotechnical investigation and testing – identification and classification of soil – part 1: Identification and description. 2017. URL: https://www.sis.se/api/document/preview/80000191.

[5] N.J. Doesken, A. Judson, and Colorado Climate Center. *The Snow Booklet: A Guide to the Science, Climatology, and Measurement of Snow in the United States*. Colorado Climate Center, Department of Atmospheric Science, Colorado State University, 1997. URL: https://climate.colostate.edu/pdfs/snowbook.pdf.

[6] Noritaka Endo. Simple stochastic cellular automaton model for starved beds and implications about formation of sand topographic features in terms of sand flux. *Progress in Earth and Planetary Science*, 3(1):28, 9 2016. doi:10.1186/s40645-016-0102-9.

[7] Paul Fearing. Computer modelling of fallen snow. 05 2001. doi:10.1145/344779.344809.

[8] Simon Filhol and Matthew Sturm. Snow bedforms: A review, new data, and a formation model. *Journal of Geophysical Research: Earth Surface*, 120, 07 2015. doi:10.1002/2015JF003529.

[9] Martin Fousek. Simulace vzniku a vývoje písečných dun. *Master's thesis, CTU FEL*, 2006.

[10] Francisco Gutiérrez and Mateo Gutiérrez. *Desert and Aeolian Landforms*, pages 237–259. Springer International Publishing, Cham, 2016. doi: 10.1007/978-3-319-26947-4_13.

[11] Giulio Iovine, Salvatore Di Gregorio, D. D'Ambrosio, Lupiano Valeria, Rocco Rongo, and William Spataro. Simulating debris flows through a hexagonal cellular automata model: Sciddica (release s3a). 04 2002.

[12] Jasper Kok, Eric Parteli, Timothy Michaels, and Diana Francis. The physics of wind-blown sand and dust. *Reports on progress in physics. Physical Society (Great Britain)*, 75:106901, 09 2012. doi:10.1088/0034-4885/75/10/106901.

[13] Ron Kroetz. Rippling sand - mojave national preserve, ca, usa, 2015. URL: https://live.staticflickr.com/5715/23046250785_d1f08e5037_k.jpg.

[14] Nicholas Lancaster. *Dune Morphology and Dynamics*, pages 557–596. 01 2009. doi:10.1007/978-1-4020-5719-9_18.

[15] Shiguang Liu, Zhangye Wang, Zheng Gong, Lei Huang, and Qunsheng Peng. Physically based animation of sandstorm. *Computer Animation and Virtual Worlds*, 18(4-5):259–269, 2007. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.190, doi:https://doi.org/10.1002/cav.190.

[16] Zhen Liu, Hao Sun, Ke Lin, Cuiying Zhou, and Wei Huang. Occurrence regularity of silt–clay minerals in wind eroded deserts of northwest china. *Sustainability*, 13(5), 2021. URL: https://www.mdpi.com/2071-1050/13/5/2998, doi:10.3390/su13052998.

[17] Andrea Lo Giudice and Luigi Preziosi. A fully eulerian multiphase model of windblown sand coupled with morphodynamic evolution: erosion, transport, deposit and avalanching. *Applied Mathematical Modelling*, 79, 08 2019. doi:10.1016/j.apm.2019.07.060.

[18] Hiroshi Momiji, Ricardo Carretero-Gonzalez, STEVEN BISHOP, and Andrew Warren. Simulation of the effect of wind speedup in the formation of transverse dune fields. *Earth Surface Processes and Landforms - EARTH SURF PROCESS LANDF*, 25, 08 2000. doi: 10.1002/1096-9837(200008)25:83.0.CO;2-Z.

[19] Joanna Nield and Andreas Baas. Investigating parabolic and nebkha dune formation using a cellular automaton modelling approach. *Earth Surface Processes and Landforms*, 33:724 – 740, 04 2008. doi:10.1002/esp.1571.

[20] Axel Paris, Adrien Peytavie, Eric Guérin, Oscar Argudo, and Eric Galin. Desertscape simulation. *Computer Graphics Forum*, 38, 10 2019. doi:10.1111/cgf.13815.

[21] F.J. Pettijohn, P.E. Potter, and R. Siever. *Sand and Sandstone*. Springer Study Edition. Springer New York, 2012. URL: https://books.google.cz/books?id=FFEyBwAAQBAJ.

[22] Amelia Carolina Sparavigna. Peruvian transverse dunes in the google earth images. 12 2014.

[23] Mahdad Talebpour. *Numerical Investigation of Turbulent-Driven Secondary Flow*. PhD thesis, 08 2016. doi:10.13140/RG.2.2.28589.28644.

[24] Haim Tsoar. Types of aeolian sand dunes and their formation. 2001.

[25] Ning Wang and Bao-Gang Hu. Real-time simulation of aeolian sand movement and sand ripple evolution: A method based on the physics of blown sand. *J. Comput. Sci. Technol.*, 27:135–146, 01 2012. doi:10.1007/s11390-012-1212-5.

[26] B. T. Werner. Eolian dunes: Computer simulations and attractor interpretation. *Geology*, 23(12):1107–1110, 12 1995. doi:10.1130/0091-7613(1995)023<1107:EDCSAA>2.3.CO;2.

[27] Bob Wick. Cadiz wilderness and valley. URL: https://www.flickr.com/photos/blmcalifornia/24849806461/.

[28] Giles F.S. Wiggs. Desert dune processes and dynamics. *Progress in Physical Geography: Earth and Environment*, 25(1):53–79, 2001. doi:10.1177/030913330102500103.

Appendix A

Application Usage, Third-Party Addons

A.1 Application Usage, Source Code

Visual Studio 2019 was used as IDE for development. The application uses C++17, OpenGL 4.6, and Freeglut and GLEW libraries. The computer specifications are shown in Table A.1 below. Doxygen was used to generate the application's code documentation. In order to run the application, a multi-threaded processor is required. While using the application, the memory requirements vary by the generated simulation grid size. Therefore we recommend using lower grid dimensions. Using higher grid dimensions may also cause frame rate drops due to CPU-GPU bottleneck because the elevation map update is performed each frame to ensure real-time visualization. This problem might be resolved by implementing a heightmap update pipeline, where only selected heightmap elevation values would be updated.

Model	DELL G5 5587
OS	MS Windows 10 Home 64-bit
Processor	Intel Core i7-8750H
GPU	NVIDIA GeForce GTX 1060 6GB
RAM	16GB

Table A.1: Computer specifications.

The user can launch the application using the binary 'SandAndSnow.exe' attached to this thesis. Configuration files can be loaded from the file browser dialog when clicking on 'LOAD CONFIG.' By clicking on the button 'CREATE,' the simulation is created. Then the user specifies the required number of time steps and clicks on 'RUN.' Example configuration files can be loaded from folder '../cfg/' by clicking on 'LOAD CONFIG.'

The source code is attached to this thesis and can be reviewed in Visual Studio 2019. Building the project in Release and x64 architecture is recommended after opening the project.

The memory requirements of our application depend on the simulation grid size, the simulation model used, and the operating system. The simulation speed is dependent on the computer specifications.

A.2 Third-Party Addons

- Dear ImGUI, a graphical user interface suitable for OpenGL. Available at 'github.com/ocornut/imgui'.
- ImGUI add-ons, add-on widgets for GUI library Dear ImGui. Used for File browser dialog. Available at 'github.com/gallickgunner/ImGui-Addons'.
- ImGUI Bezier curve widget. The code was slightly modified to fit the implementation. Available at 'github.com/ocornut/imgui/issues/786'.
- Stb_image, public domain library for loading images. Available at github.com/nothings/stb.
- JSON parser, used for creating and parsing configuration files. Available at 'github.com/nlohmann/json'.
- ParaView, a third party software suitable for visualisation of volumetric data and export into modelling software. Available at www.paraview.org.
- Overleaf, an easy to use, online, collaborative LaTeX editor. Available at www.overleaf.com.

Appendix B

List of Attachments

- This thesis as a PDF file
- An executable application
- A folder containing the application source code
- Doxygen documentation of the source code
- Screenshots of the application