Real-Time Transition Texture Synthesis for Terrains

John Ferraris

Faculty of Science and Technology

Bournemouth University

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of

Doctor of Philosophy

June 2014

Copyright Statement

Abstract

Depicting the transitions where differing material textures meet on a terrain surface presents a particularly unique set of challenges in the field of real-time rendering. Natural landscapes are inherently irregular and composed of complex interactions between many different material types of effectively endless detail and variation. Although consumer grade graphics hardware is becoming ever increasingly powerful with each successive generation, terrain texturing remains a trade-off between realism and the computational resources available. Technological constraints aside, there is still the challenge of generating the texture resources to represent terrain surfaces which can often span many hundreds or even thousands of square kilometres. To produce such textures by hand is often impractical when operating on a restricted budget of time and funding.

This thesis presents two novel algorithms for generating texture transitions in real-time using automated processes. The first algorithm, Feature-Based Probability Blending (FBPB), automates the task of generating transitions between material textures containing salient features. As such features protrude through the terrain surface FBPB ensures that the topography of these features is maintained at transitions in a realistic manner. The transitions themselves are generated using a probabilistic process that also dynamically adds wear and tear to introduce high frequency detail and irregularity at the transition contour.

The second algorithm, Dynamic Patch Transitions (DPT), extends FBPB by applying the probabilistic transition approach to material textures that contain no salient features. By breaking up texture space into a series of layered patches that are either rendered or discarded on a probabilistic basis, the contour of the transition is greatly increased in resolution and irregularity. When used in conjunction with high frequency detail techniques, such as alpha masking, DPT is capable of producing endless, detailed, irregular transitions without the need for artistic input.

Contents

List of Figures

List of Tables

Acknowlegements

I would like to thank my supervisors, Feng Tian and Christos Gatzidis, for their help, guidance and knowledge, without whom this PhD would not have been possible. I would also like to thank my family for their endless love and support throughout this journey of mine. Finally, I would like to thank Bournemouth University and its administrative staff for giving me this opportunity to pursue my passion and ensuring each milestone was met in a timely fashion.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

John Ferraris
June 2014

Chapter 1

Introduction

Cutting edge, 3D graphics real-time applications have always presented a trade-off between processing power and realism (Rong-hua, 2011). The resources required to accurately model and render each and every object in a scene to a fine degree of detail have historically been beyond the reach of practical technology, particularly in real-time. Even the most mundane of natural scenes has a vastness of scope and detail we take for granted which casts a long shadow over cutting edge computer generated environments. Instead, computer graphics is about creating the illusion of reality with an approximation of the complexity found in the natural world through technological sleight of hand. Take for instance a typical football pitch; there will be some 190 million blades of grass alone, each of which will have a unique shape, size and texture that would be unfeasible to recreate by hand and render in any reasonable timescale. Instead, like a stage magician's well-honed trick, the appearance of detail and complexity is achieved with an elaborate setup of smoke and mirrors, creatively pushing the technology to its limits (Boulanger, 2008; Boulanger et al., 2006; Shah et al., 2005).

Modelling the subtle and unique nuances of an object's surface is simply not feasible for most scenes, thus simpler geometric shapes are used as approximations (Herrera, 2010). The basic rendering primitives used by the majority of graphics hardware are points, lines and triangles (Akenine-Möller et al., 2008, p. 8) as these are the primitives that have hardware paths on modern GPUs. One of the first 3D games ever was "Battlezone", a simple tank simulation that used low-polygon-count wireframe graphics to portray a simple landscape interspersed with cubes, pyramids, and tanks (Rollings and Morris, 2003, p. 516). Modern 3D games use far more sophisticated techniques and effects and are capable of rendering considerably more geometry and detail. However, the triangle budget for typical scenes is still far below that which can reflect reality using geometry alone. For example, CryTech's

CryEngine 3 recommends a budget of up to 11.5k triangles for main characters, which includes all body parts and attachments, except weapons (Crytek, 2011a). Compared to id Software's "Quake" (arguably the first truly 3D first-person shooter) where triangle counts were measured in the hundreds this is a significant improvement but still falls far short of real life, particularly when viewed up close.

The field of real-time, 3D terrain rendering has historically felt the brunt of these technological limitations. As such, indoor environments dominated the first few generations of 3D games in the 1990s, starting with simplistic interiors populated with sprites (Id Software, a,c) before moving swiftly on to fully 3D environments (Epic Games; Id Software, b). The reason for this head start is the relative simplicity of interior environments: they are typically composed of simple geometric shapes, contain repetitive and/or uniform surface patterns that lend themselves to smaller texture sets, contain rooms and corridors that make for efficient occlusion culling and, are of course smaller in size compared to the natural landscape.

For terrain rendering, a mesh structure called a *heightmap* is commonly used, consisting of a vector field where the height components of each vertex are displaced by a value representing the elevation of that point on the mesh. Heightmaps are more efficient to store and render than polygon soups as they take advantage of the contiguous layout of vertices, allowing for easy rendering and level of detail optimizations (Duchaineau et al., 1997; Losasso and Hoppe, 2004). The displacement values can be encoded as a texture for sampling on the GPU, allowing for a purely hardware accelerated rendering path and, as properties such as normals can be derived from the displacement texture the vector field, the vertex data is both compact to store and can be re-used for different terrains by simply swapping out the displacement texture. The displacement texture itself can be generated algorithmically using techniques such as Perlin noise (Musgrave et al., 1989; Perlin, 1985, 2002) and mid-point displacement (Müller et al., 2007) for a realistic-looking fractal terrain or even generated by hand in any conventional image editing software.

Geometry alone is often insufficient to convincingly model real-time scenes so it is common to have images projected onto the geometric surface to simulate greater detail, a procedure known as texture mapping. Texture mapping has its roots from over 40 years ago and has continued to be a busy area of research (Azariadis and Aspragathos, 2000; Heckbert, 1986, 1989; Norton et al., 1982). While the overall shape of the object remains unchanged, the surface detail is greatly increased by the image that has been wrapped around it. From some distance away, it can be difficult to even distinguish what pieces of visual detail are the shape of the object and which are simply features of the image applied to the surface (van Verth and Bishop, 2008, pp. 292-293).

In 1998, Nvidia released the Riva TNT graphics card, the first consumer hardware accelerator to utilize a technique called multitexturing in hardware (nVidia, 2013), where textures are bound to different texture units on the GPU and mapped within a single geometry pass. This allowed for more elaborate and detailed scenes using techniques such as light mapping (first demonstrated in real-time on consumer PCs in 1996 with the release of id Software's Quake (Abrash, 1996)) to be performed more efficiently (Blythe et al., 1999) as well as increasing the illusion of interior and exterior surface detail with techniques such as bump maps (Blinn, 1978a; Cohen et al., 1998; Lewis, 1989; Max and Becker, 1994) and continuing to ever more complex parallax effects (Chen and Chang, 2008; McGuire and McGuire, 2005; Policarpo et al., 2005). Whereas the Riva TNT had a modest two texture units at its disposal, high end consumer hardware like AMD's Radeon 9670 family of GPUs have as many as 96 texture units (AMD, 2010) as multitexturing has become a cornerstone of many techniques.

As terrains in modern computer games measure in the kilometres, the tessellation of the mesh is too crude to convey detail for any given material so texture mapping is crucial to add detail to differentiate the different surface types that make up the terrain. The use of multitexturing in real-time terrain rendering allowed techniques such as texture splatting (Bloom, 2000) to be performed more efficiently, allowing for a greater diversity of materials to be applied to the mesh for more realistic looking landscapes. When combined with programmable GPU hardware, multitexturing can allow for a wide array of terrain texturing techniques, such as procedural shader splatting (Andersson, 2007) and virtual texturing (Mittring and GmbH, 2008)) as well as greatly increasing the surface detail with normal mapping and parallax effects.

In the natural world, when we observe an object we are in fact are seeing our brain's interpretation of the visible wavelengths of the electromagnetic spectrum (Schacter et al., 2010, p. 134). Our eyes perceive colour through some 6 million cone cells evenly which are densely packed in the centre and sparsely distributed over the rest of the retina (Boff et al., 1986; Schacter et al., 2010, p. 136). These cone cells allow us to perceive some 10 million colours (Judd, 1953, p. 338) and luminescence with a dynamic range of 5 to 9 orders of magnitude (Larson et al., 1997), far exceeding the dynamic range of 256:1 for 24 bit RGB colour.

The colours of an object we perceive is the complex interaction of light as the photons bounce and trade energy with other objects in the scene. In the natural world, without light we cannot see but in computer generated imagery the absence of light equates to a uniform illumination of the scene. Without light in computer generated scenes we lose our sense of depth perception as the interaction between an object and light sources provides information about the 3D shape of the surface (O'Shea et al., 2010). Lighting (Immel et al., 1986; Kajiya, 1986) in particular has been the focus of much research over the years due to the inherent complexity of the phenomenon, from modelling basic interaction between photons and surface material (Blinn, 1978b; Cook, 1984; Newell et al., 1974; Nishita et al., 1985; Phong, 1975; Warn, 1983) to complex interactions and cumulative contributions from multiple light sources (Goral et al., 1984; Guitton et al., 1995; Immel et al., 1986; Kajiya, 1986; Ritschel et al., 2008; Teller et al., 1994).

Terrain lighting can adopt the usual illumination and shadowing techniques but can be simplified when using a heightmap. For example, calculating ambient occlusion, slope lighting, shadow maps (Marghidanu, 2002) and horizon maps (Stewart, 1998) offline is a relatively now trivial matter of tracing rays across the displacement texture compared to the complex ray tracing and intersection testing of arbitrary geometry. However, for scenes that have a day and night cycle such offline techniques would be inappropriate so instead real-time techniques must be utilized where possible. Techniques such as atmospheric scattering (Bruneton and Neyret, 2008; Hoffman and Preetham, 2003; Pegoraro et al., 2011) and dynamic shadowing (Dimitrov, 2007; Ma et al., 2009; Snydre and Nowrouzezahrai, 2008; Zhang et al., 2006) can be used to great effect to enhance the mood and realism of outdoor scenes by approximating this natural light cycle.

1.1 Focus of Thesis

The size and detail of the natural landscapes makes it a particularly challenging are of research for real-time computer graphics. Although the power of consumer graphics hardware continues to increase at an exponential rate we are still far from being able to reproduce every nuance and detail of the natural world in real-time. As such, the modelling, texturing, lighting and shadowing of computer-generated terrains are but approximations of their real-world counterparts. More powerful hardware has allowed closer algorithmic approximations to reality but the vastness and endless variation of the natural world means that at every turn it is a matter of utilizing all of the latest tricks and techniques of cutting edge computer graphics to deliver the best results to the end user.

One technique in particular that is essential to terrain rendering is texture mapping. It would be impossible to model every detail of the natural world using geometry alone so, instead, texture maps are used to add detail and definition to compensate for this lack of tessellation. The location where two or more materials meet is called the *transition* where the texture maps of said materials gradient from one into the other. The detail and realism of these transitions is important to suspend the user's disbelief as an unnatural, unrealistic transition band will quickly destroy the illusion of detail and complexity.

The focus of the research presented in this thesis is on the synthesis of transitions between differing material textures. As the surface of a terrain is often covered with a number of materials, the modelling of these interactions between differing materials is an important aspect of realistically depicting natural landscapes. Particular focus will be on the challenges of synthesizing such transitions in real-time through an automated process.

A common technique for transition synthesis is texture splatting (Bloom, 2000), where a greyscale alpha mask texture for each material is mapped across the terrain to dictate the translucency of the material at any given point. As said alpha mask textures are of low resolution comparative to the size of the terrain, they are capable of depicting gradual transitions but lack the fidelity to represent sharp, detailed transitions between materials. Derivative techniques (such as blend maps (Hardy and Mc Roberts, 2006) and procedural shader splatting (Andersson, 2007)) instead use complementary textures or algorithms to add high frequency detail to these otherwise low frequency transitions.

This additive process of modulating low resolution alpha masks with high resolution detail textures greatly improves the otherwise vague and blurry transitions that can result from using alpha masks alone, though they fail to address two key shortcomings of texture splatting. Firstly, the topology of the materials themselves are not taken into account when synthesizing the transitions. Prominent features that should protrude through any underlying materials instead fade into full translucency as their opacity follows the gradient of the alpha mask texture. Secondly, while the added high frequency information greatly increases the detail of the transitions when viewed up close, the broader contour of the transition remains untouched, resulting little variation to the contour shape due to the low frequency detail of the alpha mask texture.

Overcoming these issues would improve the realism of material transitions. A topographically-aware algorithm would ensure that features protruding through underlying materials would eliminate the translucency artefacts that can occur with texture splatting. Sometimes this gradual fading of features into underlying terrain is desirable (for example, sand or dust, where partial coverage of features is likely) but for many underlay materials (for example, grass), it would be more desirable for the material to instead gradually transition in the low topographical regions of the material (such as in the cracks or mortar spaces between features) but leave the features intact. Likewise, the contour of transitions in the real world are limitless in their detail and variation, something that is orthogonal to the smooth, unvaried gradients of texture splatting. By breaking up these low frequency contours with more stochastic and higher frequency details, more varied and realistic transitions can be produced.

This thesis presents two novel approaches to address these issues with texture splatting. Firstly, Feature-Based Probabilistic Blending synthesizes transitions with awareness of topography, allowing for prominent features (such as stones and cobbles) to protrude through the underlying surface materials. Secondly, Transition Contour Synthesis With Dynamic Patch Transitions modulates the contour of the transition to introduce intermittency and variation in the otherwise low frequency contour.

1.2 Organization of Thesis

Chapter 2 outlines the problem domain of terrain texturing and the challenges it presents for transition synthesis along with a set of criteria to evaluate the research presented in the chapter. The research is broken down into three broad sections: alpha blending, tile-based rendering and virtual texturing. Finally, the conclusion of the chapter summarizes the research and presents a direction for future research.

Chapter 3 presents the challenges of transitioning between materials featuring salient details that contribute significantly to the topography of the texture before assessing the existing research with respect to addressing these challenges. The technique “Feature Based Probabilistic Blending” is presented as a technique for transitioning with materials containing salient features (such as cobbles and bricks) whilst adding dynamic wear and tear to the features in real-time.

Chapter 4 presents the challenges of generating detailed and varied transition contours using alpha blending and derivative techniques. The technique “Transition Contour Synthesis with Dynamic Patch Transitions” is presented as a technique for modulating the contour of material transitions to add detail and complexity to the transition.

Finally, chapter 5 summarizes the development of transition synthesis along with a summary of the contributions presented in this thesis.

1.3 Summary of Original Contributions

Feature-Based Probabilistic Blending introduces intermittency and irregularity at transitions for materials that contain distinct features. It achieves this by ensuring that all texels belonging to a given feature are drawn with full opacity or full transparency on a probabilistic basis by using the alpha mask weighting from texture splatting as the probability of said features appearing. Further detail and variation is introduced through dynamic wear and tear, modulating the shape and appearance of features to help break up the repetitive nature of the tiled texture maps used in texture splatting. The performance and memory overhead of Feature-Based Probabilistic Blending is competitive with alternative techniques (such as blend maps) yet adds greater detail and variation to the terrain transitions.

Transition Contour Synthesis with Dynamic Patch Transitions generates irregular blend contours of near-endless variation for materials that do not contain distinct features. This is achieved by dividing texture space into grids where each cell is drawn with full opacity or full translucency on a probabilistic basis before being bilinearly interpolated with surrounding cells to produce varied and detailed contours from the low resolution alpha mask. This process is entirely automated and requires no additional assets or preprocessing and offers performance competitive to alternative techniques, allowing it to be integrated into existing texture splatting implementations seamlessly.

Table 1.1 Summary of publications.

Title	**Type**	**Journal/Conference**	**Year**
Feature-Based Probability Blending	Poster	SIGGRAPH Asia	2010
Feature-Based Probabilistic Texture Blending with Feature Variations for Terrains	Full Paper	Computer Animation & Virtual Worlds	2012
Automatic Terrain Texturing with Dynamic Patch Transitions	Short Paper	Computer Graphics International	2013
Transition Contour Synthesis with Dynamic Patch Transitions	Full Paper	Computers in Entertainment (accepted and unpublished)	2014

Chapter 2

Literature Review

2.1 Chapter Overview

This chapter surveys the research pertaining to the challenges of mapping surface textures to terrain meshes in real-time. These surface textures are defined as the texture representing the albedo surface colour at any given point on the mesh. Such surface textures are composited from different materials (such as grass, sand, rubble etc.) in order to authentically reproduce the diversity and endless detail of the real world. As capturing these qualities effectively is an important part of any terrain texturing technique, particular focus is placed on the depiction of transitions where differing materials meet.

2.2 Background Information

The very nature of terrains makes them particularly difficult to model and display in detail and thus presents a series of technological challenges for real-time rendering. If one was to assume the earth is a perfect sphere, a person measuring 2 meters in height standing at sea level would be able to see the horizon approximately 5.1km from their view position. If this person's perspective was modelled with a virtual camera with a field of view of 90 degrees, the far plane would measure approximately 7.2 km in width and the view frustum would cover an area of approximately 13 km^2 (Plait, 2009). For flight simulators, this challenge of draw distance is further compounded. For an aircraft traveling at an altitude of 1km, the horizon is approximately 113 km away with the frustum covering nearly 6,400 km^2 (Plait, 2009). Thus, the first challenge of real-time terrain rendering is the sheer expansiveness of real-world landscapes.

The interior of a typical building is relatively simple in its geometric structure. The floors, walls and ceilings can be approximated with planes and the interior decorations with rectangular prisms, cylinders, spheres and other meshes of crude tessellation. Terrains, however, are fractal in nature and thus there is effectively no end to the complexity of their geometry. In Mandelbrot's landmark paper he argues that geographical curves are so involved in their detail that their lengths are often infinite or, more accurately, undefinable (Mandelbrot, 1967). Thus, the second challenge of terrain rendering is approximating this endless detail.

Consider this building further. Any room or corridor will follow the same design theme in terms of shape and regularity. The floor may be of the same material and/or pattern, the walls painted or wallpapered in the same manner and the ceiling may be plastered and stylized consistently throughout the building. To model these details in a computer generated scene one need only model the core stylistic themes of the building and repeat these throughout the environment. Landscapes, however, are stochastic in nature so repetitive patterns are unusual. Even for a landscape populated nearly exclusively with one material such as the shingle on a beach, each pebble of the many millions will be unique in size, shape and texture. Thus, the third challenge is rendering landscapes with this illusion of endless irregularity and diversity.

Modelling the complex features of terrain with geometry alone is currently not feasible for terrains of non-trivial size. To render a terrain on a 2 megapixel display with a maximum geometric screen space error below one pixel some 30 million triangles would be required (Dick et al., 2009a,b). Although features such as tessellation stages of modern GPUs can be used to increase the geometric detail at run-time (Microsoft, 2012; OpenGL Specification, 2012), modelling even a 1km by 1km terrain down to nearest half centimetre would take some 40 billion vertices, nearly 80 billion triangles and nearly 240 billion indices. If we were to assume that each vertex is composed of a position, normal and colour vectors of three floating point values, this would total to over 3.3 TB of vertex and index data, far beyond the capabilities of existing consumer graphics hardware. Instead, the general topography is modelled with a comparatively lower tessellation mesh and the higher frequency details of the surface and topography are approximated using texture mapping (Azariadis and Aspragathos, 2000; Heckbert, 1986, 1989; Norton et al., 1982).

However, trading geometric tessellation for texture maps presents a similar challenge as the size of the texture map needed to convey a convincing approximation of a large landscape surface is also very large. A 512x512 texture map covering a 2m^2 region of the mesh offers a resolution of 256 texels per meter for a texel size of approximately 0.4

cm so our aforementioned terrain mesh would require a texture map measuring 256,000 by 256,000 texels. Assuming 32 bits per channel, this texture would require around 197 gigabytes of storage space, further reduced to about 49 GB using DXT compression (Intel, 2012a). This is considerably less than the geometry required in our previous example and is certainly within the range for consumer storage but as a typical GPU at the time of writing such as Nvidia's GeForce GTX 660 has merely 2 GB of memory (nVidia, 2012) it becomes clear that naive texture mapping is insufficient to meet the challenges of terrain rendering.

Texture synthesis (Perlin, 1985; Rhoades et al., 1992; Wu and Yu, 2004; Zelinka and Garland, 2004) is a technique for generating large textures from an algorithm or smaller example textures. The advantages of this technique are that it is far more compact than storing and using the larger texture directly so it is of no surprise that such techniques have been used throughout the history of terrain rendering. The simplest form of texture synthesis is texture tiling, where a small, tileable example texture is repeated across a plane or surface to create the illusion of one large texture. However, terrains are rarely uniform in their surface composition but one way the problem can be addressed is by breaking down the surface into discrete materials (such as grass, sand, rock, rubble and so on). These materials can then be mapped independently to the terrain to create the illusion of complexity and detail on the surface. However, the trade-off is the inherent repetitive nature of texture tiling that is at odds with the endless irregularity of terrain.

A single surface texture painted by hand can produce terrains that are both detailed and diverse in their surface properties. This texture would then be sampled on the GPU in real-time to produce convincing, realistic landscapes limited only by artistic skill and texture resolution. Modern games set in immersive, outdoor environments can span many kilometres in each direction so generating the texture assets is both time consuming and highly skilled work. Thus, for practical purposes a trade-off is usually made between the artistic freedom of freehand texture painting and automating such tasks as much as possible with procedural methods in order to deliver within the time constraints of the game development schedule.

The rest of this chapter will assess existing approaches for producing large, detailed and varied textures for terrains. Of particular focus will be the transitions where different surface materials meet. Such transitions particularly encapsulate the challenge of detail and irregularity since they are invariably stochastic in nature as how the different materials interact with one and other is a complex interplay that must be faithfully reproduced if authenticity is to be achieved. For example, sand transitioning into pebbles will look different from sand transitioning into grass and grass transitioning into pebbles will look different from grass transitioning into mud. This bleeding of material boundaries into one and other are integral to suspending the user's disbelief in order to successfully immerse them in the computer generated environment.

2.3 Technique Categorization

Discussed in this chapter are three categories of terrain texturing and transition synthesis: texture splatting, tile-based texture mapping and virtual texturing. As each technique may incorporate elements of more than one category, they are placed in the category that predominantly defines their design. The categories themselves are defined as follows:

Texture Splatting: Techniques that composite a larger surface texture by tiling and layering a discrete set of smaller material textures. Each material has an alpha mask that determines the opacity of that material at any given point on the terrain mesh.

Tile-Based Texture Mapping: Techniques that synthesize and/or arrange a discrete set of pre-generated, connectible tiles according to an algorithm or placement map to composite a larger surface texture. Each tile depicts variations of a given material and/or transitions between two or more materials.

Virtual Texturing: Techniques that stream the portions of a pre-generated surface texture required by a given render frame on-demand as the camera moves around the 3D environment. This surface texture is usually generated by artists rather than an algorithmic process.

2.4 Evaluation Criteria

The strengths and weaknesses of each category of techniques are evaluated according to the criteria described below. Whilst each technique in a category may be designed to address specific needs, each category will be evaluated as a whole although the typical use of each technique will also be discussed.

Performance and Scalability: The performance of a category is assessed in terms of the hardware targeted by the techniques, as well as upstream to higher specification hardware. Some techniques will not run on older hardware so performance on such hardware is meaningless, therefore these initial hardware requirements will be covered in the Fixed Costs. In addition to raw performance, how well a given category of techniques performs with and depicts terrains large and small will be assessed as the scalability of that category.

Fixed Costs: The techniques in a given category have a number of fixed costs that must be met in order to be successfully implemented. These costs include the computing resources needed by the algorithm (such as hardware, memory and storage space) along with any offline pre-processing of data (such as texture masks, material data and transition synthesis). As the performance of each category will be assessed in terms of the algorithm's target hardware, the fixed costs offer a useful metric for measuring how well techniques will perform on a variety of hardware as well as measuring the amount of offline work needed in preparation for any given technique.

Transition Variation and Detail: The focus of the entire project will be on evaluating the detail and variation of material transitions. The detail can be loosely broken into two components: low frequency detail (the detail of the overall shape and contour of the transition) and high frequency detail (the finer details of a transition, usually at the texel level). Real-world material transitions are rarely regular and uniform so how well a technique can depict these stochastic and diverse transitions is evaluated in conjunction with the overall fidelity of a technique.

2.5 Texture Splatting

Technological constraints are still the limiting factor for real-time photo-realistic rendering so research has been focused on achieving the maximum level of detail and realism from the finite resources of the available hardware. Perhaps the simplest form of synthesizing large textures is texture tiling, a technique where a larger texture is composited by repeating a smaller sample image periodically over a plane (Dungan et al., 1978). This technique is so primitive and ubiquitous that it is often the default setting of graphics APIs for texture sampling (Akenine-Möller et al., 2008, pp. 154).

The problem with simple texture tiling is that it is impractical to map multiple materials to the terrain mesh without hard, abrupt transitions defined along the edges of the geometry where the vertices reside. Bloom (2000) proposed a method called 'texture splatting' for compositing terrain surfaces in the frame buffer by using an additional alpha mask for each material to dictate the translucency of a given material at any given point on the terrain surface (Figure 2.1). The materials themselves are a set of example textures that are tiled across the terrain using the alpha masks to control the translucency of the materials by using the mask values as the t value in a linear blend. This technique is often extended to use low resolution samples for the low frequency detail and further blend high resolution detail textures (typically gray scale images) to add high frequency detail in regions that are close to the viewer.

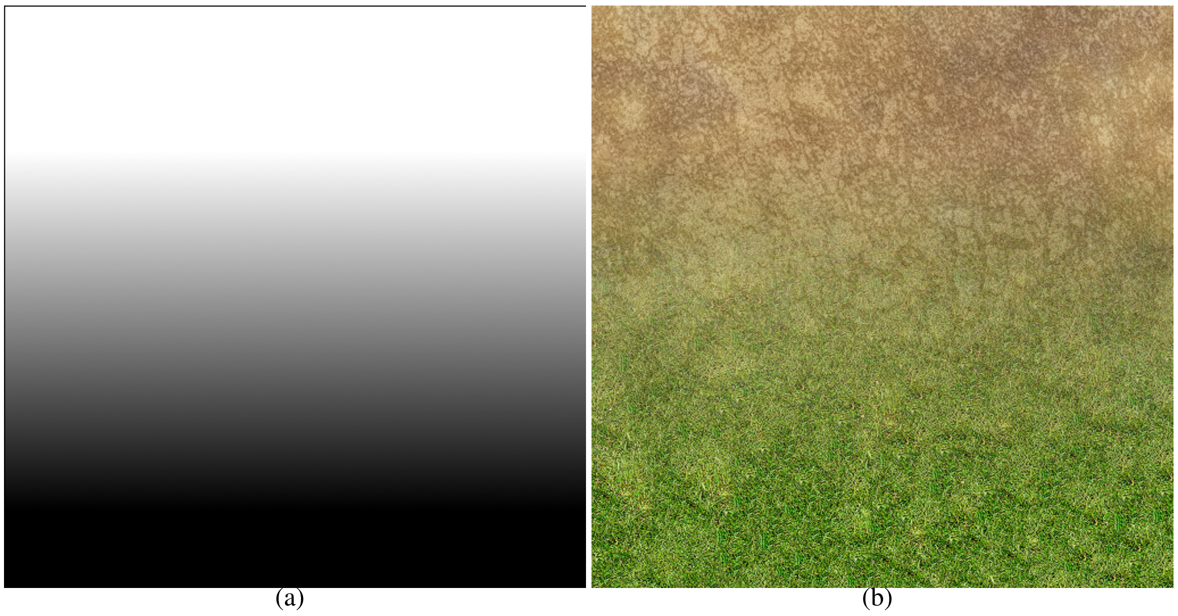


Figure. 2.1 Texture Splatting weight texture (a) and generated transition as per the Bloom algorithm (Bloom, 2000) (b).

In Figure 2.1, we can see an example alpha mask depicted as a greyscale image (a). This alpha mask is used to determine the translucency of the dirt texture, starting with full opacity at the top of the image and transitioning into full translucency at the bottom. By enabling blending in the graphics API and using the value from the alpha mask as the blend alpha of the dirt texture we can see that the dirt texture is blended over the grass texture to give the illusion of a transition between these two materials.

2.5.1 Implementation

At the core of texture splatting is the linear interpolation between two materials: the material currently being blended (variable b in Equation 2.1 below) and the material (or sum of materials) preceding it (a in the equation below), with the alpha value t determining the opacity of material b. This in turn determines the surface value (typically albedo or normal) s at that point on the terrain. The value t is in the range $[0,1]$ with values of 1.0 being full opacity and 0.0 being full translucency. However, there is the potential for the sum of t values to total up to less than 1.0, revealing whatever resides in the frame buffer before splatting takes place. There are two options to deal with this issue, either ensure that at any point in the terrain the alpha values sum to 1.0 or, alternatively, designate a neutral base surface texture that is mapped across the terrain as a pre-processing step. The latter is favoured for its simplicity and the fact that this step can be achieved effectively for free as part of an early Z pre-pass of the terrain.

$$s = a \cdot (1.0 - t) + b \cdot t \tag{2.1}$$

In Bloom's original proposal, the alpha mask took the form of a texture that was generated offline by artists or an algorithmic process. The resolution of this texture was frequently restricted by the hardware restrictions of the time. Bloom suggested a resolution of 2x2 texels for a triangle pair quad so these alpha textures were very low resolution but modern implementations commonly eschew the use of dedicated weight textures and instead use shaders to algorithmically determine the alpha masks on the fly in real-time. Historically, it was often more practical to perform the technique using multiple rendering passes as this was often cheaper than using multi-texturing which, at the time, was still in its infancy. With modern hardware, the relative cost of multi-texturing has come down making it possible to map an entire set of materials to a mesh within a single rendering pass.

2.5.2 Applications

Although largely superseded by more modern techniques where cutting-edge realism is required, texture splatting remains popular for seamless texture compositing due to its efficiency and simplicity (Catanese et al., 2011; Intel, 2012b), allowing artists to 'paint' terrains in an intuitive manner with interactive tools (Crytek, 2011b; Epic Games, 2009a) or off-the-shelf image editing software. Still, the resolution of the alpha mask is insufficient to depict blends with per-texel accuracy resulting in vague, undefined transitions. Whilst this is adequate for materials lacking in high frequency detail, materials containing such detail are depicted with gradual transitions in translucency.

Hardy and Mc Roberts (2006) address the feature agnosticism of linear blending by modulating the low frequency blend weight with a material's high-frequency blend map to produce higher resolution blends. This blend map is generated by the artist (or algorithmically) to produce a map that governs the strength of bias used in the blending equation. The technique adds fidelity at material transitions and helps reduce the translucency artefacts for material with salient features by biasing the alpha mask in favour of such details (Figure 2.2).

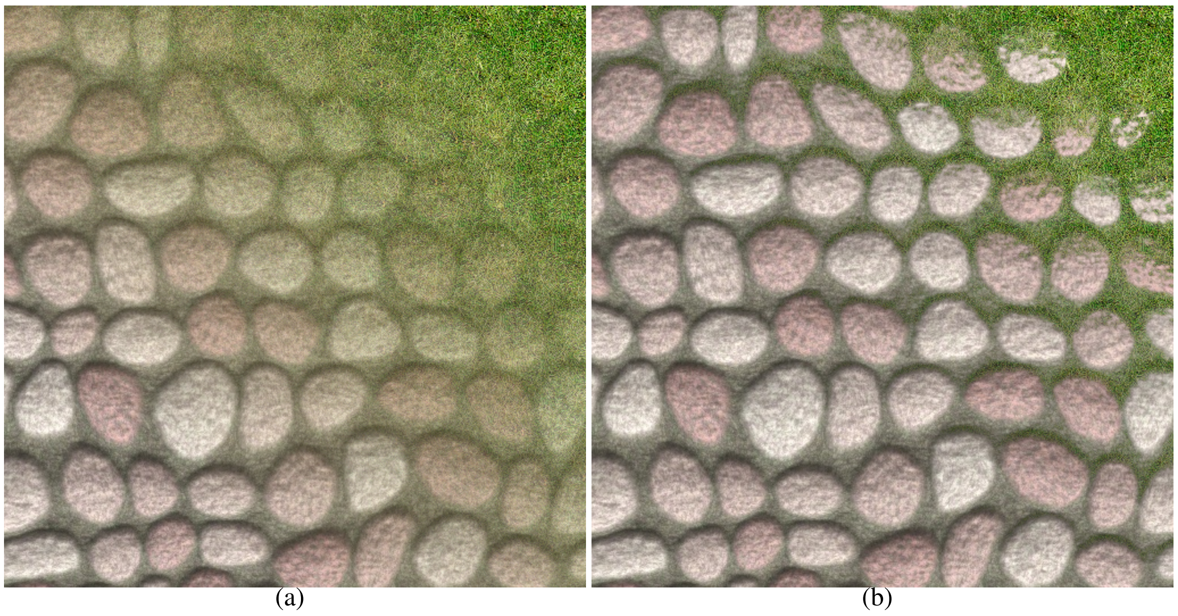


Figure. 2.2 Linear blending (a) and blend maps as per the blend maps algorithm (Hardy and Mc Roberts, 2006) (b).

This simple, computationally cheap technique is frequently used in video games to add high frequency detail to texture splatting, although, whilst it reduces the translucency artefacts for prominent features it cannot eliminate them entirely. As the low frequency detail is accommodated by texture splatting, the broader contour of the blend becomes apparent with low resolution alpha masks. However, as the blend map meta-data is stored with the material itself (typically in the alpha channel of the material's albedo texture), the technique takes up little or no additional video memory and is unaffected by the terrain size.

Texture splatting itself has evolved in many shapes and forms, such as the procedural shader splatting technique developed by Dice and used in their Frostbite 2 game engine (Andersson, 2007). Rather than using a low resolution alpha mask to determine the blend weight at any given point on a surface, the height, slope and normal information of that point are used to composite an alpha mask for each material in the fragment shader (Figure 2.3). This allows for far more detailed transitions between materials whilst avoiding the square law increase in video memory needed for the alpha mask textures as the terrain size increases.

Procedural shader splatting automates the task of synthesizing the alpha masks for each material type so in principle it can scale for terrains of any size. The terrain materials

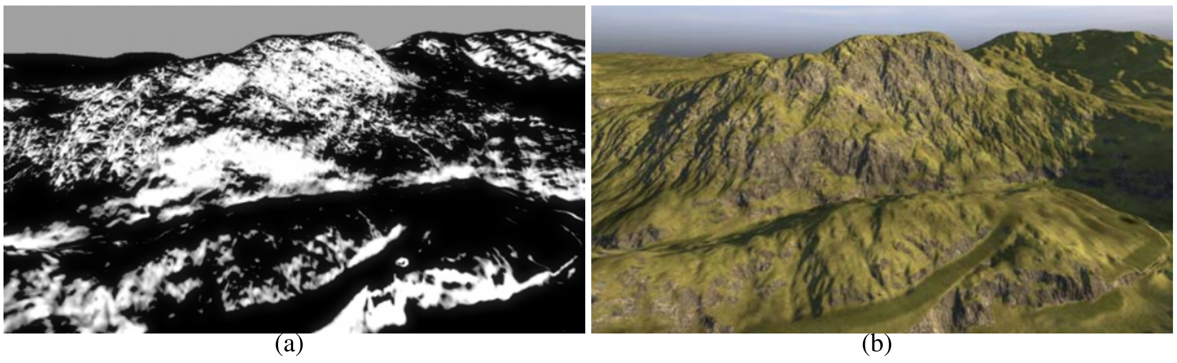


Figure. 2.3 Procedural Texture Splatting (Andersson, 2007).

themselves need no prior processing to integrate with the algorithm and their intuitive artist-driven shader composer makes material generation simple and easy to use for non-programmers. However, the resolution of the blend masks becomes coupled with the resolution of the terrain geometry so noise is added to increase fidelity and intermittency at material transitions. Areas of the terrain that contain little or no height or slope information will still require conventional alpha masks if multiple materials are desired.

Zhang et al. (2008) synthesize transitions between materials by analysing the elevation model of a terrain mesh (real or synthetic) and using it to construct a feature mask that determines which materials will be applied where on the surface of the mesh (Figure 2.4). Although such an approach reduces the time required for an artist to texture a terrain and can adopt material distribution on parts of the terrain not recognized by simple height/slope analysis, the transitions suffer from the same translucency artefacts of feature-agnostic linear blending (Figure 2.4(a)).

The utilization of albedo, relief and other pertinent data obtained from satellite photography to construct interactive 3D visualizations of real-world landscapes has practical uses for planning and simulation purposes. A common challenge is to compensate for shortcomings of the photography process such as visual discontinuities and limited/incomplete data. Typically, the albedo imagery is of a resolution too low to convey detail when the viewer is at ground level so Roupé and Johansson (2009) propose a technique for adding high-frequency detail to the low-frequency satellite imagery to help with urban planning visualizations. They achieve this by synthesizing colour-coded masks that are laid over the satellite imagery to dictate where a given high-frequency detail texture will be mapped (Figure 2.5). By using real-world data from city planning authorities they manage to reduce the artistic workload, allowing integration with the satellite imagery with minimum manual input. This process of blending high frequency detail on to low frequency imagery delivers

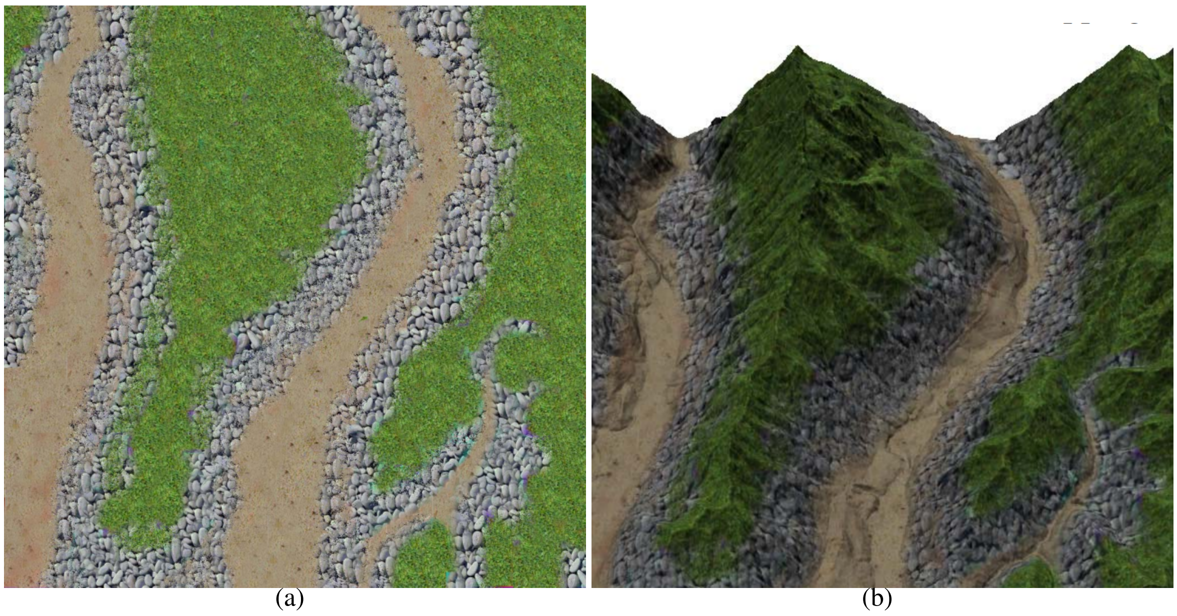


Figure. 2.4 Texture Synthesis Based on Terrain Feature Recognition (Zhang et al., 2008).

sharp, distinct surface materials when using imagery as low resolution as 5 pixels per meter.

Whilst the use of real-world imagery is suitable for civil and military visualizations where large, real-world data sets need to be processed, the technique's inherent inflexibility for artistic control makes it less suitable for procedurally or artistically generated terrain assets. Computer games typically create entirely new worlds with much emphasis of artistic aesthetic to create atmospheric and immersive environments so the techniques for surface texture compositing using real-world data will be of limited use.

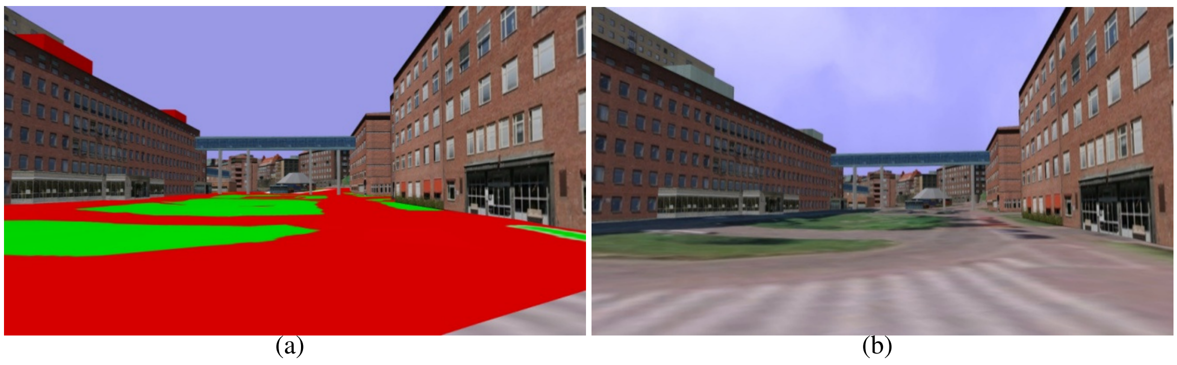


Figure. 2.5 Visual Quality of the Ground in 3D Models (Roupé and Johansson, 2009).

2.5.3 Evaluation

Whilst publicly posited by Bloom in 2000, he remarks that texture splatting had found use in commercial use in games prior to this. As such, it has a comparatively long history compared to other approaches. Whilst the results are potentially less visually pleasing compared to more modern techniques, its efficiency and ease of implementation have prolonged its shelf life, even though it has been steadily surpassed by partial resident texturing techniques in cutting edge video games.

Performance and Scalability: As a technique with over a decade of use, it it unsurprising that texture splatting can be adopted on pretty much any GPU hardware on the consumer market. Even legacy, fixed function hardware can utilize this technique through alpha blending rather than shaders, as described in Bloom's original report. Mobile devices in particular can benefit from this technique due to its low consumption of computational resources, although as mobile computing continues to increase in power this may be one of the last hold outs for texture splatting in its purest form as a viable solution to high performance, cutting edge terrain rendering. Instead, hybrid techniques could leverage the automation of procedural texture splatting techniques to lay down a template to be further processed by artists to reduce the amount of manual labour for surface texture composition.

As the computational costs of the technique are low and constant, texture splatting can be scaled up to theoretically any size terrain, making it suitable for terrains both small and large. However, if the original alpha mask implementation of texture splatting is to be used, the square law increase of the size of these alpha masks may well render the performance and fixed costs benefits of this technique moot, although this is less of an issue with procedural techniques such as Procedural Shader Splatting (Andersson, 2007) where the blend weights are calculated in real-time from geometry data.

For terrains without much surface variance, the number of materials needed for varied and convincing results can be modest but for larger terrains a more diverse set of example materials may be required to convincingly depict the wide variety of surface materials found in the natural world. As the number of texture lookups increases with the number of materials, this could cause the application to become prematurely fill-rate bound when an excessive number of materials is used. For modern GPUs this should not present much of an issue but for older and/or embedded/mobile hardware with more restricted processing power such limitations should be taken into consideration.

Fixed Costs: By tiling and blending a number of example material textures, the fixed cost of texture splatting is considerably lower than using a single, detailed surface texture for the entire mesh. A common optimization is to use a low resolution albedo texture for low frequency detail and a high resolution detail texture for high frequency detail (Crytek, 2011b). Once the desired number of materials and blend resolution per meter has been decided, the fixed costs of texture splatting can be calculated accurately for any given terrain. As the blend maps are stored in the material's alpha channel, the fixed costs of this particular technique effectively comes for free.

The use of alpha mask textures does present issues in terms of increasing texture memory required for larger terrains as the size of the alpha mask texture is directly proportional to the size of the mesh and the desired blend resolution. However, shader-based implementations can alleviate this fixed cost entirely by deriving the blend weights procedurally in the fragment shader. Of course, this does increase the fixed costs of the GPU hardware itself but with the proliferation of programmable consumer cards this should not be an issue on modern hardware.

For the artist, one need only create an example albedo/normal texture for each material and the generation of the alpha mask can be achieved with any off-the-shelf image editing software (or even interactively within the game engine itself) and/or generated procedurally from parameters such as height, slope and normal information. This means that the number of man hours required to produce art assets for even very large terrains can be kept to a minimum as the same set of materials can be applied to any terrain without any additional pre-processing. In principle, by using procedural techniques to generate the terrain meshes and the blend weighs, there is no additional artistic cost to adapt a given set of materials to any terrain. As the only pre-processing requirement for material textures is that they be tileable, swapping materials in and out of the terrain's set incur no additional labour or offline processing penalties, making experimentation both rapid and effortless.

Transition Variation and Detail: As with any approach to compositing surface textures from tiled example textures, the repetitive effect of the tiling process will be prominent for areas of sparse, uniformly textured terrain. This effect is somewhat minimized by the use of multiple materials to break up such parts of the terrain and can be further reduced by populating the surface with vegetation such as shrubbery, bush and grass. The main drawback of texture splatting is the low resolution of transition blends due to the low resolution of the alpha masks. This can lead to vague, washed out transitions that are not suited for materials containing salient details. This effect is due to the simple linear blend used to composite the transitions although this can be alleviated by using blend maps or noise to help break up the transition and add high frequency fidelity.

Materials containing salient features suffer from noticeable translucency artefacts due to blind interpolation between materials. As the blending process is unaware of the topography of the material, prominent details that protrude through the surface can be truncated by the blend contour causing such features to fade into translucency in an unrealistic manner. This effect is particularly evident in transitions that fade gradually or sharply from one material to another. Blend maps offers a computationally cheap way of reducing these artefacts by weighting texels belonging to such features such that they are sustained for longer but the effect is not eliminated entirely.

There is little variation in the transitions texture splatting can produce so large swathes of terrain transitions can produce uniform, periodic transitions in areas containing little variation in weighting. This is largely due to the blending taking place at the fragment level so higher order details such as contour remain largely unaffected. Although the technique works well with synthetic and stylized textures, the inherent limitations of texture splatting become particularly noticeable with photo-realistic textures where the translucency artefacts are particularly obvious.

2.6 Tile-Based Texture Mapping

The images synthesized from repeating textures are inherently periodic. For man-made patterns such as wallpapers, brickwork and floor tiles this is not a great issue but this repetitive nature is at odds with many of the surfaces in the natural environment which instead tend to be endless in variation. Texture synthesis (Lewis, 1989; Perlin, 1985) has received a great deal of focus over the decades (particularly from the film industry), allowing materials of near-endless variation and uniqueness to be generated procedurally but few techniques directly address the issue of transition synthesis between terrain materials.

Tile-based texturing is one approach for introducing aperiodicity into textures by using a small selection of example tiles. These tiles are then assembled off-line or on-the-fly to produce the final aperiodic image. The individual tiles themselves must connect seamlessly with other tiles in the set to properly assemble into the final image so compatible tiles must share a common edge to avoid seams. Wang tiles (Berger, 1966; Wang, 1961) address this connectivity problem by defining a set of tiles with colour-coded edges to denote connectivity compatibility along the borders of tiles with matching edge colours. The number of tiles within a given set will determine the true variation of the tiling although Culik demonstrated that a strictly aperiodic tiling can be produced from as little as 13 tiles with 5 edge types (Culik, 1996).

2.6.1 Implementation

Although there are many approaches to transition synthesis using tiling, a common theme observed throughout the literature is the production of different material transition variations that are assembled to create the final aperiodic transition. These tiles may be generated by hand or through some algorithmic process depending on the scope and application of the technique. Prior to the advent of powerful consumer graphics hardware, these tiles were assembled offline but by using an indirection texture as a tile placement map (lookup table) the entire assembly process can take place on the GPU in real-time.

This placement map is typically an image that divides up the terrain's surface into regular squares where each texel represents a tile location. The colour of each texel can represent a variation of that given tile or simply a tile of appropriate connectivity, such that a regular 8 bit colour channel has the capacity to represent up to 255 tile connectivity types or variations (one value, typically zero, must be set aside to represent an empty tile). For four colour channel images, it is possible to encode up to four separate tile sets in the placement map by utilizing a colour channel for each tile set.

Inside the fragment shader, the fragment's texture coordinates are then transformed from terrain space into the coordinate space encompassing the dimensions of the placement map. The placement map can then be sampled with nearest neighbour filtering using the newly transformed texture coordinates to obtain the lookup value for the tile position upon which the fragment resides. The table entry can be used to pull a specific tile from the set from either an atlas texture or a texture array and sampled accordingly. As this form of texture synthesis is performed in real-time, mip-mapping will not be effective at reducing aliasing artefacts for distant tiles when the camera moves. Instead, this anti-aliasing must be performed as a post-step within the fragment shader.

2.6.2 Applications

Stam (1997) originally described the construction of a set of Wang tiles to be used for aperiodic texture mapping with Cohen et al. (2003) extending this work by proposing a technique for populating these tiles with image data and assembling them into the final image. Using Wang tiles for tile-based texturing has potential for generating a large number of texture transitions in real-time using graphics hardware (nVidia, 2004; Wei, 2004) although the number of tiles needed to accurately depict every transition junction with enough variations to break up monotony could be impractical for large terrains with many surface materials.

Wang tiles allow for aperiodic surfaces rendered from a compact set of tile shapes, with the offline generation of these tiles giving great control and flexibility to the artist in generating realistic transitions between materials. However, the level of aperiodicity of the transitions themselves across a large surface is ultimately determined by the number of tile variations generated for the set as the human brain is apt at picking out patterns in noisy environments. Although the algorithm used to assemble the tiles is not particularly computationally expensive, the size of the placement map increases with terrain size and can place a practical limit on the maximum terrain size addressed by the placement map on limited hardware for satisfactory levels of detail and aperiodicity.

Lai et al. (2005) propose a technique for synthesizing transitions between surface materials by identifying 16 fundamental transition shapes between any two given materials and synthesizing a tile texture for each shape (Figure 2.6(a)). This offline approach uses a patch based sampling method (Liang et al., 2001) that exploits the cellular automaton 'Game of Life' (Conway, 1970) to synthesize realistic, organic transitions between materials. The final surface texture is then composited by mapping the connectivity tiles to the succession patterns generated prior.

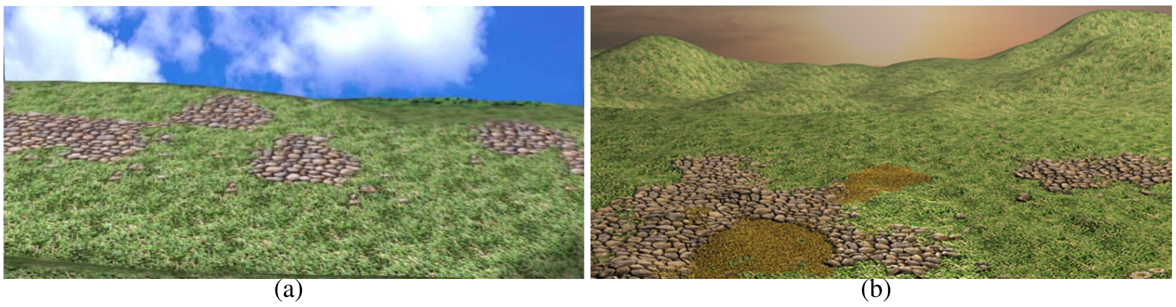


Figure. 2.6 Synthesizing Transition Textures on Succession Patterns (Lai et al., 2005) (a) and Transition Texture Synthesis (Lai and Tai, 2008) (b).

The resulting transition tiles are detailed and natural looking with few artefacts, even for terrain materials containing numerous salient details such as the pebble texture used throughout the paper. The tiles must be generated as an offline process and the number of tiles required to successfully synthesize each possible combination of transitions between materials is $16 \cdot (n-1)$, where n is the number of materials on a given terrain. For terrains with only a few materials this would not present much of a problem but for more complex surfaces the amount of video memory required to accommodate all of the possible transition tiles could become impractical on limited hardware.

Lai and Tai (2008) expand on the work of Lai et al. (2005) and implement the final surface texture synthesis in real-time using Wang tiles. Additionally, their work can synthesize transition tiles between more than two textures, greatly reducing the tiles sets to depict multiple terrain materials across a mesh surface (Figure 2.6(b)). For instance, in their paper they demonstrate their technique for transitioning between three and four materials on a given tile. As more material transitions can occur from a given tile set size, resources can be diverted to expanding the set size for more aperiodic results.

As with their prior work, the results are detailed and convincing, largely due to their transition cutting algorithm that takes into consideration the shape and placement of salient features such as pebbles, grass blades and flowers along the contours of transitions. The algorithm enjoys the relatively low overhead and decent performance of Wang tiles as well as the detail and intricacy of the work of Lai et al. (2005). As the process is almost entirely automated, the technique scales well to any terrain size. However, as with other tile-based techniques, a limited tile set size can become apparent on larger terrain surfaces.

While texture synthesis techniques offer exciting possibilities for generating terrain materials, only peripheral research within the field directly addresses the challenges of compositing a terrain texture from multiple, overlapping materials. Lefebvre and Neyret (2003) demonstrate a tile-based technique by compositing a texture from a set of 48 tiles and an 8 by 8 probability map to determine where surface materials and transitions should be (Figure 2.7). However, even with a dedicated set of 32 transition examples the broad shape of the probability map's contour was still evident. Furthermore, such a set of transition examples would be required for each material to ensure that every possible combination of material transitions can be reproduced.

2.6.3 Evaluation

The adoption of tile-based texture mapping for terrains has not been as widespread as other approaches discussed in this chapter. Whilst its use was ubiquitous during the 2D and isometric era of video games, the increasing power of GPUs and the shift towards programmable pipelines led way to techniques that synthesize the surface texture in real-time or stream it on demand from system memory or physical media. However, under the use cases where tile-based texturing is appropriate it does offer some distinct advantages.

Performance and Scalability: As the generation of the tile sets occurs through an off-line process, the assemblage of these tiles in real-time is comparatively cheap, requiring

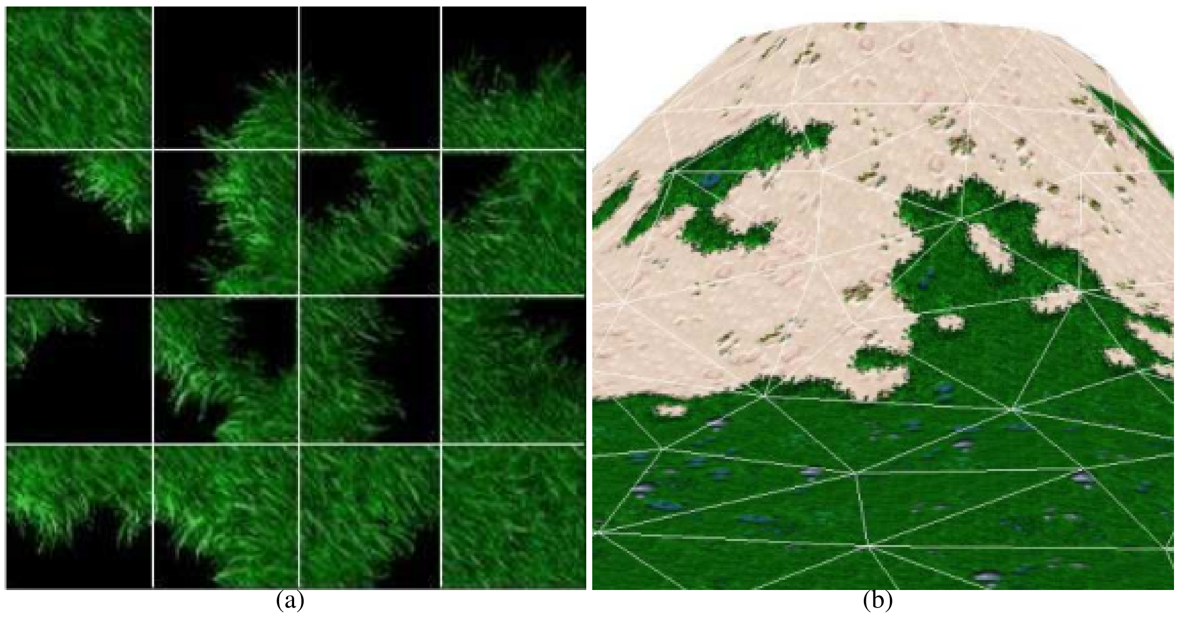


Figure. 2.7 Pattern-Based Procedural Textures (Lefebvre and Neyret, 2003).

only the sampling of a placement map on the GPU. As such, provided that the GPU can accommodate the tile set in video memory, tile-based texture mapping can run on a wide range of hardware. The run-time costs are low and constant and can even offer performance advantages over texture splatting as the tiles are rendered as-is without the need for layering and blending, freeing processing cycles for other duties. In principle, it can run on legacy fixed-function hardware by rendering regions of the mesh on a per-tile basis although the tile sets should be kept smaller in order to achieve acceptable performance.

Tile-based texture mapping can scale up and down to terrain meshes of various sizes but the number of tiles in a set may be insufficient to offer diversity at transitions. Larger tile sets can be used to circumvent this problem but as the tiles themselves depict the material transitions (as opposed to being synthesized in real-time like texture splatting), the number of tiles required for irregular and detailed surfaces on larger terrains can become a limiting factor. However, where small tile sets are used the overall performance benefit can make it more suitable for mobile devices and other limited architectures for producing detailed landscapes and transitions.

Unlike texture splatting, the tile assembly process does not lend itself too well to procedural techniques for tile placement, at least in real-time. In principle, one such implementation could use the height, slope and normal to deduce an appropriate tile and then perform the same calculation for surrounding tiles in tile space to deduce the required tile for connectivity. For tile variations, a noise texture could be sampled to obtain a random tile using a persistent seed such as the fragment's tile position in tile space. However, such an implementation would require at the very least this process to be performed five times to ensure that proper tie connectivity is maintained, negating the run-time performance benefits of tile-based texture mapping over other approaches.

Fixed Costs: The biggest fixed cost of tile-based texture mapping is the tile set. Each set must contain tiles not just representing the material themselves but also the transitions between each material. Lai and Tai (2008) demonstrate transition between up to four transitions on a single tile which can help slow the sets from inflating with material count but a number of tile variations would be needed to avoid too much repetition. For smaller terrains, this repetition may not be so noticeable but for large terrains with many material transitions it can become evident.

The placement map itself is akin to the alpha maps of texture splatting is not needed per-material. Rather, a single colour channel of a texture would be sufficient; that would be sampled as a lookup table in the fragment shader to pull the appropriate tile from the set. Unlike texture splatting, where the resolution of the alpha map determines the resolution of the low frequency blend, the placement map need only be of a sufficient resolution such that each texel represents a single tile unit on the terrain surface.

The addition and removal of materials from the set incurs a significantly larger cost compared to texture splatting as new transition tiles and variations must be generated to accommodate the new (or lack of) material. Whilst approaches such as Lai and Tai (2008) synthesize these transitions automatically, the removing of a material from the set could cause the transitions with that material to become obsolete, requiring a new set (sans the material) to be generated. Likewise, adding new materials requires another set of transitions to be generated so materials cannot be mixed and matched like texture splatting without the overhead of generating these new transition tiles.

Transition Variation and Detail: A surprising level of variation can be achieved from even small tile sets and the detail of the transitions is limited only by the resolution of the tiles and the competency of the artist/algorithm used to generate the tiles. As the tiles themselves are usually generated offline, the power of direct artistic control cannot be under estimated. As resource limitations are considerably less of a burden for offline processes, heavyweight algorithms that are incapable of running in real-time can be used to generate rich, detailed transitions, as exhibited by the work of Lai et al. (2005) and Lai and Tai

(2008). In particular, when used with real-world textures, the results are far more accurate and authentic than texture splatting as translucency artefacts are eliminated entirely by the artist/algorithm generating the transition. This is particularly evident for materials containing salient features as the topography of the texture can be taken into account when creating the transitions.

The drawback of using tile sets for terrain rendering is that the variety of the transitions is limited by the number of tiles in the set. As the number of tiles in the set increases, so does the required storage and video memory so terrains utilizing large tile sets featuring many transitions and variations may experience diminishing returns in terms of performance as the texture data required inflates accordingly.

As tile connectivity must be maintained, this places constraints on the artist or algorithm as completely arbitrary tile generation will exhibit undesirable seams where incompatible edges meet. By using tiles uniformly sized and laid on a grid, the transitions noticeably share the same "flavour" in terms of contour shape and variation when used over large areas of terrain due to this need for regular shape and connectivity.

2.7 Virtual Texturing

The idea of circumventing the hardware limitations of available texture memory is not a new concept entirely as texture streaming is a common technique for on-demand loading of texture resources as and when the scene requires them. For large immersive 3D environments, a significant portion of the texture resources are not required for any given frame, thus those which are unneeded may be swapped out of memory in place of those that are. Mipmap chains in their entirety may be streamed or merely the pertinent mipmap levels required by the frame to further increase memory savings, although the latter strategy has a less stable performance due to the increased frequency of stalls (Mayer, 2010; Mittring and GmbH, 2008).

Different strategies of streaming and caching may be taken depending on the requirements of the application but for terrain rendering the surface texture can be subdivided into regions of sizes supported by the graphics hardware that are loaded or unloaded as the player enters of leaves sections of the terrain. The main issue with this strategy is that the terrain (and other geometry) must be subdivided to these regions as the geometry must not overlap into multiple regions and that one must be careful to avoid visual "popping" of geometry and texture tile data as they are loaded and unloaded on-demand. These issues can further increase the complexity of texture streaming implementations due to the knock on effect they have on related areas such as geometric level of detail management.

Whereas both texture splatting and tile-based texturing work within the texture size restrictions of hardware to produce results that appear to exceed these restrictions in terms of size and detail, virtual texturing (Mittring and GmbH, 2008; van Waveren, 2009) instead tackles the problem from the opposite angle by attempting to circumvent these restrictions all together. The technique is a refinement of texture streaming approaches and takes advantage of modern hardware to offer a system that streams texture data on-demand smoothly and consistently. By utilizing intermediate texture coordinate spaces and indirection textures to stream from cached resources, the result is a system that operates transparently to the artists, allowing them to compose textures of any size for maximum control over the end result.

2.7.1 Implementation

Virtual texturing is a technique that is loosely based on the concept of virtual memory where texture address space is divided into chunks called pages (Cornel, 2012; Hollemeersch et al., 2010; Lefebvre et al., 2004). By subdividing a large texture (far larger than the amount of video memory available) into such chunks, only those that are visible (the "working set") are required to be resident in memory. Similar to the concept of clipmaps (Tanner et al., 1998), only the mipmap levels required by the scene are streamed into memory to further reduce the memory and bandwidth cost of updating the virtual texture (Mayer, 2010). By loading the lower mipmap levels initially, the renderer can fall back to these lower resolution mipmap levels if the higher levels cannot be streamed in time for a given frame. This helps to keep the frame rate consistent across a wider range of hardware and whilst the popping from lower to higher mipmap levels can be noticeable under certain conditions, the effect is less abrasive than rendering without the required level all together.

The end result of the virtual texturing process is indistinguishable from traditional texture mapping as the technique is only a means of utilizing textures in real-time that are far greater in size than GPU resources are available. This allows artists to work on large and detailed textures in a transparent manner without being encumbered by hardware restrictions. As a testament to the uptake of this technique, both the OpenGL and DirectX graphics APIs have included hardware support for virtual texturing (Group, 2013; Microsoft, 2013).

2.7.2 Applications

Clipmaps (Tanner et al., 1998) are a precursor to virtual texturing and present an elegant solution to terrain rendering with large textures by utilizing a moving window focused around the camera that streams clipped portions of the mipmap chain that are pertinent to the visible frame. By defining concentric rings of increasing coarseness as the distance increases from the camera, outer rings use progressively lower mipmap levels to further reduce memory and bandwidth constraints. As player movement is typically smooth and gradual, the updating of this clipmap chain is efficient and minimal. The technique is simple to implement compared to the more versatile virtual texturing as the assumption that the texture data is singular and planar makes the clipping, caching and streaming far less involved. However, this assumption about the texture data was devised specifically for terrain rendering and so is not appropriate for general use cases.

The groundwork for virtual texturing was laid down by Hall (1999) where he illustrates an implementation for the hardware management of texture memory with the aims of reducing memory fragmentation and pipeline stalls whilst increasing performance and stability. Whilst this system was a feature of the hardware and operated in a transparent manner to the user, the echoes of this principle can be found in the subsequent works of the more general techniques used today. One of the first public demonstrations of a general purpose virtual texturing implementation was Barret (2008), where he demonstrated a means for addressing and managing textures in a virtual system. Whilst no formal publication exists of his work, the slides, video and source code presented on his website provided a simple and elegant demonstration of how such systems could be implemented.

One of the first commercial games to utilize virtual texturing was id's "Rage", employing a technique they called "megatexturing" (van Waveren, 2009). Utilizing huge textures measuring 128k by 128k texels, the game embraced virtual texturing by utilizing it for throughout the pipeline to produce highly detailed environments (Figure 2.8). The technique has found its way into many of the big name game engines such as Crytek's CryEngine and Epic Games' Unreal engine (Epic Games, 2009b; Mittring and GmbH, 2008).

Efficient implementations of virtual texturing are complex to implement compared to other techniques but are capable of rendering stunning landscapes in the hands of skilled artists. As the technique is essentially an evolution of mapping a single texture to the mesh, any number of materials may be used and transitions of great detail can be depicted. However, this manual generation of the virtual texture makes it unsuitable for procedural applications and whilst manually generating texture transitions is perhaps the most flexible approach for the artist; it can become impractical when large volumes of surface textures

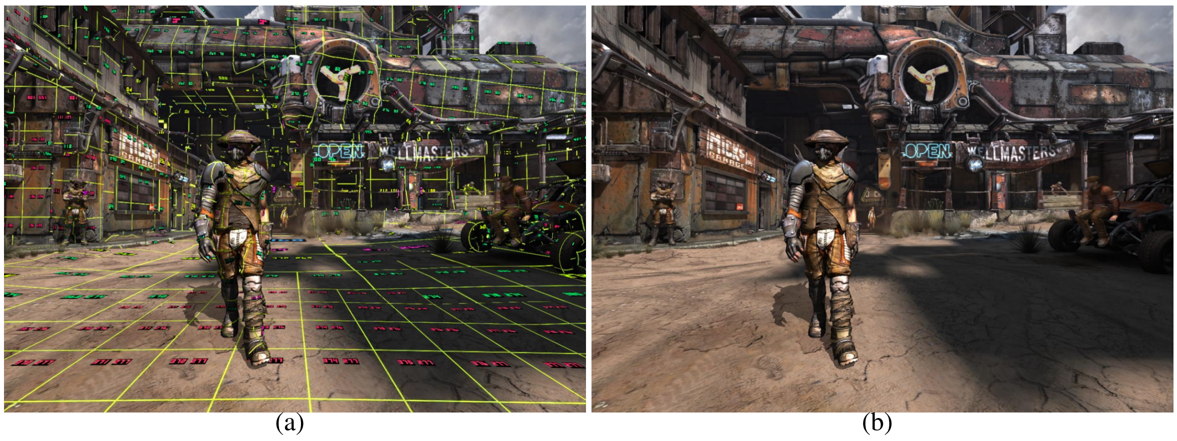


Figure. 2.8 id Software's Megatexturing (van Waveren, 2009).

need to be composited.

Virtual texturing overcomes the scaling problems of using a single texture to represent the mesh surface but there are practical limits on today's hardware. For large terrains, these generated virtual textures consume large volumes of disk space, with compressed texture sets measuring in the tens of gigabytes (Kooima et al., 2009). Whilst desktop computers typically have hard disk storage space measuring in the hundreds of gigabytes to terabytes for virtual texture storage, mobile and hand-held devices do not have such large volumes of disk space available at the time of writing.

Dice (Widmark, 2012) developed a hybrid approach of virtual texturing and their procedural shader splatting for terrain rendering in their title Battlefield 3 (Figure 2.9). They note that whilst artists are capable of creating detailed terrains using shader splatting, the technique itself can be slow to render and is not scalable in view distance so they instead splat into a texture to leverage frame-to-frame coherency for performance and to allow multi-pass rendering for scalability (Widmark, 2012). Whilst they state that they are capable of utilizing extremely large textures in the terapixel range, in practice the virtual textures typically measured 64k by 64k texels with a resolution of 32 samples per meter.

The work of Kooima et al. (2009) expands the concept of terrain transitions to a planetary scale with their multi-scale synthesis technique (Figure 2.10). Rather than using the satellite imagery directly, they use the data as input to an example-based texture synthesis algorithm to procedurally generate larger sets of perceptively similar non-periodic textures such as that used by Han et al. (2008). Using a single exemplar as input restricts the range of scales that can be practically achieved as features larger than the exemplar image or smaller than the resolution of said exemplar are missed altogether at widely differing scales. Instead,

Figure. 2.9 Terrain in Battlefield 3 (Widmark, 2012).

they use a small selection of low frequency example satellite images (in their figures a set of 16 user-selected exemplars at different scales measuring 256x256 texels is used) to composite a 16k by 16k virtual texture exhibiting features at all scales, although managing such large data sets presents its own problems in terms of storing, caching and streaming the data (Okamoto et al., 2008). The procedural nature of this technique makes it suitable for applications where terrains need to be generated at the planetary level, a feat that would be very time consuming to perform by hand. Whilst this technique could be considered niche, it is applicable to stellar simulations and games where the user can explore on a scale that dwarfs typical terrain rendering.

Andersson and Goransson (2012) explore the potential for virtual texturing across computer networks, specifically the internet. Virtual texturing has the potential to give performance gains for texture-heavy applications so the concept of streaming only the pertinent texture data across a network can greatly reduce the bandwidth between the client and server. This approach would be of particular interest to developers of browser-based games where thin browser clients require the server to stream all asset data needed to render the scenes.

HTML5 includes native support for WebGL (Khronos Group) although the authors noted that at the time of writing Google Chrome is the only browser that supports this new technology. The uptake of WebGL is on the rise as the technology matures although the main bottleneck they experienced was the execution time of the client side JavaScript and the WebGL command overhead. They conclude that the use of virtual texturing can reduce

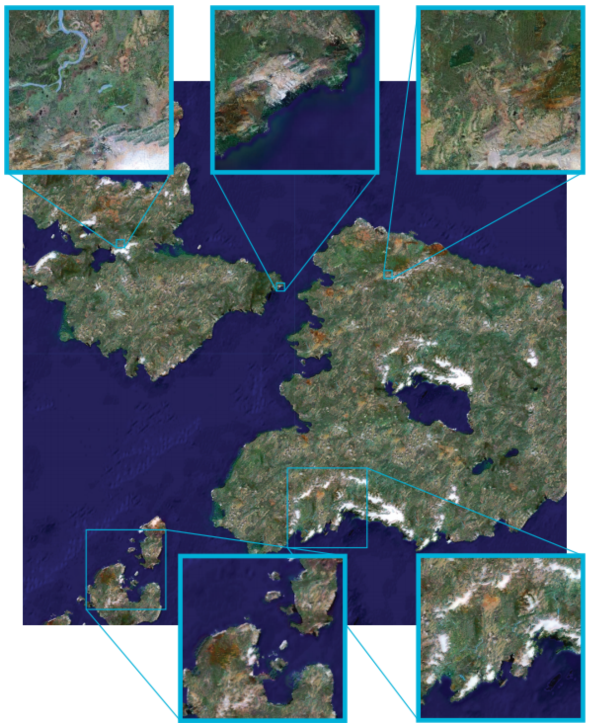


Figure. 2.10 Multiscale Texture Synthesis (Kooima et al., 2009).

bandwidth and browser support could decrease the required number of features required for the technique to be implemented. The benefits of virtual texturing over the web are the reduced load times, circumventing of server-side file size restrictions, less intermediate storage space and fewer state changes.

2.7.3 Evaluation

Virtual texturing is a relatively new technique so it does not have the ubiquitous foothold that more mature techniques such as texture splatting have although the support for this technique from the major graphics APIs and engine developers demonstrates that it will usurp traditional approaches for terrain texturing where highly detailed landscapes are required.

Performance and Scalability: When all other approaches have reached their saturation point in order to deliver their goals of diverse and detailed surface textures in real-time, virtual texturing comes into its own. Whilst the fixed costs are much higher than other approaches, once the minimum hardware requirements have been met, virtual texturing can deliver smooth and consistent performance regardless of terrain size. For applications where large volumes of texture data are required, virtual texturing can deliver performance gains due to its on-demand approach to texture data fetching, streaming only the pertinent data as the scene requires it.

For web applications in particular, virtual texturing has the potential to reduce the bandwidth requirements from client to server, allowing web-based outdoor environments to experience a level of detail that surpasses that of traditional approaches. As only the required data from pertinent mip-map levels is streamed, the network bandwidth costs are greatly reduced and network lag can be masked by falling back to a lower resident mipmap level.

Virtual texturing can scale up to any terrain size without compromising performance, detail or variation. This is in direct contrast to texture splatting and tile-based rendering where the diversity of results is limited to the example textures or tiles. Large terrains need only be provided with the appropriately sized texture and, other than the increase in storage space, this should not impact the run-time performance of the approach. The downside is that the fixed costs make it inappropriate for smaller terrains as the performance costs may be disproportionate to the actual size of the terrain surface texture. In such instances it may be more appropriate to use texture splatting or tile-based texturing and provide enough example tiles/textures to approximate the detail and diversity required for the surface texture.

Fixed Costs: The technique enjoys the novel benefit of being entirely transparent to the artist compared to other techniques where the artist has to operate within the constraints of the hardware and algorithm. However, virtual texturing's strength of complete artistic control and uniqueness can be its biggest weakness as, unlike the other approaches discussed here (where the transition synthesis is largely handled by the algorithm), each and every virtual texture must be painted by hand. Thus, for large outdoor environments measuring kilometres across, this poses a lot of work for the artist. In practice, shortcuts may be taken by re-using portions of the terrain surface or compositing and tweaking from prefabricated surface templates, somewhat counter to the benefits of virtual texturing.

Whilst performance can be gained by the on-demand texture streaming, there is a fixed overhead from the indirection texture and other data structures that make it unsuitable for small-scale terrains or those that do not have significant variation in surface detail. These fixed costs also restrict the algorithm to modern hardware as the implementation relies heavily on shaders to correctly sample the appropriate areas of the virtual texture.

As the surface texture is generated off-line, the storage costs of virtual texturing are considerably higher than other approaches with textures measuring in the gigabytes being common. Certain devices, especially mobile and embedded platforms, are greatly restricted in terms of storage space compared to their desktop counterparts so storing large, detailed surface textures in the terapixel range is not currently feasible. Memory bandwidth is also an issue as new texture data must be streamed on a frequent basis, placing further restrictions on the hardware classes capable of implementing the technique.

Implementations must utilize physical media and multi-threading effectively for a smooth experience. For consoles, this is of particular importance as often the data will be streamed from comparatively slow DVD media. Although hybrid approaches (Widmark, 2012) leverage the advantages of both texture splatting and virtual texturing, it remains to be seen how valid this approach is for procedural applications or where large volumes of terrains must be textured without artistic input.

Transition Variation and Detail: The detail of terrain transitions is only limited by the skill of the artist, giving them full control over the final result without being encumbered by hardware texture size limitations. The irregularity diversity of terrains can be approximated to an unprecedented level, providing rich, detailed and unique transitions throughout the terrain. Unlike texture splatting and texture tiling, virtual texturing does not rely on example textures to synthesize the surface texture so repetition artefacts of the texture tiling process can be avoided completely.

Both the high and low frequency details of a terrain transition can be modelled to an arbitrary resolution in a manner beyond the reach of texture splatting and tile-based texture mapping. By not relying on low resolution alpha masks or contiguous tile layouts, contours of any size and shape may be painted with each transition on the surface being bespoke in terms of shape and contour. This can greatly help suspend the user's disbelief as virtual texturing can faithfully reproduce the irregularity of real-world landscapes in a manner that cannot be achieved with the other approaches discussed.

Virtual texturing suffers from none of the translucency artefacts found in texture splatting as the artist is in direct control over how the transitions are formed. By not relying on an algorithmic process, the artist is free to design transitions that are both accurate and detailed. Whilst the individual tiles of tile-based texture mapping offer a level of detail comparable to virtual texturing, by not being restricted to a discrete set of tiles to represent the surface artists are able to create textures that are free from any repetition artefacts to create truly unique and diverse transitions.

2.8 Summary

The vastness of size, detail and variation in real-world landscapes makes it a particularly challenging field of research for real-time rendering. For decades, all but the most simplistic approximations of the natural world were out of reach for consumer-grade hardware. However, the rapid adoption and advancement of dedicated consumer graphics processing units has accelerated the field, allowing for a variety of techniques to be used to add depth and definition to computer generated terrains. The use of texture maps is still prevalent in terrain rendering so of particular importance is the synthesis of transitions where any given materials meet. A successful algorithm will reproduce transitions that are rich and varied in their shape and detail but for ultimate artistic control the human touch is still required. However, due to the sheer scope and expanse of outdoor environments this presents its own set of challenges as generating material transitions by hand is both laborious and skilled work. The techniques pertaining to the synthesis of material transitions have been described in this chapter and a summary of these techniques can be found below in Table 2.1 below.

Alpha blending techniques work well on a wide range of hardware and can produce transitions either procedurally (such as deriving weights from the mesh geometry) or with minimal effort using common, off-the-shelf image editing software. However, the transitions produced are low frequency in detail although some techniques augment the basic premise with additional stages in the blending process such as specialist high frequency blend maps (Hardy and Mc Roberts, 2006) and noise (Andersson, 2007). Still, the broader shape of the contour is tied to the resolution of the blend mask and the use of discrete material textures can make large material sets prohibitively expensive to store and blend.

Tile-based texture mapping has its roots in the 2D era of video gaming but did not quite gain the adoption for 3D terrain rendering compared to other techniques. Like alpha blending, it can run on a wide range of hardware but because the transition tiles are generated prior to assembly the results can be much more detailed than alpha blending. The drawback of tile-based texture mapping is that the variety and shape of transitions is limited by the number and size of the tiles in the set. As each tile contains a static combination of materials of a given connectivity, expanding the variety of transitions can result in a sharp increase of video memory.

Virtual texturing is a relatively new technique that removes the restrictions on texture size that were the driving force for alternative terrain texturing techniques. Instead of attempting to fit the entire surface texture in video memory, pertinent portions are instead streamed into memory on-demand. Unlike alpha blending and tile-based texture mapping, the technique requires relatively modern GPU hardware to execute in real-time but allows the greatest control over the detail and variety of material transitions. However, this absolute artistic control comes at a cost of time, man power and skill.

The next chapter will address one of the major shortcomings of alpha blending, namely the translucency artefacts exhibited in materials containing salient features and the effect alpha blending has on the blend contour. Although the work of Hardy and Mc Roberts (2006) makes improvements in this regard, the inherent lack of awareness of topography means that whilst tempered, these artefacts still remain. As alpha blending techniques work by blindingly blending fragments according to a blending equation, modulating the overall shape of the blend contour is beyond the scope of such approaches. As such, the technique described in the following chapter will address the translucency and contour shortcomings of alpha blending for features containing salient details.

Table 2.1 Summary of Evaluated Techniques

Technique	Performance and Scalability	Fixed Costs	Transition Variation and Detail
Texture Splatting (Bloom, 2000)	Performs well on fixed-function and programmable hardware, memory increases with terrain size and texture set	Terrain textures, blend masks	Low frequency detail, minimal variation
Blend Maps (Hardy and Mc Roberts, 2006)	Performs well on any programmable hardware, memory increases with terrain size and texture set	Terrain texures, shaders, blend maps, blend masks	High frequency detail, minimal variation
Procedural Shader Splatting (Andersson, 2007)	Performs well on modern programmable hardware, memory increases with texture set	Terrain textures, shaders	High frequency detail, reasonable variation
Texture Synthesis Based on Terrain Feature Recognition (Zhang et al., 2008)	Performs well on fixed-function and programmable hardware, memory increases with terrain size and texture set	Terrain textures, blend masks	Medium frequency detail, minimal variation
Visual Quality of the Ground in 3D Models (Roupé and Johansson, 2009)	Performs well on fixed-function and programmable hardware, memory increases with terrain size and texture set	Satellite imagery, shaders, blend masks, detail textures	High frequency detail, reasonable variation
Synthesizing Transition Textures on Succession Patterns (Lai et al., 2005)	Performs well on fixed-function and programmable pipeline, memory increases with tile set	Offline transition synthesis, tile sets, placement map	High frequency detail, reasonable variation
Transition Texture Synthesis (Lai and Tai, 2008)	Performs well on fixed-function and programmable pipeline, memory increases with tile set	Offline transition synthesis, tile sets, placement map	High frequency detail, reasonable variation
Pattern-Based Procedural Textures (Lefebvre and Neyret, 2003)	Performs well on fixed-function and programmable pipeline, memory increases with tile set	Offline transition synthesis, tile sets, placement map	High frequency detail, minimal variation
Megatexturing (van Waveren, 2009)	Performs well on modern shader hardware, memory costs remain constant	Shaders, modern hardware, virtual texture, manual transition synthesis	High frequency detail, high variation
Terrain in Battlefield 3 (Widmark, 2012)	Performs well on modern shader hardware, memory costs remain constant	Shaders, modern hardware, virtual texture, manual transition synthesis	High frequency detail, high variation
Multiscale Texture Synthesis (Han et al., 2008)	Performs well on modern shader hardware, memory costs remain constant	Shaders, modern hardware, virtual texture, manual transition synthesis	Multiscale detail, high variation
Virtual Texturing with WebGL (Andersson and Goransson, 2012)	Performs well on modern shader hardware, memory costs remain constant	Shaders, modern hardware, virtual texture, manual transition synthesis	High frequency detail, high variation

Chapter 3

Feature-Based Probabilistic Blending

3.1 Chapter Overview

The use of linear interpolation to blend different material types with distinct features produces translucency artefacts that can detract from the realism of the scene. The approach presented in this chapter addresses the feature-agnosticism of linear blending and makes the distinction between features (bricks, cobble stone, etc.) and non-features (cement, mortar, etc.). Using the blend weights from Bloom's texture splatting, intermittent texture transitions are generated on the fly without the need for artistic intervention. Furthermore, feature shapes are modified dynamically to give the illusion of wear and tear, thus further reducing repetition and adding authenticity to the scene. The memory footprint is constant regardless of texture complexity and uses nearly eight times less texture memory compared to tile-based texture mapping. The scalability and diversity of this approach can be tailored to a wide range of hardware and can utilize textures of any size and shape compared to the grid layout and memory limitations of tile-based texture mapping.

In addition to being used for tiled texture splatting, feature-based probabilistic blending (FBPB) can be used by artists to help generate transitions for virtual textures that would otherwise be laborious to paint by hand. After generating such transitions, the artist can import the computed blend masks into image editing software as layers and use them as templates for transition synthesis to be further fine-tuned by hand. The work presented in this chapter has been published as a poster in SIGGRAPH Asia and a full paper in Computer Animation & Virtual Worlds (Ferraris et al., 2010b, 2012).

3.2 Background Information

The textures representing the materials of a terrain depict the surface detail and topography of the material in question. For some materials (such as those in Figure 2.1(b)) the detail is vague and ambiguous or small enough to not contribute significantly to the overall topography of the surface. For example, a sand texture would depict a swathe of small grains, each measuring a few texels at most in such a manner that no particular grain stands out to the viewer. Likewise, a texture representing a dirt surface would also depict a mix of dust and other particles such that the sum of these particles is an indistinct cluster of matter that has no clear detail or topography. Transitioning between such materials can be as simple as a linear interpolation using an alpha mask as there are no salient details to be subjected to the translucency artefacts of linear blending.

Other materials (such as the cobble in Figure 2.2) contain salient details that should protrude through the surface of the underlying mesh and have their own distinct topography, giving them a three dimensional quality that alters the topography of the mesh in a significant way. Examples of such materials are cobble stones, brickwork or perhaps a rocky terrain surface. Such details are at odds with the two dimensional nature of texture mapping but can be given an approximated three dimensional feel through lighting, parallax effects or a combination of the two. Although the effect of these techniques is diminished or eliminated entirely at oblique angles, when used carefully and strategically by an artist they can help with the illusion of three dimensional topography, thus greatly increasing the perceived complexity of the underlying geometry.

However, such illusions are easily shattered when transitioning between materials containing salient details. The linear blending process of texture splatting works blindly on a fragment by fragment basis without taking into consideration the topography of the materials. This can cause such salient details to shift from opacity into translucency should they lie within the transition band, ruining the three dimensional effect. For non-salient details this is not an issue as such details are either small enough for this effect not to be noticeable or ambiguous and non-protruding enough for this translucent partial covering of other materials to be plausible. For salient details however, their protrusion through the underlying surface should have the effect of them being exempt from the occlusion of overlaying materials and thus should protrude through, regardless of the alpha mask value or simply be occluded entirely.

The obvious solution to this problem would be to use an alpha mask texture of sufficient resolution that the artist can manually exempt such salient details from the blending process or obscure them entirely towards the end of the transition. However, the resolution required to depict such detail in a blend would be near-enough the resolution of the material texture itself in order to offer per-texel (or close to) blending accuracy. As the efficiency of texture splatting is derived from using low resolution alpha masks, increasing the resolution to a sufficient level to alleviate the translucency artefacts is nearly always impractical.

3.2.1 Focus of Research

The material's features are defined as the areas of the texture that protrude through underlying materials with full opacity rather than being linearly blended according to the alpha mask weightings. The features include bricks, cobble stones and other salient details whilst non-features are the areas of a texture that are not part of a feature, such as (but not limited to) mortar and cement. Non-features lend themselves to linear blending or blend mapping because they neither protrude from the surface nor contain distinct visual details. Figure 3.1 shows a sample material (a) along with the isolated features (b).

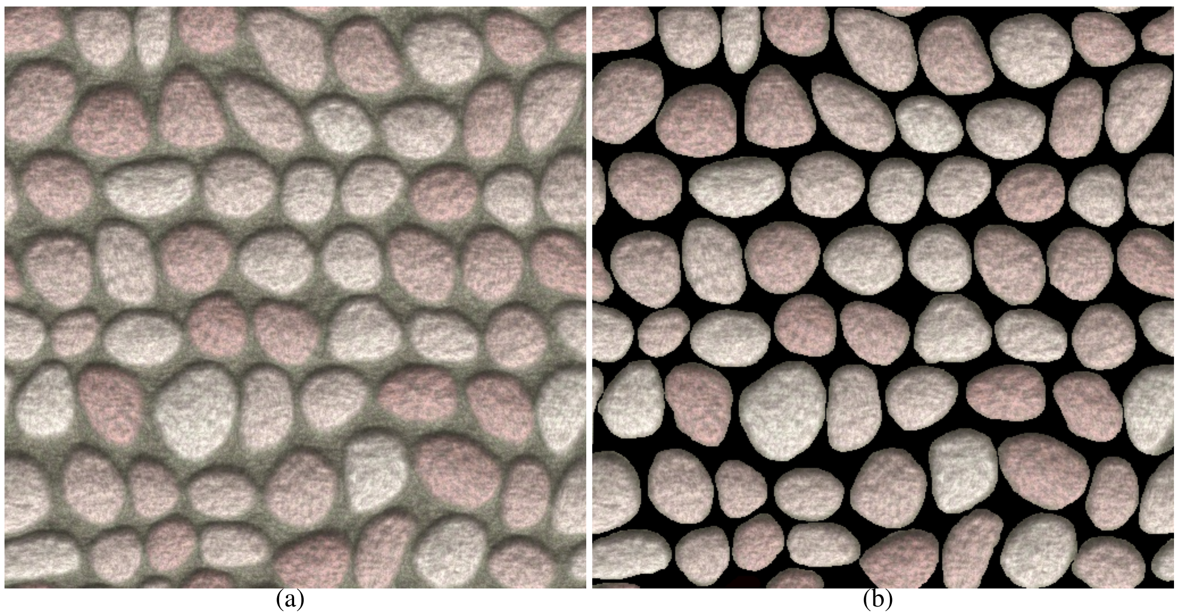


Figure. 3.1 An example material texture (a) along with isolated features (b).

When materials containing salient details transition with other materials, the typically regular layout of man-made materials should increase in disorder and intermittency until the material has fully transitioned. This intermittent transitioning of a material's features has a large impact of the material's contour as the distinct details of the features themselves draw the eye to the overall shape of the transition. A material containing such features that transitions in a uniform manner will likewise result in a uniformly regular contour. However, by introducing even subtle variation in which features protrude through or are occluded by another material will have a big impact on the overall transition contour. As the transitioning of texture mapping occurs in real-time, an algorithm to introduce such irregularity of feature placement would also be executed in real-time.

The features of man-made materials in particular are frequently subjected to wear and tear if the surface is not regularly maintained. A cobble stone path may have holes for missing stones or chipped, cracked and other damage to the features. This effect is particularly prominent at transitions as a real-world surface subjected to the elements and other wear would have an irregular shape to its features along the contour as they bear the brunt of such abuse. Even if the material is not man-made but natural (such as a rocky surface), the regularity of tiled materials detracts from the realism of the terrain as their uniform layout and detail is at odds with the irregularity of the real-world. In order to approximate this natural disorder, the features of materials should be depicted with the wear and tear they would be subjected to in the natural environment. Depicting such effects at the texture level would be insufficient as the irregularity would be negated when the texture is tiled so, instead, a real-time solution must be found.

3.2.2 Novelty of Research

Feature-Based Probabilistic Blending (FBPB) is a novel approach for adding detail and definition to the transition contour. By ensuring that salient features that protrude through the surface are drawn with full opacity, the translucency artefacts of texture splatting are removed entirely. Rather than use a static blend map to alter the transition contour, such features are dynamically drawn or discarded on a probabilistic basis to add stochastic shape to the contour. To add further detail and variation, the features themselves are dynamically modified with wear and tear to give the illusion of detail and variation to the material itself. The key to this approach is to ensure all texels of a given feature receive the same blend weight and thus are drawn (full opacity) or discarded (full translucency) together. Figure 3.2 illustrates an overview of the FBPB process.

To deduce whether a feature is drawn or not, the blend weights used in Bloom texture splatting are used as the probability of a given feature appearing. For each feature, a random number is generated. If that number lies under or equal to the probability of the feature appearing, the feature will be drawn with full opacity. Otherwise, the feature will be discarded, exposing underlying texture detail. If the texture sample is that of a non-feature, a standard

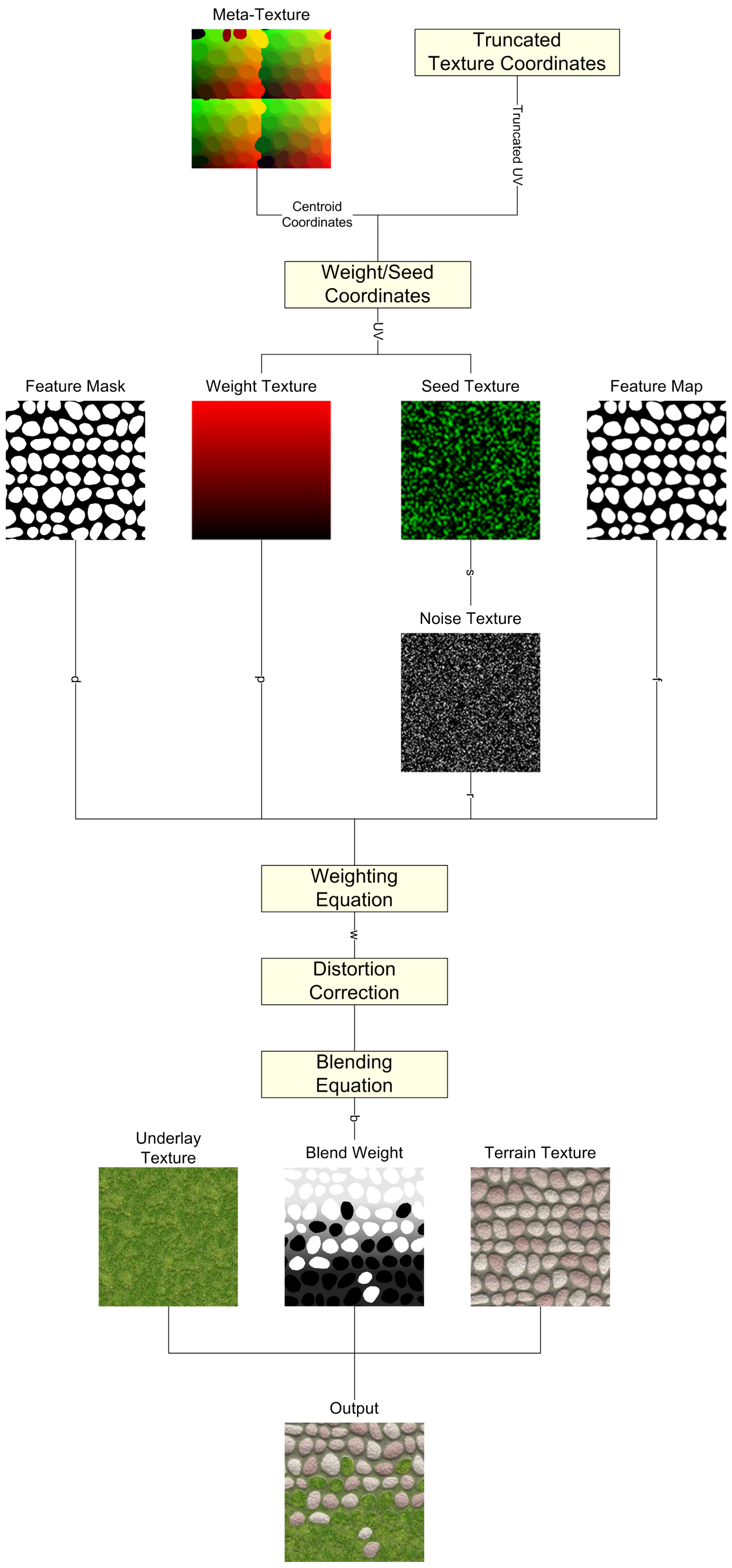


Figure. 3.2 Overview of FBPB.

linear blend using the Bloom blend weights is performed.

3.3 Related Work

Texture splatting is particularly susceptible to translucency artefacts as the blending process is agnostic to the detail and topography beyond the particular material texel(s) being sampled. In Figure 2.2(a), a cobble material is transitioning from full opacity to full transparency into a grassy material. The non-salient details of the mortar in between the cobble features has the same translucency artefacts of the cobble features but the effect is not as strong as the mortar does not have any distinct detail or topography. It is comprised mostly of dust and uniform sections of cement so it is difficult to distinguish any part clearly from another. Although the contour of the transition is uniformly horizontal, the patchy covering of such non-salient details does not make this too noticeable.

On the other hand, the features themselves fade from opaque into fully transparent. These translucency artefacts not only makes them look ethereal and unrealistic but also they do not look anchored to the surface, appearing detached and "‘floating"’ instead of firmly rooted to the ground. The features should clearly protrude through the grass material as they have a distinct height and topography but instead they fade away. This fading effect is due to the blind mixing of the cobble and grass texel data but in the real-world, a 50% coverage of grass and cobble would not result in a 50/50 mix of these materials. For non-salient materials such as sand and dirt this effect would not appear too unrealistic but as can be seen, when there is distinct detail and topography the result is far from realistic.

Blend maps (Hardy and Mc Roberts, 2006) are a simple embellishment of texture splatting that uses a grayscale image (the eponymous ‘blend map’) to weight all texels within the material’s texture in order to give certain texels more prominence during the blending process. As the blend map is typically of the same resolution as the texture itself, it offers a resolution with accuracy to the texel level. Whilst the results do still suffer from the translucency artefacts of linear blending, they are more suppressed and less noticeable. The fraying of edges along the border of features help reduce the ethereal floating of translucent features and the overall effect does offer more detail and realism than straight linear blending.

Perhaps the biggest drawback of blend maps is the lack of variation amongst repetitions of the material when it is tiled across the terrain. As can be seen in Figure 3.3, the features of the cobble stones in each repetition of the texture all receive the same weighted blend along the contour of the transition. This effect would not be so noticeable for non-salient details but for prominent features the eye is drawn to the regularity and uniformity of the

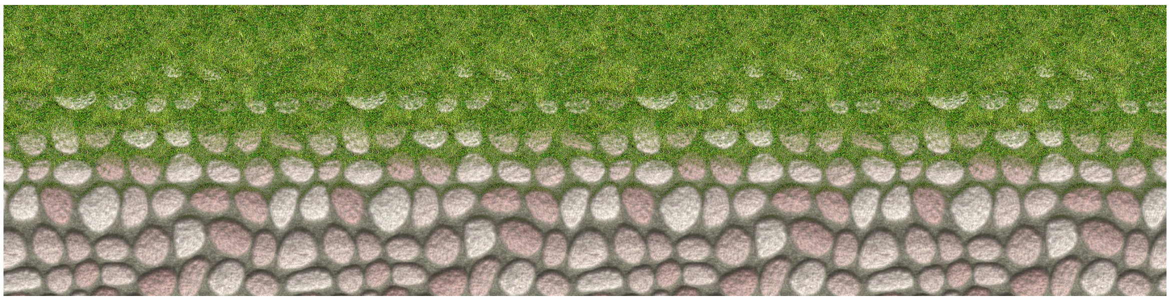


Figure. 3.3 Contours from blend maps.

transition contour. These artefacts are particularly noticeable along sections of the terrain that contain broad transitions spanning large sections of the underlying terrain surface.

Lai et al. and Lai and Tai's work on transition synthesis (Lai and Tai, 2008; Lai et al., 2005) is capable of producing results that far exceed texture splatting or any derivative technique in terms of detail for transitions containing salient features, especially materials using real-world photography. The transition cut process eliminates translucency artefacts entirely as it does not rely on a masked blending to combine multiple materials. The contours are both distinct and varied, producing transitions that have both detail and variation on a level that is beyond simple linear blending. The variety of contours helps minimize the repetition artefacts when tiling textures as the detail of the transition contour draws the eye to it, making it focus less on the tiled texture itself.

However, the works of Lai et al. and Lai and Tai are not a real-time process. Whilst the results can certainly be rendered in real-time, the transitions themselves are generated procedurally offline. As an offline process, the algorithm used to generate the transitions has the luxury of time, computational power and (perhaps most importantly) holistic information of both materials that is difficult to achieve efficiently with per-texel accuracy inside the GPU. However, such a technique could be utilized by artists when using a virtual texturing system to aid them in generating transitions that would be difficult and laborious to do in volume by hand, although with less direct control over the final results than a real-time algorithm.

Virtual texturing for transitions has the distinct advantage of both being an offline process and having full artistic control over the resulting textures. Translucency artefacts can be eliminated entirely and the contour and variation of transitions featuring salient details can be as detailed and varied as time and artistic skill permits. However, as discussed in the Literature Review section of this thesis, this absolute manual control over transition detail poses the problem of just how much time can be realistically allocated to generating transitions across numerous large terrains that would be required for a modern computer game. A 512x512 texture of small pebbles or gravel could contain hundreds or thousands of individual pebbles that must be manually isolated and transitioned in a realistic fashion. To do this by hand for a terrain spanning many kilometers in each direction would be very labour intensive and almost certainly impractical for most scenarios.

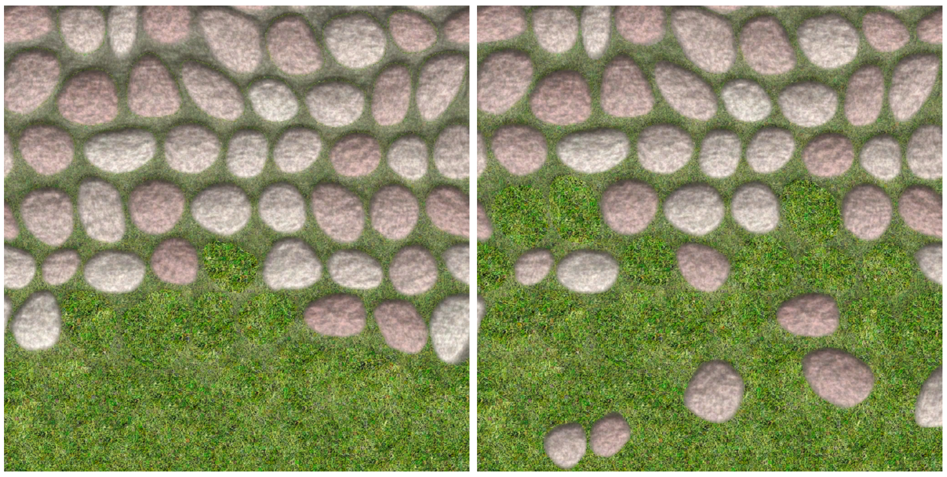
Instead, the works of Lai et al., Lai and Tai and Hardy and Mc Roberts could be used to generate transition templates that could be further tuned manually by the artist. Using Lai et al. and Lai and Tai's transition cutting as an offline process could help produce a variety of templates to be stamped across the terrain but also real-time techniques such as the Hardy and Mc Roberts one have the benefit of the artist getting instant feedback for the results. Whilst offline techniques have the advantage of producing more realistic results, there is certainly an advantage to having a real-time technique that an artist can experiment with before committing to a particular style and shape of a transition. Any technique that can help alleviate the tedium and labour when transitioning between materials containing many individual yet distinct features would be of use for applications utilizing the power of virtual texturing.

3.4 Implementation

3.4.1 Probabilistic Blending

FBPB uses the Bloom blend weight as the probability of a given feature appearing. A random number is thus generated at run-time to perform a probability check against each feature to deduce whether it will be drawn or discarded. Two approaches to random number generation are used, each yielding different results as to how a material's features are distributed at transitions. The first approach is to generate and store a random number in the $[0, 1]$ range for each weight texture texel and use this to check against the probability of a feature appearing (alternatively, one can use the fragment's texture coordinates in terrain space as a unique value pair). The result of this approach gives a neater, less random effect as features trail off abruptly and in a uniform manner (Figure 3.4(a)). This is due to the fact that surrounding features share similar random numbers due to the low resolution of the blend weight texture. If the weightings (and thus probabilities) at surrounding features are high and the random numbers are low (or visa versa), groups of features are more likely to pass or fail their probability checks together.

The second approach uses the random number as a seed to a noise function to derive a more random value at run-time. This produces a wider variance in feature draw/discard tests as similar seed values result in different random values, meaning that neighbouring



(a) (b)

Figure. 3.4 Random values (a) and seeded random values for probabilistic blending (b).

features are drawn or discarded with less predictability. This introduces more intermittency of features appearing as the material weightings trail off at terrain transitions (Figure 3.4(b)).

3.4.2 Uniform Blending of Feature Texels

The most important aspect of FBPB is to ensure that all texels within a given feature share the same probability and random value to ensure the check will pass or fail uniformly for all texels that make up that feature. If the probability and random value of the fragments sampling the material texture were used, a random pattern of feature would appear instead of a uniform block. Instead, the probability blending needs to be applied at a per-feature level rather than at a per-fragment or per-texel level. Figure 3.5 illustrates the difference between per-fragment and per-feature probabilistic blending.

Each material texture to be probabilistically blended has an accompanying meta-texture that describes the parent/child relationship of every texel in the material and the feature (if any) they belong to. Each texel that lies within a feature is considered a child texel of that feature. To ensure that all child texel samples of a given feature receive the same weight and seed, a single texel for each feature is nominated to be the parent texel whose weight and seed will be shared between all other child texels of that feature. Any child texel may be nominated as the parent texel although typically the centroid texel is used as the parent, considering it holds the average weight of the feature. The exception to this are features which are split along the texture boundary (such as the features along the perimeter of Figure 3.1(a)), more of which will be discussed in the section below.

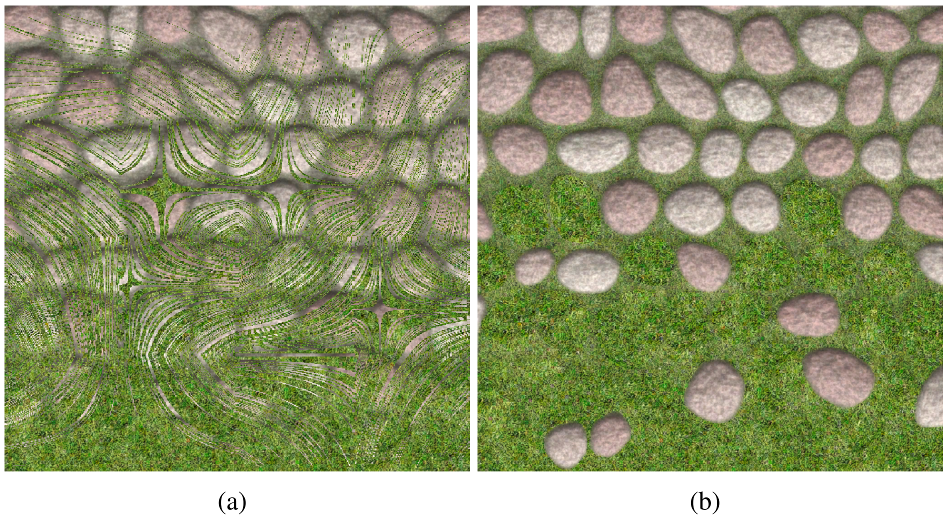


Figure. 3.5 Per-fragment (a) and per-feature probabilistic blending (b).

3.4.3 Tiling Considerations

Tiled textures are designed so that the features are laid out in a manner that allows the texture to be tiled without any seams or artefacts. Often, to improve the results of texture tiling, segments of the features are placed on the tile boundaries, with the other segments that complete the feature placed on opposing boundaries. In Figure 3.6, features that are split on the tile boundaries of the texture are highlighted in the same colour.

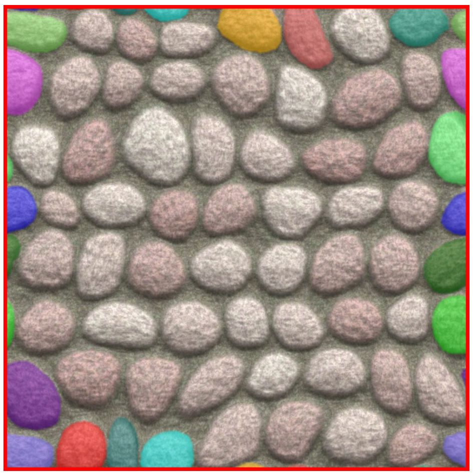


Figure. 3.6 Split features of the same colour belong to the same feature.

Note that although the split features in the above diagram only cross at most two boundaries, a feature may cross as many boundaries as necessary so long as the feature centroid is placed in a position that is common to all the split segments of that feature. In Figure

3.7(a), each split feature has its own centroid, yet this violates the principle of one centroid per feature and thus is prone to artefacts as centroids of split segments belonging to the same feature may pass or fail their probability check differently. By placing the centroid on the boundary that is common to all segments of a given split feature, all child texels of the feature are guaranteed to receive the same weight and seed value (Figure 3.7(b)), ensuring that all segments succeed or fail uniformly.

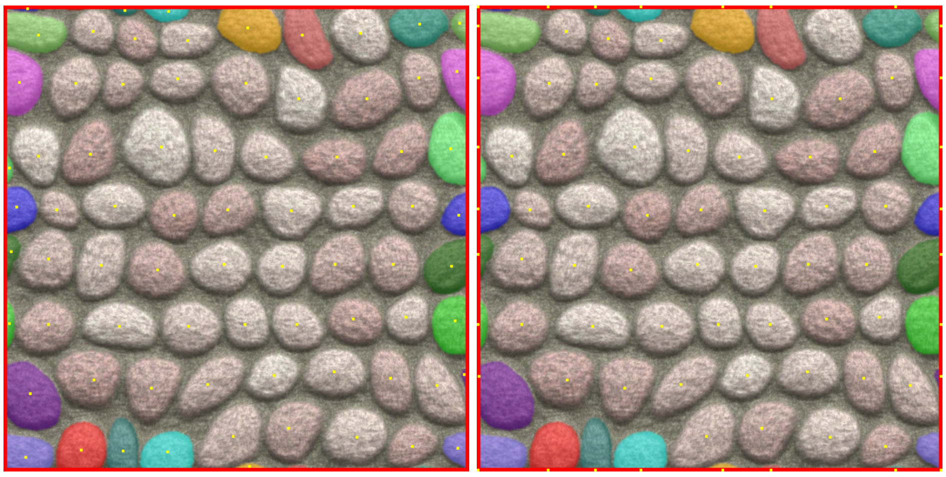


Figure. 3.7 Incorrect centroid placement for split features (a) and correct placement of feature centroids on the boundary (b).

3.4.4 Creating the Meta-texture

Figure 3.8 illustrates the meta-texture generation process. The feature list contains the colour-coded list of features where each feature receives a unique colour, thus all child texels of a given feature are of the same colour. Split features are considered separate features for this meta-texture generation process and thus receive their own colour. Non-feature texels are coloured black. The centroid list is a black and white image where the centroid texel for each feature is coloured white whilst all other texels are coloured black.

Before the feature centroids can be calculated, the features themselves must be isolated. Currently this is a manual task but through some image processing by adding contrast and brightness to the texture to bring the lighter, protruding features to the forefront an algorithm could make a best guess at what features lay where before allowing the user to fine-tune the regions by hand. Alternatively, if a procedural approach to texture generation is used, this can be done algorithmically without the need for artist intervention if the procedural

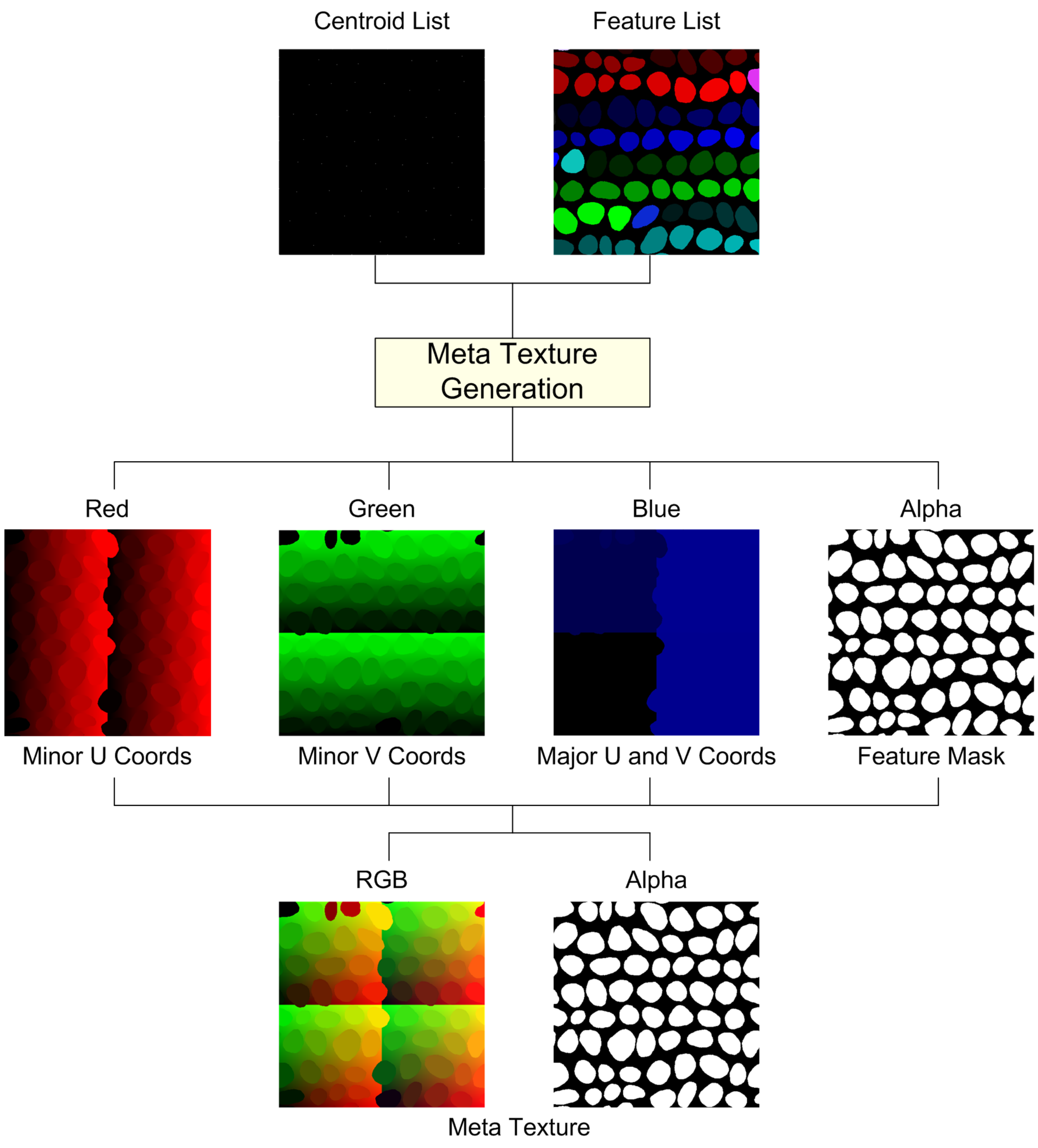


Figure. 3.8 Overview of the meta-texture generation (the major U and V coordinates image has been exaggerated for clarity).

algorithm is adapted to be made aware of the distinction between features and non-features. This need be done only once offline for each texture as the processed results are stored in an associated meta-texture for future use. Figure 3.9 shows a sample material texture along with a colour-coded region list to illustrate the isolation of said features.

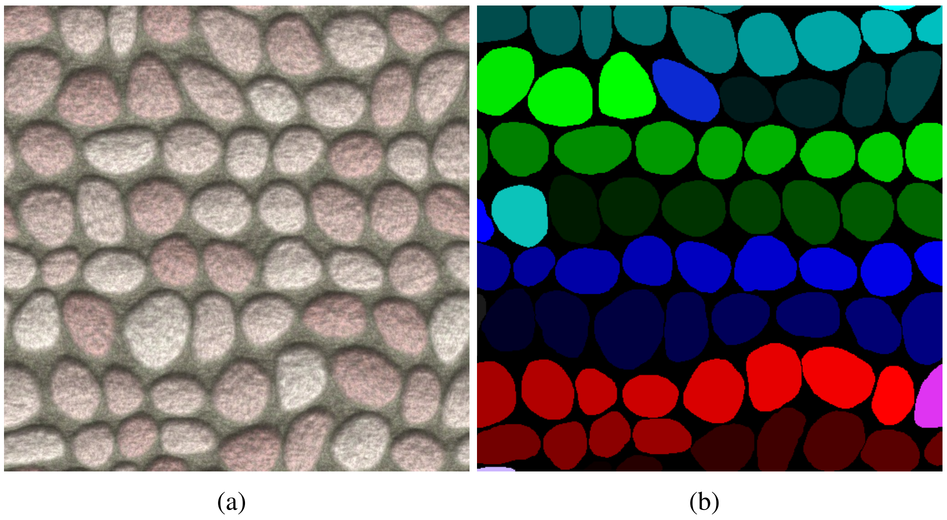


Figure. 3.9 A sample terrain texture (a) and the isolate features (b).

With the features being isolated, the centroids can then be calculated algorithmically or placed manually. Figure 3.10 shows a sample material texture along with each feature's centroid (yellow dots).

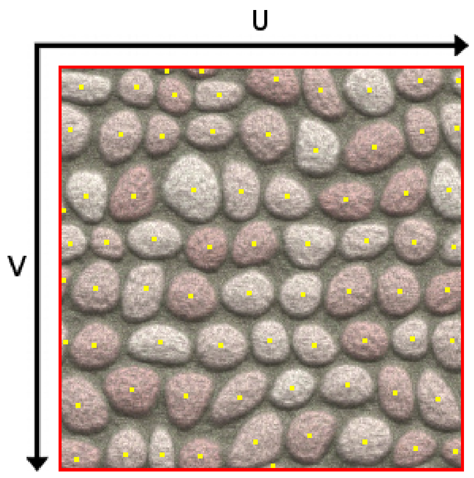


Figure. 3.10 A sample terrain texture along with the feature centroids highlighted yellow.

Once the centroids have been calculated in texture space, it will need to be known which texels belong to which feature and this information is stored in the separate meta-texture that can be accessed within the fragment shader at run-time. The meta-texture acts as a look-up table, with the input being the texel's UV coordinates. This meta-texture stores an entry for every texel position of the material texture in texture space, where upon each entry will store 2 values: the texture space position (U and V) of the centroid for the feature that the texel belongs to (if the texel is a non-feature, the output position will be the same as the input position used to perform the look-up) and the feature map which determines how a feature texel will be blended (0 indicates a non-feature, 1 to 255 indicates a feature, with the number dictating the opacity of the texel when blended). The table below shows some example entries in the meta-texture.

Table 3.1 Example entries for the meta-texture.

Input UV Coordinates	Output UV Coordinates	Feature Map Entry
(0.0, 0.0)	(0.6, 0.6)	255
(0.3, 0.6)	(0.6, 0.6)	128
(0.75, 1.0)	(0.75, 1.0)	0
(0.0, 0.25)	(0.2, 0.1)	255
(0.9, 0.6)	(0.9, 0.6)	0
(0.85, 0.75)	(0.85, 0.75)	0

The first two entries in the table are separate texels that belong to the same feature. As such, although the input coordinates differ, the output UV coordinates are the same (i.e. that of the feature's centroid). The first entry is 100% opaque (like most feature texels) whereas the second entry is 50% transparent (possibly a texel on the periphery of the feature). The 3rd entry in the table is texel that is a non-feature. Because it does not belong to a feature, the output UV coordinates are the same as the input coordinates used to make the look-up. The equations later on in this thesis use the feature map along with the output UV coordinates to perform a probability blend for features and a standard linear blending or blend maps for non-features.

The centroid position coordinates of the meta-texture are stored in base 256, where the red and green channels hold the digits of the 0th column for the U and V coordinates respectively (called the minor coordinates), whilst the blue channel stores two nibbles (packed into a byte, U being the high nibble and V being the low nibble), with each nibble denominating the digits of the 1st column (called the major coordinates). This allows the centroid positions to be stored in three colour channels rather than four, freeing up the alpha channel for the feature map.

To construct the meta-texture, the centroid list is parsed to gather the centroid positions of all features in the colour-coded feature list. The feature list is then parsed to populate the meta-texture. For black texels, the position of the texel being read is encoded in the corresponding texel of the meta-texture. For non-black feature texels, the centroid position for the associated feature is instead encoded.

Figure 3.11 illustrates the meta-texture generated for the cobble terrain texture used throughout this document. Note that in Figure 3.11(a) some feature colours look out of place. This is due to the centroids being clamped to the texture and centroid number base boundaries. For example, the red-ish features about halfway down the meta-texture are so because the border at which their centroid lies is of that colour. Feature texels are uniform in colour, whereas non-feature texels transition in a gradient across the axis of the red and green channels in colour space.

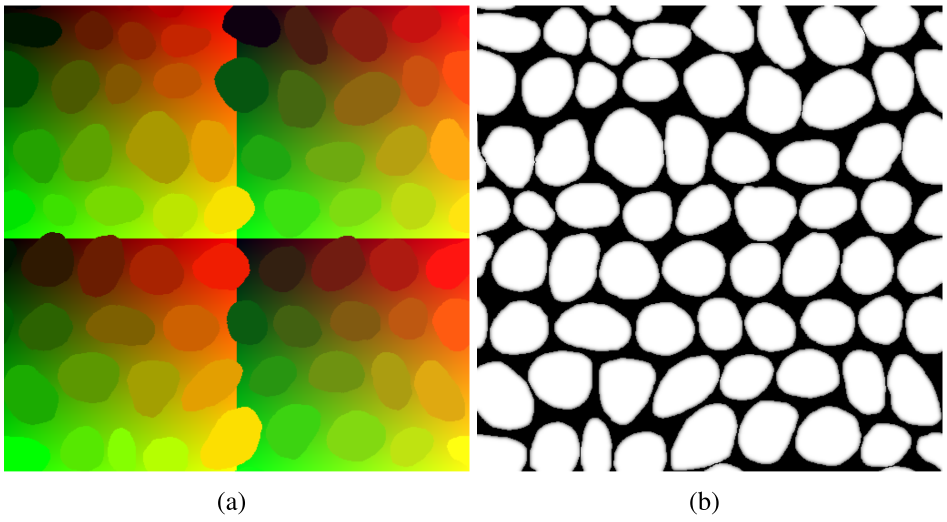
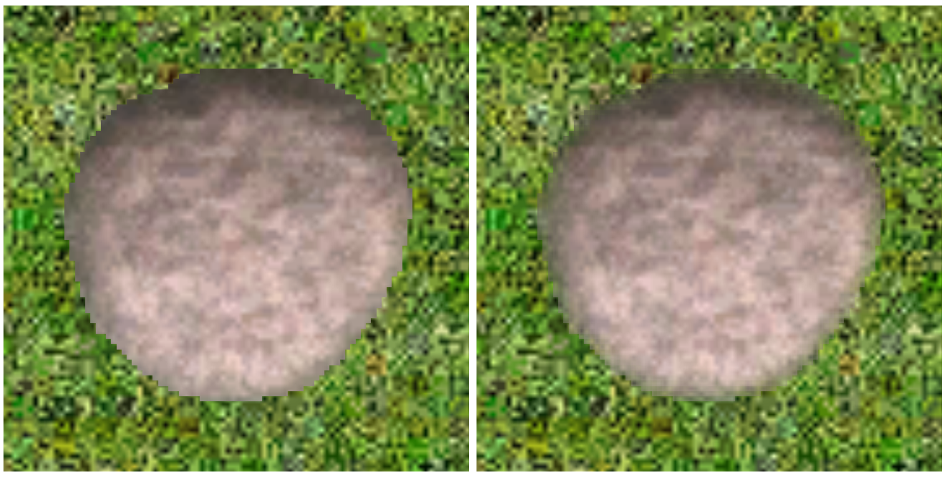


Figure. 3.11 The centroid coordinates (a) and the feature map (b) for a meta-texture.

Figure 3.11(b) shows the feature map for said material texture. Non-features are black whereas features are largely white. The exceptions to this are texels near the border of the parent feature which trail off in transparency. This softens the edges of features when they are blended on top of underlying terrain types to take away the harsh, jagged edges that would otherwise appear (Figure 3.12).

3.4.5 Centroid Position

The meta-texture is sampled using the texture coordinates $\overrightarrow{uv}$ to obtain the centroid position of the sampled texel, giving the minor coordinate vector $\overrightarrow{min}$ and the major coordinate vector $\overrightarrow{maj}$, as unpacked from the nibbles into the range $[0,255]$. The centroid vector $\vec{c}$ is obtained



(a) (b)

Figure. 3.12 A feature rendered without (a) and with (b) softened edges by using the feature map values to determine blending translucency.

by adding the vectors $\overrightarrow{min}$ and $\overrightarrow{maj}$ and expressing them as decimal fractions of the meta-texture dimensions in the range of $[0,1]$.

3.4.6 Weight/Seed Texture Lookup

The blend weights and seeds are stored in a texture of the same dimensions as the terrain mesh where the coordinates $\overrightarrow{ws}$ used to perform the weight/seed texture lookup are calculated by truncating the vector $\overrightarrow{uv}$ to obtain the integer components and adding them to the centroid position. The range of $\overrightarrow{ws}$ is reduced to $[0,1]$ proportional to the weight/seed texture dimensions. Sampling the weight/seed texture with the newly transformed $\overrightarrow{ws}$ coordinates yields the weight value p (which is also the probability) and the seed value s from the red and blue channels respectively.

3.4.7 Weighting Coefficients

A set of weighting coefficients can be introduced to the weighting equation (Equation 3.1) to further shape and control the blending of feature and non-feature texture samples when $p \leq 1.0$. These coefficients are stored in the vector $\overrightarrow{C}_{mfn}$ and refer to the feature map, feature texel and non-feature texel coefficients respectively. Coefficients greater than 1.0 sustain a given parameter whilst coefficients in the range $[0,1]$ dampen a given parameter (values of 1.0 leave the parameter untouched).

The feature map is a blurred version of the meta-texture's feature mask stored in the alpha channel of the texture itself and is used within the blending equation to taper the perimeter of features from full opacity to slight translucency, causing the edges of features to be smoothed rather than appear sharp. The sampled feature map coefficient $\overrightarrow{C}_m$ is used to smoothen the edges of features when viewed up close.

The feature coefficient $\overrightarrow{C}_f$ and non-feature coefficient $\overrightarrow{C}_n$ are used to sustain or dampen the feature and non-features. These optional coefficients are stored in the weight/seed texture at each vertex to offer finer control over how little or much features and non-features appear on the terrain mesh.

3.4.8 Weighting Equation

The weighting equation (Equation 3.1) yields the blend weight w and uses the variables f, d, r and p, all within the range $[0,1]$. f is the optional feature map value as sampled from the texture's alpha channel, d is the feature mask as sampled from the meta-texture's alpha channel and r is the random value obtained by sampling the noise texture using the seed value s as input. Areas of the terrain with a 0% blend weight should always fail the probability test, thus the noise function should produce values in the range $(0,1]$.

The purpose of the weighting equation is to return either a binary translucency value (0 or 1) for feature texels and a translucency value in the range $[0,1]$ for non-feature texels. The key to this is the binary feature mask value d which ensures that a value of 1 will nullify the right hand side of the equation and a value of 0 will nullify the left hand side.

$$w = \underbrace{\text{sgn}(\lfloor \frac{1}{r} dp\overrightarrow{C}_f \rfloor)\overrightarrow{C}_m \max(p,f)}_{\text{feature texels}} + \underbrace{p(1.0-d)\overrightarrow{C}_n}_{\text{non-feature texels}} \tag{3.1}$$

The first part of the weighting equation accommodates feature texels and returns a value of either 0.0 or 1.0. The reciprocal of r is multiplied by p to give a value of < 1.0 for texels whose probability is less than the value of r and ≥ 1.0 for probabilities greater than r. The floor function ensures that values < 1.0 are truncated to zero, allowing this side of the equation to nullify if the probability check fails. The sgn function caps results of the floor function to 1.0 should the result be greater than 1.0.

If p or d is zero or if the random value is greater than the probability, the sgn function returns 0.0, causing the 'feature texels' part of the equation to null, allowing the 'non-feature texels' to generate a non-feature weight. A feature map value f that is less than the non-feature translucency at that point will cause visible seams along the boundary of features so the max function is used to take the greater value of the variables p and f.

The second part of the equation accommodates non-feature texels. If the feature mask value d is 1.0 (a feature texel), this side of the equation will null, allowing the left-hand side to generate a feature weight. For non-feature texels, the Bloom weight p is used to perform a linear blend in the range of $[0,1]$ with the underlying texture.

3.4.9 Blending Equation

Previous work of Ferraris and Gatzidis (2009) and Ferraris et al. (2010a) blended multiple textures in order of precedence, such that lower precedence textures were masked by higher precedence textures of a higher blend weight. With FBPB, feature texels of lower precedence textures always take priority over higher precedence non-feature texels. This ensures that features will always be visible even for lower precedence textures. Features for higher precedence textures are blended in priority over features for lower precedence textures.

Equation 3.2 keeps track of whether or not any features have been drawn for precedence levels below the level that is being blended. For each level, a visibility flag V and feature flag F is calculated for a given precedence level n. (Equation 3.3). Here, the value V is set to 1.0 should a given texel be a visible feature texel, otherwise it is set to 0.0. The value F is then set to either 1.0 or 0.0 using the sgn function should the value of V for a given precedence level or the value of F for a previous precedence level be 1.0.

$$\begin{aligned} V_n &= d_n w_n \\ F_n &= \operatorname{sgn}(F_{n-1} + V_n) \end{aligned} \tag{3.2}$$

Once the feature and visibility flags have been calculated for a given level, the blending equation (Equation 3.3) is executed for each precedence level to dictate how much of said precedence level's texture is blended with the previous. The Heaviside step function will return a value of 1.0 should the current texel be a visible feature texel or if neither of the previous levels contain visible feature texels. Should the current texel be a visible feature texel and any of the previous levels contain visible feature texels, the feature texel of the current level (of being higher precedence) will be visible instead. The result of this function is then multiplied by the weight value of this current precedence level (w_n) to give the final blend value that level (b_n).

$$b_n = \mathscr{H}(V_n, F_n) w_n \tag{3.3}$$

3.4.10 Feature Variations

Feature variations are used to dynamically introduce unique wear and tear to texture features when the probability of appearing lies below 100%. Executed prior to the weighting equation, they are achieved by using the seed and an additional random number at each weighting to sample the grayscale variation map (Figure 3.13) detailing various cracks, divots and holes at a random point. This random sample is then used to modulate a given texture to darken the colour. Furthermore, by modulating the texture's normal map with the variation normal map (generated from the variation map) and nullifying d when the variation map lies under a certain threshold, holes can be created and chunks removed, exposing any underlying texture information as shown in Figure 3.16(c). Feature variations work especially well when modulating the texture's normal map with the variation normal map to add depth to the illusion of surface deformation.

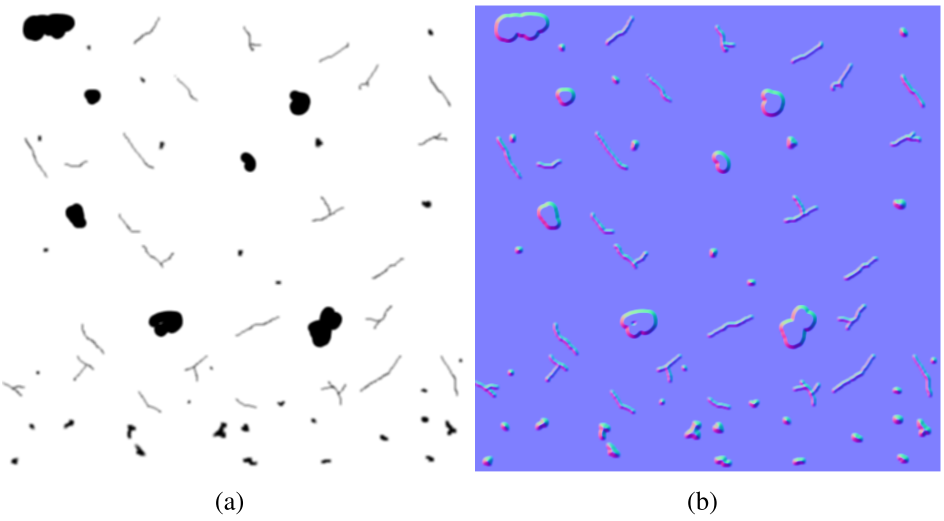

(a) (b)

Figure. 3.13 The albedo (a) and normal map (b) for a variation texture.

The variations texture is sampled using the texture coordinates $\overrightarrow{uv}$ offset by a random vector. The first component of the random vector is the original random value obtained by the seeded lookup to the noise texture. To obtain a random value for the second component, the green channel weight/seed texture is filled with another set of random seeds. Thus, when the weight/seed texture is sampled, a vector $\vec{s}$ is yielded rather than the original variable s. This vector is used to perform a lookup to a two dimensional noise texture, yielding the random vector $\vec{r}$ rather than the original variable r (the weighting equation remains unchanged, however, the first component of $\vec{r}$ is used in place of r). The vectors $\vec{uv}$ and $\vec{r}$ are added together to give the vector $\overrightarrow{var}$ which is used to sample the variation texture.

Algorithm 1 Modulating feature with the variation texture

```
if $d$ then
    if variation map $< 1.0$ then
        $t\_norm \leftarrow v\_norm + (t\_norm - v\_norm) \cdot w$
    end if
    if variation map $<$ threshold then
        $d \leftarrow 0$
        $colour \leftarrow colour \cdot coefficient$
    end if
end if
```

Algorithm 1 shows the pseudo-code for implementing feature variations. Firstly, only feature texel samples are subjected to variations; non-feature samples pass through unmodified. The texture normal map *t_norm* is modulated by the variation normal map *v_norm* by blending between the two using w as the blend amount. This allows high probability features to remain unchanged whilst features of decreasing probability receive more pronounced modulation. Optionally, should the variation map sample lie underneath a threshold, the colour of the texture is modulated by a coefficient and the feature mask d is set to zero, darkening surface deformations and allowing cuts and holes to expose any underlying texture detail. For the variations in the results in this chapter, a threshold of 0.3 and a colour coefficient $\vec{C}_c$ of 0.75 were used to darken the divots and reveal the underlying surface for particularly deep divots.

3.5 Results

In this section FBPB will be compared with linear blending (Bloom, 2000), blend maps (Hardy and Mc Roberts, 2006) and tile-based texture mapping (Cohen et al., 2003; Wei, 2004) (where applicable).

Figure 3.14 is an example of a tile terrain texture blending from right (100%) to left (0% tile). With both FBPB and tile-based blending, the tiles trail off intermittently rather than uniformly, whilst FBPB introduces deterioration in the form of cracks, chipped tiles and scratches, breaking up the uniformity of the transition. The blend map delivers a more convincing result than linear blending with the dirt appearing in the gaps between the tiles although compared to FBPB the uniform nature of the blend would result in obvious texture repetition when blended over a large area of the terrain mesh. Note that the tile-based blends are restricted to such grid-like textures whilst FBPB can be used with any feature layout and also produces random variations that increase in frequency and prominence as the blend

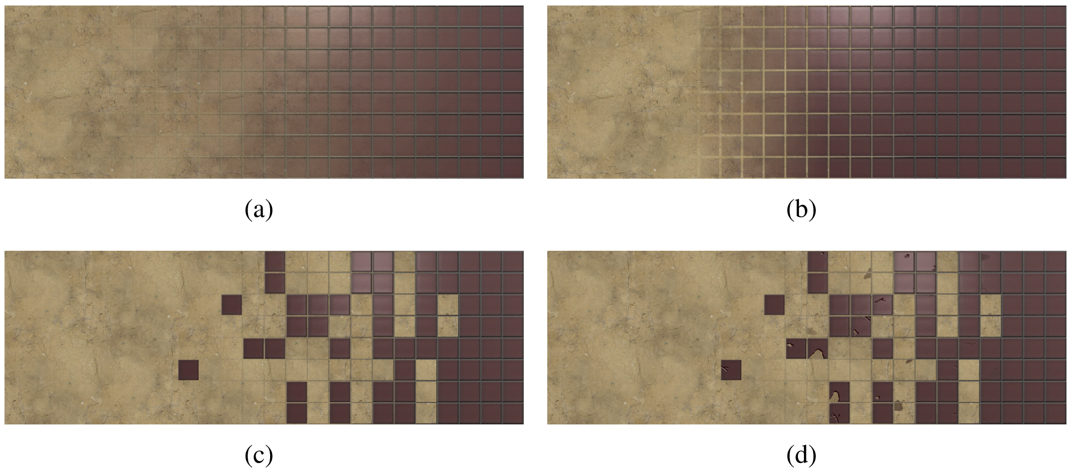


Figure. 3.14 Blending comparison of (a) linear blending, (b) blend maps, (c) tile-based texturing and (d) FBPB.

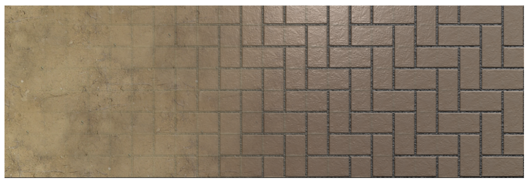
weight reduces.

The lookup table for the tile-based texturing was generated manually using the results from the FBPB blend in order to determine which features were drawn and discarded. It was found that the limitations of the grid layout and labour intensity of manually populating the lookup tables significantly impacted the work flow and flexibility of results. FBPB suffers from none of these limitations as the intermittent pattern of features is generated automatically and our approach can be used with any textures with salient detail.

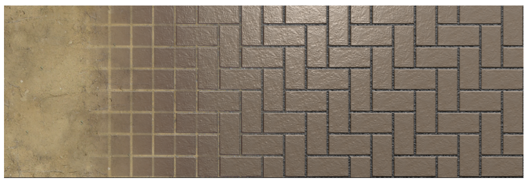
As tile-based texture mapping obtains the lookup table address implicitly from the fragment texture coordinates, the technique is restricted to features that are laid out uniformly on a grid, as shown in Figure 3.14(d). In the following examples it will not be compared with FBPB.

Figure 3.15 illustrates a particular terrain transition that cannot be reproduced properly using existing blending approaches. Both the tile and mosaic textures blend from 100% (right) to 0% (left). Whilst FBPB can achieve this complex blend in a convincing manner with no artefacts, the blend map and linear blend simply cannot represent such a transition as the two textures cannot be distinguished from one and other, giving FBPB a significant advantage.

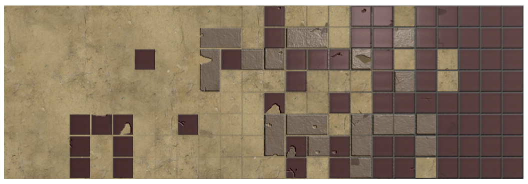
Figure 3.16 shows close up shots of a top to bottom blend with parallax mapping enabled. For this, a cobble stone texture was blended with a grass underlay. This particular texture was chosen because it represented a hypothetical 'worst case' insofar as having features of unique shapes and sizes laid out in an irregular manner. For linear blending, the mixture of texture and underlay at a mid-point of the blend transition produces an artificial



(a)



(b)



(c)

Figure. 3.15 Blending comparison of (a) linear blending, (b) blend maps and (c) FBPB.

result with heavy translucency artefacts as the cobble and grass cannot be distinguished from each other. The illusion of relief the parallax effect should be producing is undermined by these heavy blending artefacts. The blend maps still suffer from the translucency artefacts, albeit to a lesser degree as the two textures can be distinguished but, like linear blending, the parallax effect is still lost when the artefacts are at their most prominent as the blend weight approaches closer to 0%. In some instances, the opacity effects of linear blending and blend maps may be desirable, for example if the occluding material is a fine dust where partial coverage of features is to be expected. However, for situations where this partial coverage is not desirable, FBPB ensures that features properly protrude through the underlying surface.

For FBPB, none of these translucency artefacts are displayed, resulting in a far more convincing blend. Here, even at the mid-point of the transition, the stochastic nature of the probability blend breaks up the banding artefacts that are exhibited by linear and blend maps as they blend from top to bottom. Furthermore, the features where chunks have been removed from the corners and sides illustrate how convincing the feature variation process is when combined with parallax mapping, especially when considering the fact that all of the variations were generated dynamically with no artistic input.

Figure 3.17 depicts a terrain with 3 patches of cobble blended with a 50% mix of grass and cobble using linear blending, blend maps and FBPB. For linear blending, the 50% mix of grass gives the cobbles an artificially dull appearance and the translucency artefacts become more pronounced as the blend weight drops off from the centre towards the perimeter of the patch. The blend maps do not suffer from the dull appearance although the shape of the circular brush used to paint the patch is revealed at the perimeter. This could be fixed by having the artist manually touch up the periphery of the patch to introduce irregularity but in practice this will be limited by time and mesh resolution. FBPB breaks up the uniform shape of the brush automatically and can be further fine-tuned using the weighting coefficients (either globally or per-vertex).

3.6 Performance Analysis

A 'worst case' scenario was fabricated by blending two textures using FBPB and the existing alternatives along with a base texture across an entire mesh that was rendered with no optimizations. For tile-based texture mapping, the approach described by Wei (2004) was extended to draw or discard feature texels together in the same manner as FBPB. The weighting algorithm was a simplified version of Eq. 3.1 as illustrated below, where *m* is the binary result of sampling the lookup table:

$$w = \operatorname{sgn}(md)\max(p, f) + p(1.0 - d) \tag{3.4}$$

The terrain mesh consisted of 512 quads (513x513 vertices) with a texture scale of 1.0. The texture sets used were the lowest common denominator that the tested approaches could support (as discussed in Section 3.5). The two textures blended algorithmically measured 8x8 and 10x10 in features respectively whilst the base texture was mixed in with a standard linear blend. Three configurations were tested: a straight blend (no lighting or parallax effects), a normal mapped blend and a parallax blend. The viewport was filled entirely with blended fragments and the hardware used was a Radeon Mobility 4600 Series GPU in a Dual Core 1.5 GHz CPU laptop with 3 GB of RAM.

Algorithm	Straight			Normal			Parallax		
	FPS	Ms[1]	Memory Usage[2]	FPS	Ms[1]	Memory Usage[2]	FPS	Ms[1]	Memory Usage[2]
No Texturing	385	2.597	-n/a-	-n/a-	-n/a-	-n/a-	-n/a-	-n/a-	-n/a-
Base Texture Only	313	3.194	1,024	-n/a-	-n/a-	-n/a-	-n/a-	-n/a-	-n/a-
Linear Blending[3]	264	3.788	3,586	215	4.651	5,634	203	4.926	5,634
Blend Maps[3]	263	3.802	3,586	214	4.673	5,634	199	5.025	5,634
Tile-Based Texturing[3]	235	4.255	45,570	137	7.299	47,618	135	7.407	47,618
FBPB[3]	226	4.424	5,891	135	7.407	7,939	134	7.462	7,939
FBPB With Variations[3]	219	4.557	7,172	127	7.874	9,220	125	8.000	9,220

[1] Frame time (in milliseconds)
[2] Total texture memory usage (in kB)
[3] Straight blend between two terrain textures and a base texture with no other effects

Table 3.2 Performance results.

Table 1 details the results of the performance tests. The relative performance of the approaches is in sync with their relative complexity. Once normal and parallax mapping were enabled, the extra overhead of these techniques introduced reduced the relative difference in performance between the approaches. The four key advantages FBPB offers over tile-based texture mapping are (i) the considerably lower video memory overhead (the texture usage of tile-based texturing is nearly eight times more) (ii) the complete automation (compared to manually populating lookup tables with feature data) (iii) the scalability and diversity and (iv) more importantly, the fact that FBPB can utilize textures of any size and shape compared to the grid layout and memory limitations of tile-based texture mapping.

3.7 Conclusion and Future Works

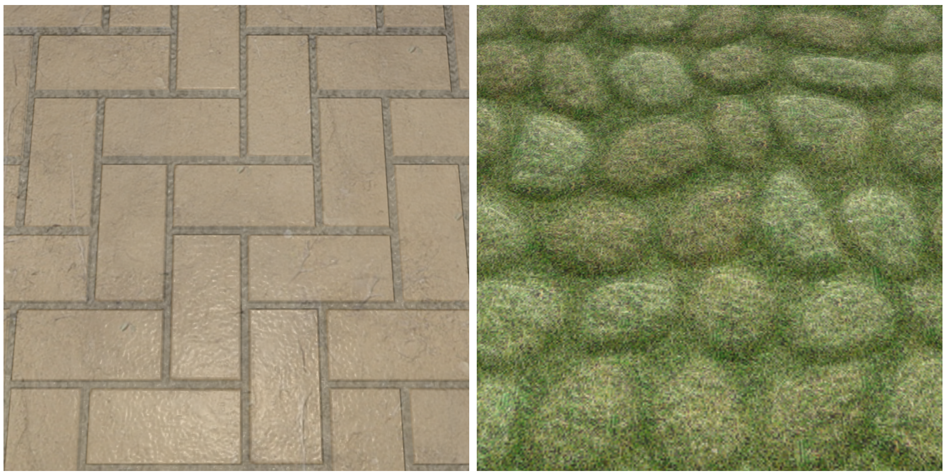
A novel approach FBPB has been proposed in this section to introduce intermittency and irregularity at transitions for terrain types that have distinct features. FBPB completely removes the translucency artefacts that exist in traditional Bloom texture mapping and can generate a near-endless number of transitional variations in real-time without any artistic intervention. Compared to tile-based texture mapping, FBPB uses considerably less memory to store texture data. Furthermore, FBPB can handle any number of features at a constant overhead in terms of memory usage and algorithmic operations.

Performance and Scalability: The performance of FBPB is as good as tile-based texture mapping but slower than blend maps although this is a reflection of the relative complexity and diversity in results. The technique is capable of scaling to terrains of any size as the meta-texture resources are required on a per-material basis rather than being an attribute of the mesh geometry. When many materials are used (how many is ultimately dependent on the hardware and texture format), FBPB (like texture splatting) is susceptible to becoming file rate bound however real-world applications can optimize the level of overdraw by masking the materials such that only those that are required for the frame are utilized (Andersson, 2007).

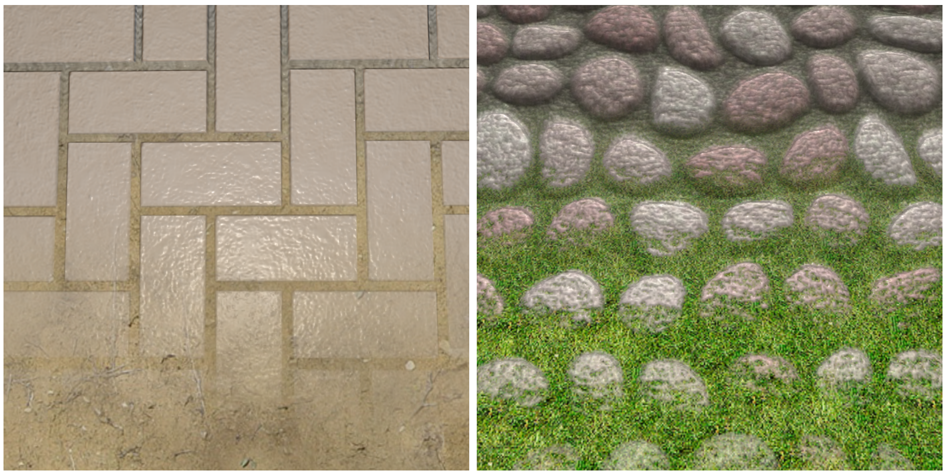
Fixed Costs: FBPB has significantly lower memory overhead compared to tile-based texture mapping as the variations of contour and feature shape are dynamic, as opposed to a static selection of pre-generated tiles. The only additional over head at run-time is the noise function and meta-texture, although meta-texture generation is an offline and currently manual process. However, the meta-textures need only be generated once and are fixed in cost for each material, making the texture memory overhead both consistent and predictable. GPUs with a programmable pipeline are required as the technique is not possible to implement with the fixed-function pipeline but this should not be an issue given that most consumer grade desktops, consoles and mobile devices meet this basic minimum requirement.

Transition Variation and Detail: The primary focus of FBPB is to alter both the contour and layout of features in real-time in order to create the illusion of the irregularity and variation found in the real world. The dynamic altering of feature shape through the wear and tear strengthens this illusion and adds details that are further embellished with normal mapping and parallax mapping. Parallax mapping in particular helps alter the topography of the features to further add detail and variation to the material, breaking up the regularity that is typical of tiled textures. The technique is however restricted to materials containing prominent features so the use case is more restricted than a more general purpose algorithm.

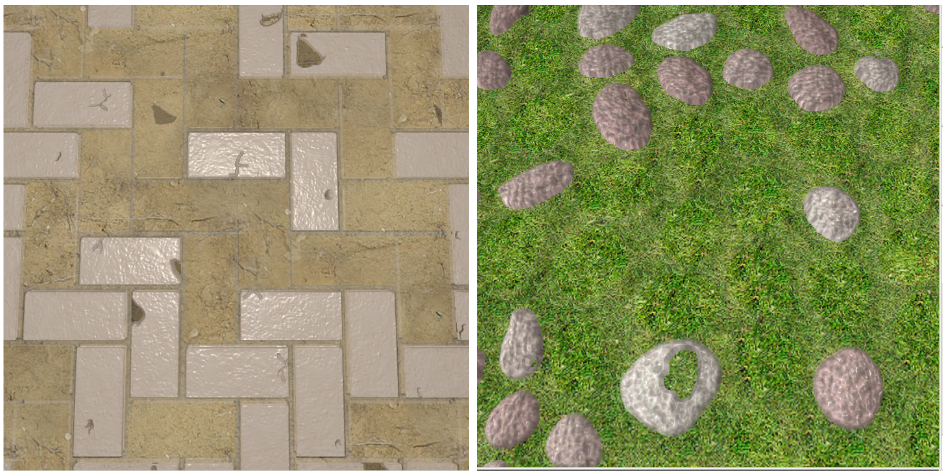
Future Works: Currently, FBPB only works with textures that contain salient details. Future work will involve expanding the technique to work with textures that do not contain distinct feature information, such as grass, mud and sand by procedurally generating "feature masks" in real-time by dividing up texture space into logical grids where, instead, each cell of the grid will be probabilistically drawn or discarded. Instead of using static feature masks, an elaboration of the feature variations aspect of our approach will be explored to generate unique shapes in real-time to deliver splatters, clumps and pockets of non-feature textures at terrain transitions. Furthermore, features of lower precedence will occlude non-features of higher precedence but currently there is no scheme to ensure that lower precedence features are not partially occluded by features of higher precedence. Further research will look into a hierarchical approach to feature occlusion to ensure that such partial occlusion does not take place.



(a)



(b)



(c)

Figure. 3.16 Close up parallax shots of linear blending (a), blend maps (b) and FBPB (c).

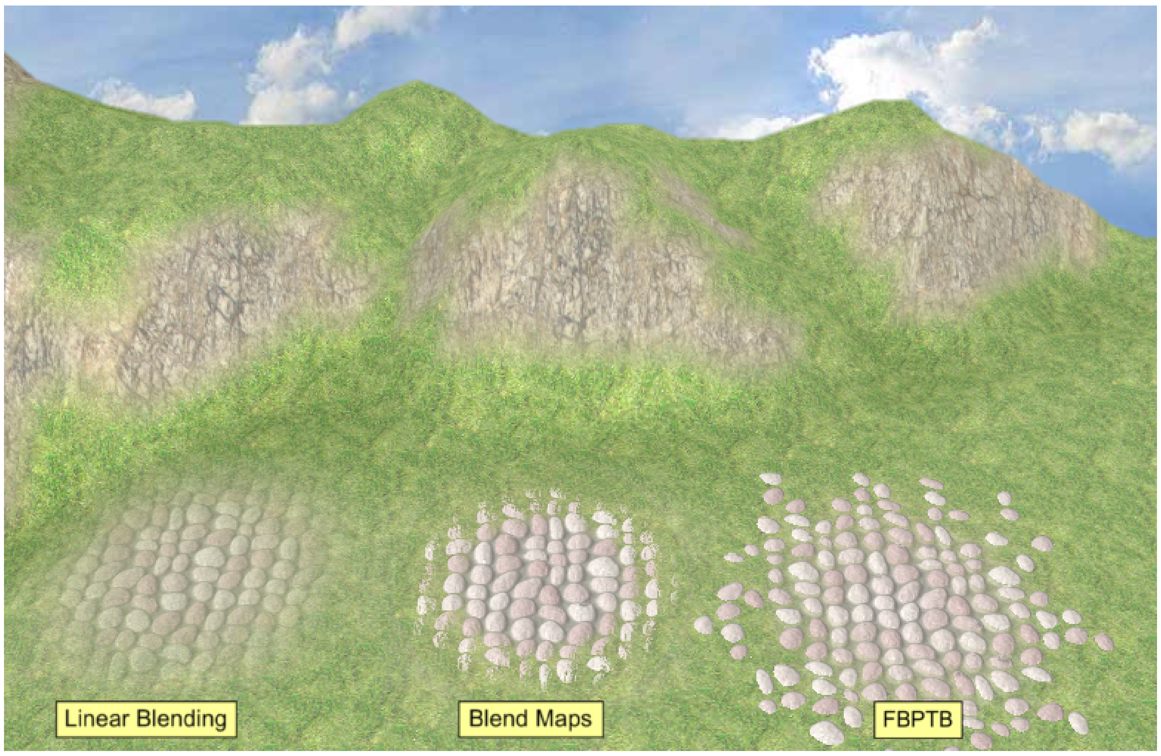


Figure. 3.17 A terrain shot with examples of a 50% mix of cobble and grass using linear blending (left), blend maps (middle) and FBPB (right).

Chapter 4

Transition Contour Synthesis With Dynamic Patch Transitions

4.1 Chapter Overview

The technique of splatting multiple textures to a terrain mesh using an alpha texture is often restricted to low frequency detail due to the texture memory overhead needed to depict highly detailed transitions. Instead, it is common to modulate this low frequency blend with a high frequency blend map (stored with each terrain material) to greatly increase the resolution of the transition. However, the overall shape of the transition remains untouched and can display unnatural uniformity when adjacent areas of the terrain mesh receive the same blend weight. This chapter presents a novel automated approach for generating stochastic contours when transitioning between material textures. This is achieved by modulating the alpha masks on-the-fly with the subdivision of texture space into different sized patches to produce irregular contours from minimal artistic input. The results have proven that the enriched detail of the transition contour can be achieved with a performance competitive to existing approaches without additional texture and geometry resources or asset pre-processing. This approach is of particular importance for applications where GPU resources or artistic input is limited or impractical. The work presented in this chapter was presented as a short paper for Computer Graphics International and accepted as a full paper in Computers in Entertainment (Ferraris et al., 2015).

4.2 Background Information

In a typical texture splatting implementation, the low resolution alpha mask is filtered linearly by the GPU when it is sampled in the fragment shader. This filtering process results in very low frequency shapes to the transition contour with smooth and gradual transitions between texel sample points on the alpha mask. This low frequency detail results in blurry, washed out transitions when interpolating between materials. When using blend maps, the contour of the transition can be “frayed” and broken up somewhat by adding high frequency detail but such detail lacks irregularity due to the blend maps being tiled alongside the material textures. This static nature of blend maps makes it inadequate for modulating the broader contour of the transition with sufficient detail and irregularity.

When the resolution of the alpha mask is low enough, the limitations of the linear filtering process can produce regular, grid-like shapes to the contour where the mask texels are too large in comparison to the material textures. For FBPB, this effect was not particularly noticeable because the dynamic drawing or discarding of the features was sufficient to break up the contour, drawing the eye away from the less noticeable non-salient details that were subjected to a linear blend. Increasing the resolution of the alpha mask will help suppress these filtering artefacts but, in order to add truly high frequency detail to the contour, the required resolution of the alpha mask would be impracticable for most applications.

In order to dynamically affect the contour of a transition between materials which contain non-salient details, storing addition modulation information along with the material textures themselves is insufficient. Instead, the process must operate at a higher domain of the blending process independent of the material textures themselves. Extrapolating the concept of blend maps to the level of the alpha mask textures to provide high frequency modulation would not be practical as the efficiency of blend maps is due to the fact that the blend maps are stored in the alpha channel of the albedo texture, effectively coming at no extra cost. Thus, a successful technique would need to balance the efficiency of texture splatting with the additional overhead needed to appropriately add detail and irregularity to the transition contour.

4.2.1 Focus of Research

An algorithm that aims to address the issues outlined in the previous section should focus on adding detail and variation to alpha mask contours in order to increase the fidelity of the material transitions. The detail added need only be sufficient enough to break up the broad shape of the filtered alpha mask as the blend maps will add the higher frequency detail to the transition. The variation should be sufficient enough to avoid obvious repetitions of the contour shape in order to give the illusion of the endless variation found in the natural world. In order to add sufficient variation, the algorithm must have the appearance of non-determinism so would need to execute at run-time rather as a static off-line process (albeit without compromising frame-to-frame coherency).

For materials containing no salient details, there are no distinct topographical features that need to be preserved so the issue of translucency artefacts is not a focus like with FBPB, allowing a greater degree of flexibility with regards to how the contours are modulated. As the high frequency detail can be accommodated with blend maps, one need only implement a means of adding mid-level frequency detail to break up the shape of the contour sufficiently to avoid the unvaried, washed out contours of regular splatting. Thus, the challenge can be broken further into three tiers: low frequency detail (derived from the alpha mask), mid frequency detail (derived from the algorithm) and high frequency detail (derived from the blend maps).

As the materials used by such an algorithm will contain little or no salient details, there should be no need to store meta-information along with the albedo texture (as is the case with FBPB). Thus, the algorithm should be able to work with a variety of materials without the need to process the materials themselves to gather meta-information about the layout and topography of the material surface. This will require the algorithm to be flexible enough in its parameters to be able to be tuned to a given material to produce authentic and varied transitions.

4.2.2 Novelty of Research

In this chapter a novel approach is proposed for transition contour synthesis with dynamic patch transitions (DPT). DPT builds on the previous work of Feature-Based Probabilistic Blending (FBPB) (Ferraris et al., 2012), expanding the concept of probabilistic blending to present a general purpose algorithm for material transitions for textures that contain non-salient details. The work presented here is an extension of a paper published in Computers in Entertainment (Ferraris et al., 2015), with this section detailing the approach, addressing minification aliasing and contrasting the performance and results of the algorithm to blend maps (Hardy and Mc Roberts, 2006).

A typical approach to increase the quality of transitions is to modulate the low frequency blend weight with a material's high frequency blend map (Hardy and Mc Roberts, 2006). Whereas this technique adds texel-level accuracy to transitions (Figure 4.1 (a)), DPT fur-

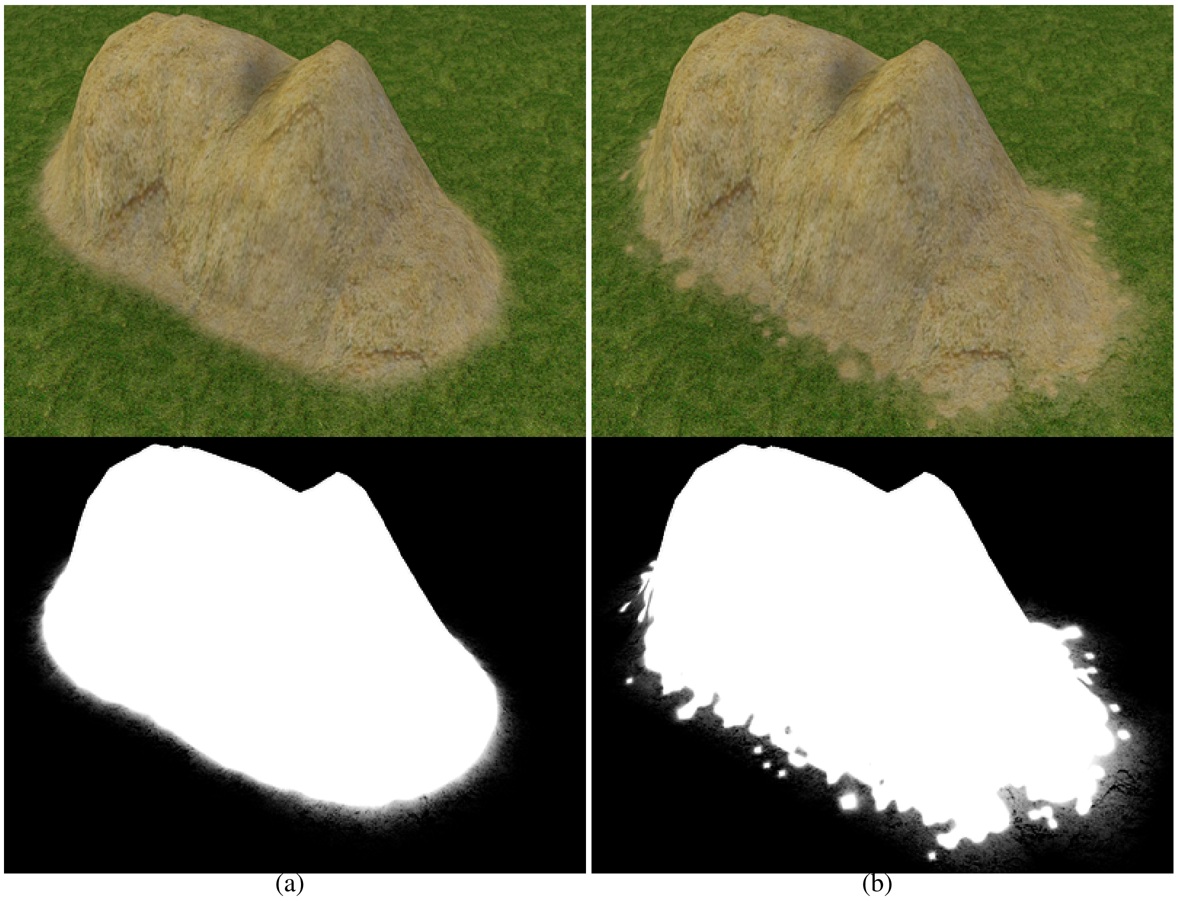


Figure. 4.1 A blend between a rock material and grass underlay using slope information for the blend weight with blend maps (a) and Dynamic Patch Transitions (b). The bottom row demonstrates the blend masks generated by each algorithm.

ther increases the resolution of the blend by addressing the broader shape and contour (b), generating stochastic, detailed material transitions of near-endless variations ranging from sporadic, intermittent transitions across flat plains to sharp, sudden transitions where abruptness is required.

The generation of the transitions themselves is entirely automatic, reducing the workload of the artist and making it suitable for procedural applications. Unlike the mapping of large, offline texture sets using virtual texturing, DPT does not require the storage or bandwidth necessary to stream such textures into memory in real-time. As DPT does not need any additional assets, it is particularly useful for environments where video memory is limited, such as hand-held devices and very large terrains. By combining patches of different sizes and parameters, complex transitions suitable for a wide range of terrain materials can be produced.

4.3 Related Work

4.3.1 Alpha Blending

There has been no direct research on contour synthesis for alpha blending although other works have indirectly touched on the topic. The contour of a blend is usually painted in by hand using image editing software or derived procedurally from a set of criteria using the topography of the mesh itself to determine how and where transitions occur. As discussed, the low resolution of the generated alpha mask alone is insufficient to depict varied and detail transitions so additional techniques are commonly used to circumvent these limitations to produce the necessary detail needed for high quality terrains.

Andersson's work with procedural shader splatting (Andersson, 2007) uses the height, slope and normal to generate the alpha masks and use procedural noise on a per-material basis to add detail to the transition where necessary. However, he notes that there are many terrain surfaces that cannot be generated in a purely procedural manner, especially when using only basic parameters of height, slope and normal. Using the example of open fields, he states that they are created artificially by the level designers in order to have full control over the shape and size. These manually generated masks are then stored in a sparse quadtree texture representation that stores only the unique 32x32 pixel tiles. Whilst this solution does allow for custom mask shapes for materials on geometrically flat regions of the terrain, the resolution (and thus detail) of the masks is limited in order to conserve space.

Zhang et al.'s work synthesizes a surface texture from the feature points and ridge lines of the terrain mesh that can then be mapped to the geometry (Zhang et al., 2008). The algorithm is capable of producing contours that follow the topography of the mesh but as with Andersson's work, such approaches fall short on regions of the mesh that are flat or continuous in topography as there is insufficient geometry to derive recognizable shape and variation to the transition.

Roupé and Johansson's work uses satellite imagery for the low frequency albedo information and blends high resolution detail masks to compensate for the low fidelity of the satellite data (Roupé and Johansson, 2009). The advantage of this technique is that the transitions and contours are both natural looking and readily available as real-world imagery is used directly. However, whilst such a technique is useful for visualization software used to explore a simulation of a real-world environment, video games usually require bespoke environments created to suite both artist and gameplay goals.

4.3.2 Tile-Based texture Mapping

The works of Lai et al. and Lai and Tai is capable of producing very detailed and realistic transitions but the perhaps biggest drawbacks are the limitations of tile-based texturing itself (Lai and Tai, 2008; Lai et al., 2005). When one is deciding on the size and number of tiles in a set, one must weigh up the costs and benefits between having tiles large enough to depict contour diversity and shape and the number of tiles required to avoid obvious repetition. Each tile must conform to the connectivity criteria required to reproduce transitions without displaying any seam artefacts so a uniform tile size is commonly used. As such, contour features must be self-contained in order to effectively connect to neighbouring tiles.

Larger tiles allow for greater diversity in shape as the contour itself is able to span a greater region of the terrain surface before properly terminating at tile boundaries. However, the number of variations of tile types needed to minimize repetition increases the amount of video memory in line with the square law increase compared to tiles of half the size. Smaller tiles allow for more diversity in tile variation for the same memory cost but reduce the variety and scope of contour shape. This limitation of contour shape presents itself in the form of a grid-like layout to the transition as if the transitions were sculpted around the square blocks that the tiles themselves are laid out on. This effect is further compounded when an insufficient number of tile variations is used in the set.

4.3.3 Virtual Texturing

As discussed in the previous chapters, the detail and variation of virtual texturing for transitions between materials containing no salient details is only limited by the time and skill of the artist. This flexibility can increase the workload of artists but transition synthesis algorithms can aid in reducing this workload. Procedural techniques for transition synthesis can be used to great effect to perform a lot of the heavy lifting when generating virtual textures as the results are rich and detailed where there is sufficient topography but such algorithms fall short on plains and other flat regions of the mesh. As such, these regions will usually be painted by hand which can be a laborious task for large terrains in the numbers required for video games.

4.4 Implementation

DPT aims to synthesize a high frequency blend mask for each material from the low frequency blend weights of texture splatting (Bloom, 2000) in real-time to produce rich and detailed transition between material textures. The outline of the approach is illustrated in Figure 4.2. The texture space (Figure 4.2(a)) is divided up into arbitrary-sized patches. The texture coordinates of the patch's centroid are assigned to each point within a given patch (b) whereupon each point samples the low frequency blend weight texture using these new texture coordinates. This sampled blend value, common to all points within a patch, is interpreted as the probability of a given patch appearing (c). These patches are then drawn or discarded in a probabilistic manner (Ferraris et al., 2012) to produce a binary patch mask (d). The patch mask is smoothed at a given point by bi-linearly interpolating the mask value of the patch upon which the point lies with that of the three adjacent patches. The point's position relative to the area enclosed by the four patches in the range $[0,1]$ is used as the interpolation amount between the four patch mask values (e).

This process can be repeated to synthesize multiple patch masks using different patch sizes and parameters to increase the detail and complexity of the transition. The resulting smoothed patch masks are accumulated and clamped within $[0,1]$ range (f) before optionally being further modulated by the material's high frequency blend map (Hardy and Mc Roberts, 2006) to produce the final blend mask (g). The details of the approach are given in the subsequent sections below.

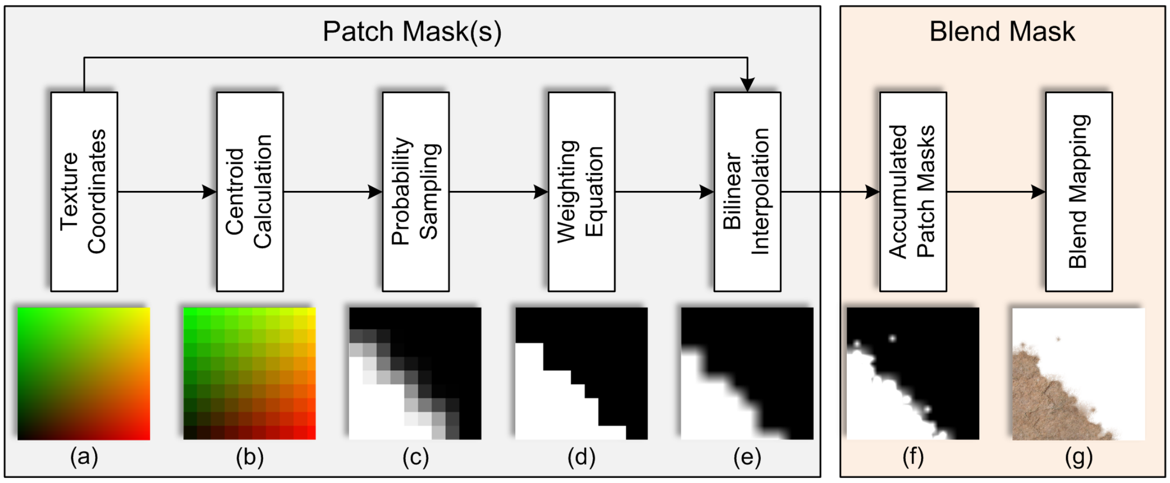


Figure. 4.2 Overview of DPT.

4.4.1 Centroid Calculation

The patch dimensions are specified by the artist as the two-dimensional vector $\vec{D}$ in the range $(0,1]$, which determines the percentage of a material's texture space a given patch spans. Assuming a bottom-left texture space origin, the origin of a given patch is defined by the vector $\vec{o}$ and is determined by Eq. 4.1 below, where $\vec{uv}$ is the texture coordinates of a given point on the surface of the terrain mesh. The centroid vector $\vec{c}$ in texture space can then be deduced by adding $1/2\ \vec{D}$ to this newly calculated origin vector.

$$\vec{o} = \vec{D}\lfloor \vec{uv}\frac{1}{\vec{D}} \rfloor \tag{4.1}$$

Figure 4.3 illustrates a point on a patch along with the three adjacent patches. To bilinearly interpolate the patch mask at the blue point, the mask values of the three adjacent patches are evaluated along with the mask value of the patch upon which the point resides. The centroids of the three patches adjacent to a given patch are calculated by adding either one or both components of vector $\vec{D}$ to the patch's centroid. For convention, the patch upon which a given point lies is denoted as the bottom left (*bl*) of the set of four patches, thus the centroids for the bottom right (*br*), top right (*tr*) and top left (*tl*) patches are calculated by adding $\vec{D}_x$, $\vec{D}_{xy}$ and $\vec{D}_y$ to $\vec{c}$ respectively to yield four centroid vectors $\vec{c}_{bl}$, $\vec{c}_{br}$, $\vec{c}_{tr}$ and $\vec{c}_{tl}$.

4.4.2 Probability Sampling

The probability for each of the four patches is obtained by transforming the centroids from the material's texture space into the blend weight texture's space and then sampling the

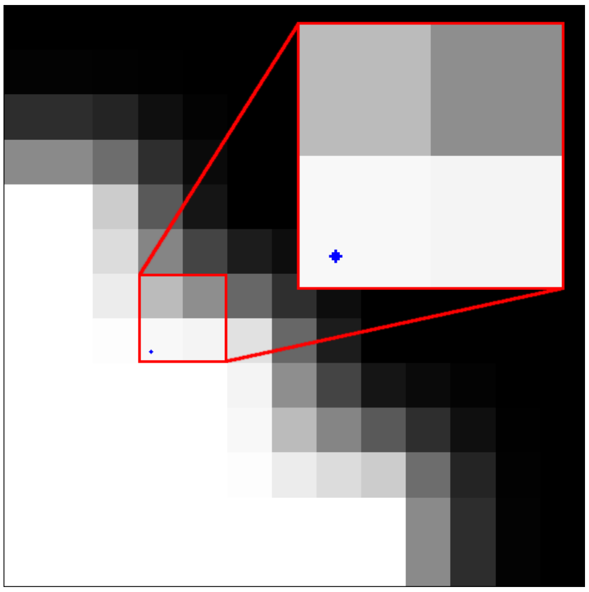


Figure. 4.3 At any point on a given patch the three adjacent patches are included to form a cluster of four patches.

blend weight with these newly transformed vectors. The random value is obtained for each patch following this same process, albeit sampling a noise function or texture instead. The four probability and random values are stored in the four component vectors $\vec{p}$ and $\vec{r}$ respectively.

The probability value of the patches is modulated by an artist-defined constant P. By raising the components of vector $\vec{p}$ to the power of P, the proliferation of patches can be sustained (when P is less than 1.0) or dampened (when P is greater than 1.0). For this see Figure 4.4. This constant is set for each patch mask allowing for artistic control over the size and shape of the transition. The constant R modulates $\vec{r}$ in an identical manner by raising $\vec{r}$ to the power of P in order to control the density of the patch mask.

4.4.3 Weighting Equation

To determine whether a patch will be drawn or be discarded altogether, the random value is checked against the probability of that patch appearing. Should the value lie equal to or under the probability, all points within that patch will contribute the value 1.0 to the binary patch mask, otherwise they will be discarded with no contribution. A simplified version of the weighting equation from FBPB (below) is used to obtain the mask values for the four patches given the probability and random value for each patch. The mask values of each of

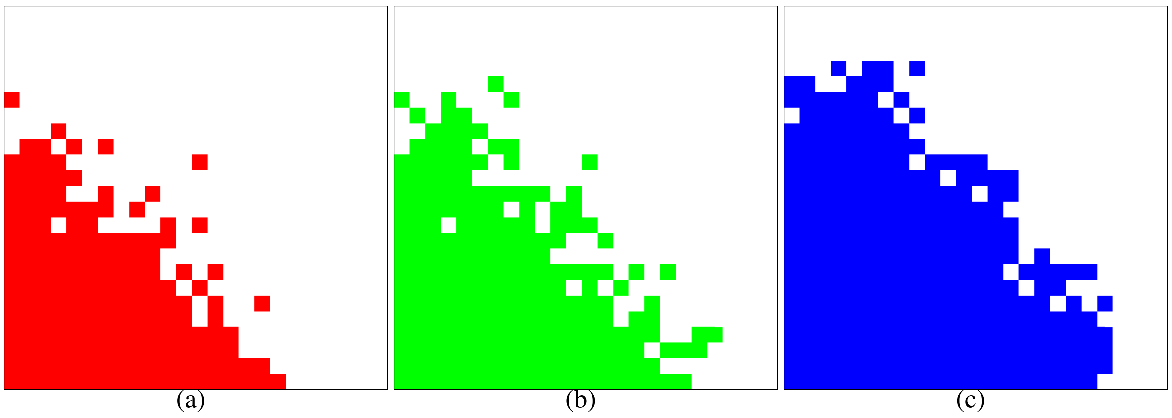


Figure. 4.4 Patch modulation when P is 0.25 (a), 1.0 (b) and 3.75 (c).

the four patches are stored in the four component vector $\vec{m}$.

$$\vec{m} = \operatorname{sgn}(\lfloor \frac{1}{\vec{r}} \vec{p} \rfloor) \tag{4.2}$$

4.4.4 Bilinear Interpolation

For any given point on a patch, the position $\vec{f}$ within the unit square enclosed by the four patch cluster (Figure 4.3) is calculated by taking the fractional components of the component-wise product of the vectors $\overrightarrow{uv}$ and $\overrightarrow{D}$. This new normalized position is then used to bilinearly interpolate the four patch mask values to obtain a smoothed scalar weight value w at that point $\overrightarrow{uv}$ using Equation 4.3 below.

$$\begin{array}{lllllllll} a & = & \vec{m}_{tl} & \cdot & (1.0 - \vec{f}_x) & + & \vec{m}_{tr} & \cdot & \vec{f}_x \\ b & = & \vec{m}_{bl} & \cdot & (1.0 - \vec{f}_x) & + & \vec{m}_{br} & \cdot & \vec{f}_x \\ w & = & \vec{t}_a & \cdot & (1.0 - \vec{f}_y) & + & \vec{t}_b & \cdot & \vec{f}_y \end{array} \tag{4.3}$$

The patch masks can be further shaped by modulating w with a constant scalar W that is set for each mask. The weight is raised to the power of W such that values less than 1.0 sharpen the mask contour whilst values greater soften it. Different patch masks can have different modulation amounts, allowing for a broad range of transition shapes. Figure 4.5 below illustrates this modulation, with the sand material on the left having a smoother, broader contour of the lower modulation amount, whilst the mud material has a sharper, more distinct contour from the higher amount.

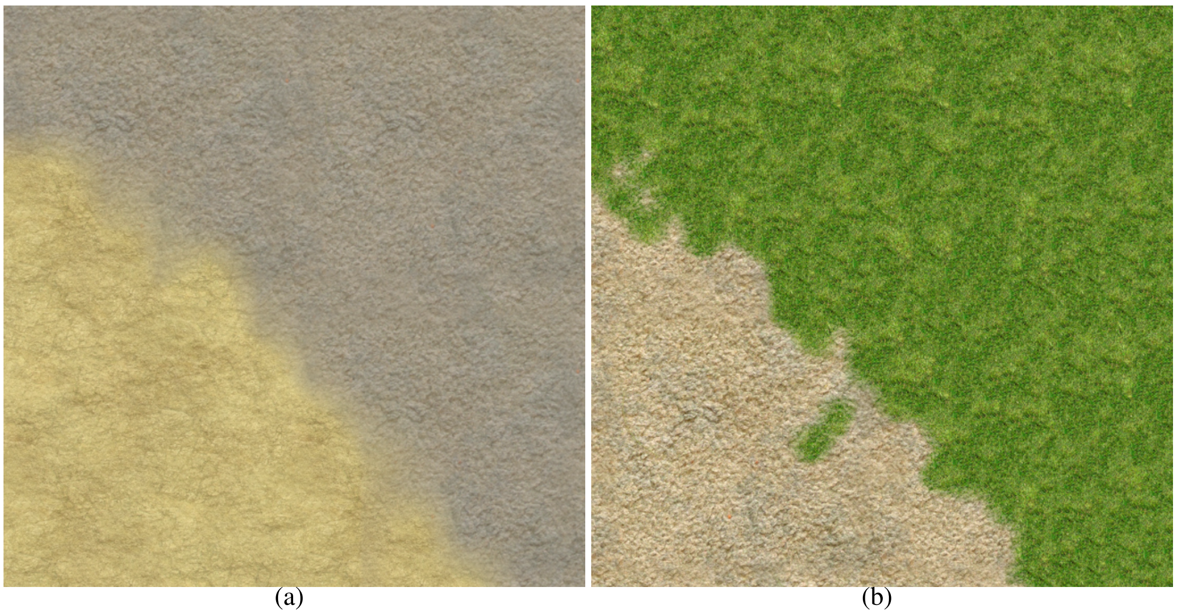


Figure. 4.5 Different materials with different weight modulation, from the smoother contour of the sand (a) to the sharper contour of the mud (b).

4.4.5 Blend Mask Composition

The steps in the grey shaded area of Figure 4.2 can be performed multiple times inside the fragment shader using different patch sizes and parameters to produce different patch masks for more complex and detailed blends. These patch masks are accumulated to compose the final blend mask. In Figure 4.6, the first patch mask was synthesized using a patch size of 0.9 (a) and the second mask using a size of 0.045 (b), thus the steps in the grey shaded area of Figure 4.2 are performed twice. These two masks are accumulated to yield a final blend mask weight of b at any given point (c).

To add further details to the transition (as seen in Figure 4.7), the material's high frequency blend map (a) is used to modulate the lower frequency blend mask b to produce the final blend weight (b). This weight is then used as the interpolation amount between the material and the underlying texture (c).

The steps to compose the final blend mask (the light brown shaded area in Figure 4.2) are illustrated in the pseudo-code below (Algorithm 2), where j is the number of patch masks to accumulate and PatchWeight is a function that calculates the patch weight value w from the artist supplied size and modulation constants for each mask (as described in previous subsections, indexed by the variable i). For materials containing high frequency details, the blend mask is then reduced to $[0,1]$ range before the material's blend map is applied with

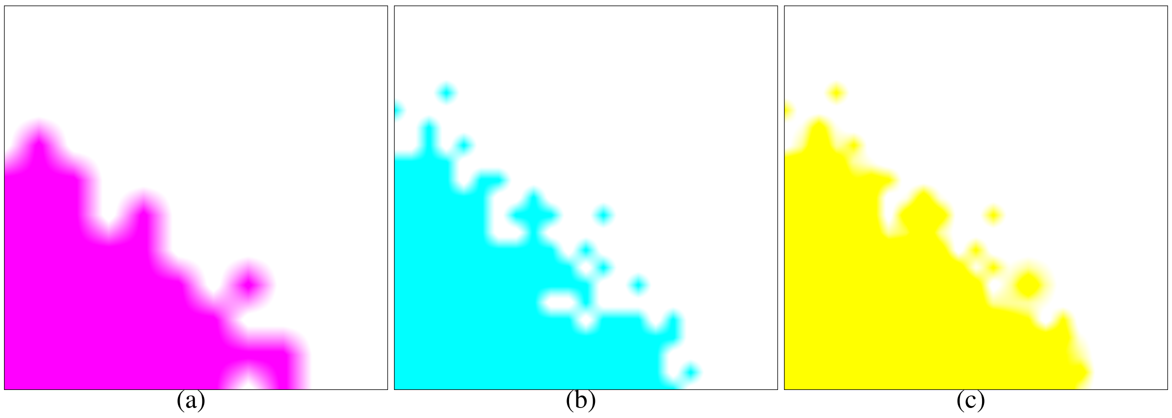


Figure. 4.6 The smoothed masks of different patch sizes and parameters (a,b) are combined together to produce the final blend mask (c).

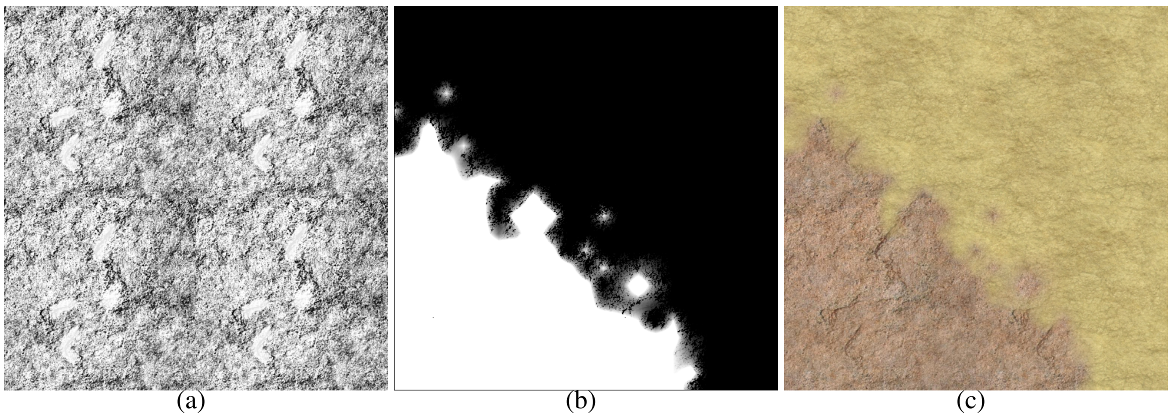


Figure. 4.7 The material's blend map (a) can further modulate the blend mask (b) to produce the final blend (c).

the function BlendMap (Hardy and Mc Roberts, 2006).

4.4.6 Parallax Mapping

Materials using displacement effects such as parallax mapping can benefit from interpreting the smoothed patch masks as height information. This is particularly useful for material textures that contain little in the way of meaningful height data, such as the textures used throughout this paper. In Figure 4.8, the rock material has been blended three times, first without parallax mapping (a), then with parallax mapping using the texture's height channel (b) and, finally, using the smoothed patch masks as height maps (c). As the rock texture has little in the way of distinct height details, the magnified element of (a) and (b) looks very

Algorithm 2 Overview of Blend Mask Composition

```
$b \leftarrow 0.0$
**for** $i = 0 \rightarrow j$ **do**
    $b \leftarrow b + \text{PatchWeight}(i)$
**end for**
$b \leftarrow \min(b, 1.0)$
$b \leftarrow \text{BlendMap}(b, \overrightarrow{albedo_a})$
```

similar, with the only difference being the noisy peppering of parallax offsetting from the texture's height map in the latter. On the other hand, the lump of rock synthesized with the DPT blending process (c) has a recognisable shape and depth to it.

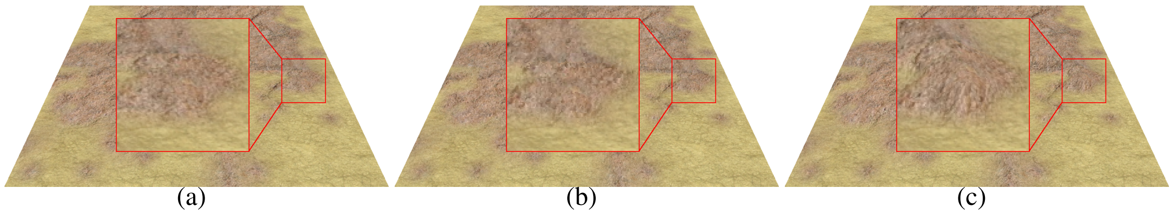


Figure. 4.8 A transition between rock and sand (the red square is under magnification) using no parallax mapping (a), parallax mapping using a height texture (b) and, finally, using the height and blend mask (c).

The parallax mapping technique is applied after the final blend mask has been synthesized so that the albedo and normal maps are properly offset by the height data. As seen in the code listing below, the heightfield h accumulates the patch masks in a similar manner to the blend mask, albeit each patch mask is being modulated by the constant H (set by the artist for each patch mask in the range $[0,1]$), allowing masks to contribute different amounts to the heightfield. In this particular code listing (Algorithm 3), Parallax is a function that samples the material's height field (typically stored in the normal texture) and modulates it with the synthesized height field before performing the parallax operations (omitted for clarity).

4.4.7 Minification Correction

As the DPT process alters the shape and content of a texture after the texture has been sampled, texture mipmapping no longer eliminates the aliasing artefacts from undersampling errors so, instead, this undesirable effect must be corrected manually. For materials containing particularly small patch sizes, these artefacts manifest themselves as "swimming

Algorithm 3 Parallax Mapping with DPT

```
$b \leftarrow 0$
$h \leftarrow 0$
**for** $i = 0 \rightarrow j$ **do**
    $k \leftarrow \text{PatchMask}(i)$
    $b \leftarrow b + k$
    $h \leftarrow h + kH$
**end for**
$b \leftarrow \min(b, 1.0)$
$h \leftarrow \min(h, 1.0)$
$\overrightarrow{plx} \leftarrow \text{Parallax}(\overrightarrow{uv}, h, \overrightarrow{Eye})$
$\overrightarrow{albedo} \leftarrow \text{TextureSampler}(\overrightarrow{plx}, Albedo)$
$\overrightarrow{normal} \leftarrow \text{TextureSampler}(\overrightarrow{plx}, Normal)$
$b \leftarrow \text{BlendMap}(b, \overrightarrow{albedo_a})$
```

pixels” as the camera moves around when the screen-space patch size approaches one pixel. An example of this is shown in Figure 4.9. Here, the camera has moved backwards slightly between the two images which has caused some of the high frequency details of smaller patches to distort or be removed completely. Whilst it is difficult to spot these artefacts in a still image, this effect is pronounced when viewing animated images of camera movement.

To eliminate these aliasing artefacts, a transition band is used to derive a distance value d in the range $[0, 1]$ which is used to modulate the translucency of a given patch, as described in Equation 4.4 below. Here, the value z represents the distance of the fragment from the viewer, $Offset$ is the distance at which the band starts and $Size$ is the distance that the band spans.

$$d = \max(\frac{z - Offset}{Size}, 0.0) \tag{4.4}$$

The value d is then used to modulate a given patch mask value w, causing it to fade over the duration transition band before being removed completely from the composition of the final blend weight. This can be seen in Figure 4.10 where the high frequency pebble details are displayed when the terrain is viewed from closer distances but gradually faded over distance to be removed entirely at further distances. The size and offset of transition bands will be entirely dependent on the size of the patch and the size of texture space relative to object/world space (in this particular example, an offset of 75 units was used with a band size of 150 units).

For larger patches, we cannot resort to transitioning them all into translucency as we will

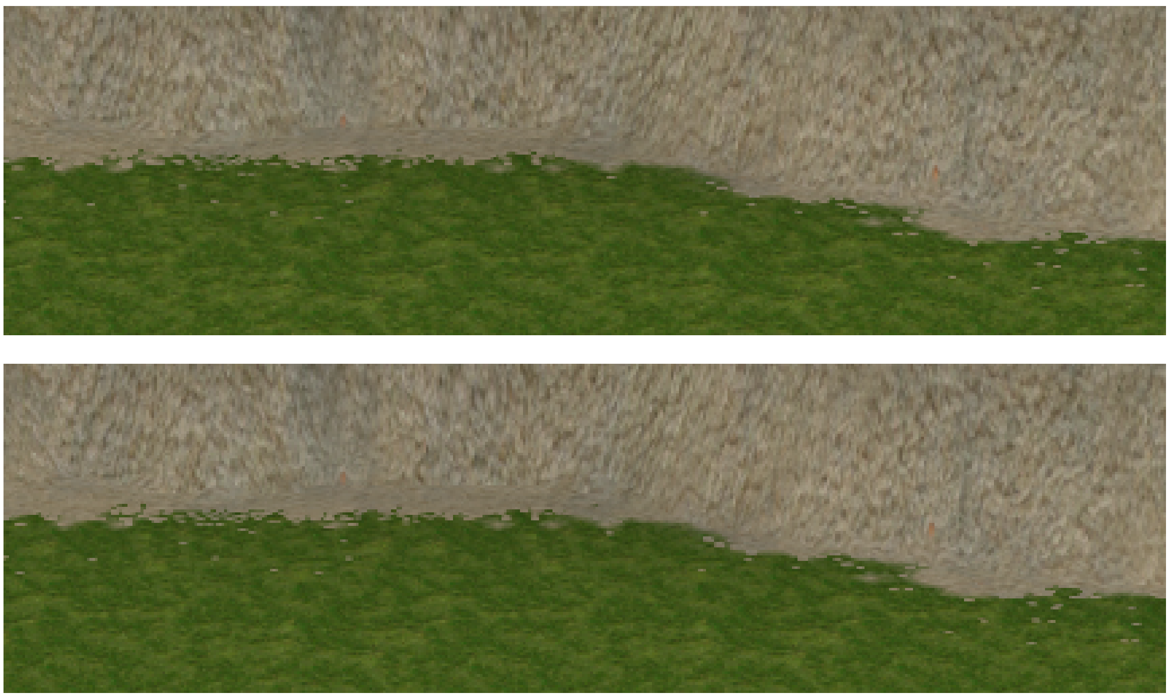


Figure. 4.9 Camera movement can cause small patches to exhibit undersampling errors.

be left with none of the original material. However, as the undersampling errors for such patches occur at greater distances we can simply transition the final blend mask b between DPT and a simple linear blend using the technique described above. As the details of DPT are difficult or impossible to pick out over large distances this also has the advantage of reducing processing overhead by falling back to the simpler linear blending process when such details offer negligible or no benefit to the scene. In Figure 4.11, we can see a terrain feature viewed from increasing distances (100, 500 and 1000 units respectively). Here, the details of DPT (a) become increasingly difficult to spot compared to the linear blend (b) such that in the final image the differences are near impossible to identify at run-time.

4.4.8 Using DPT

DPT is a texture space effect so the size and parameter range will be entirely dependent on the size of texture space and the texture information contained within. However, in our use of DPT we have developed a work flow that should translate to other materials regardless of the actual texture and parameter settings used. Consider Figure 4.12, which shows the example low resolution weight texture (a) along with a grass underlay texture that span three units of texture space along the x-axis and one unit along the y-axis.

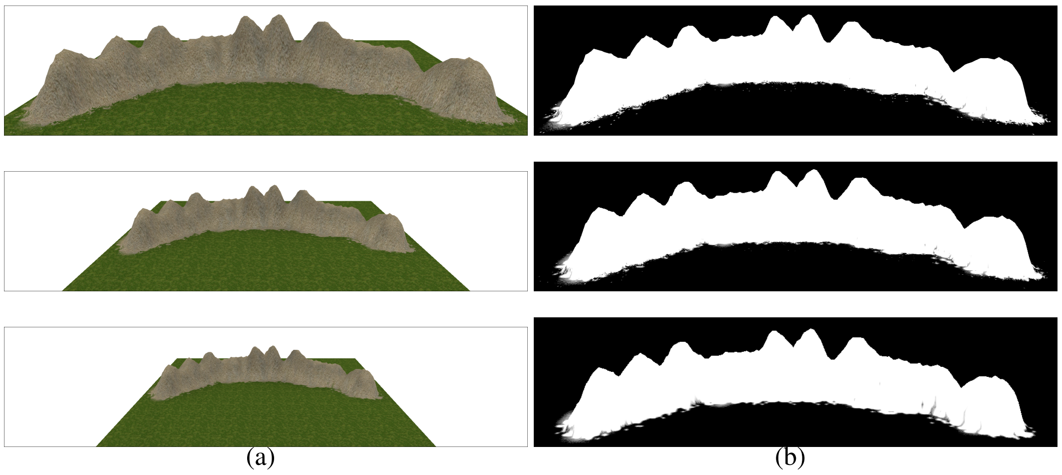


Figure. 4.10 A terrain as viewed from a distance of (from top to bottom) 80, 135 and 170 units from origin (a) along with the associated blend weights (b).

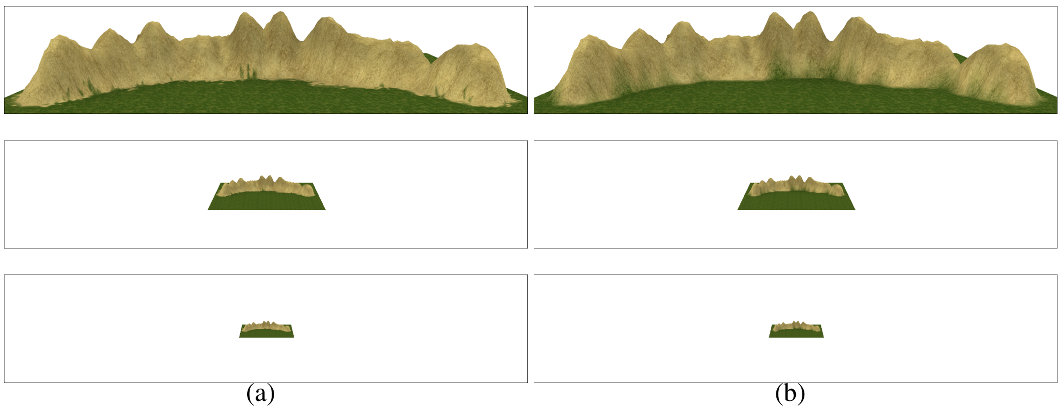
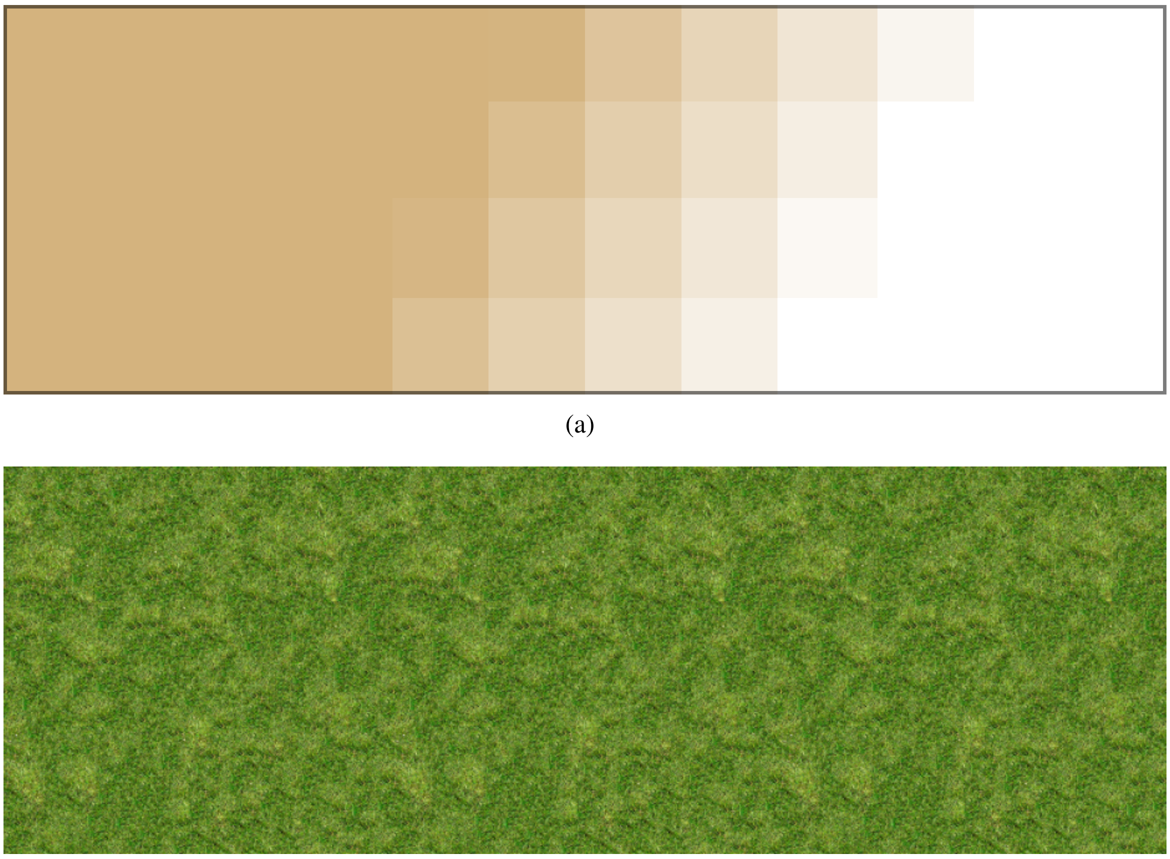


Figure. 4.11 As the distance from the viewer increases, the details of DPT (a) become increasingly difficult to pick out over a simpler linear blend (b).

DPT can work with any number of patches although it was found that two is sufficient for most cases. The first patch is used to lay down the body of the transition so we will use a relatively large size of 5 units and a probability modulation constant P value of 6.5 to cover a broad area of the transition (Figure 4.13(a)). As the material used is sand, we would expect mostly mid and low frequency details so we will soften the patch with a weight modulation constant W value of 0.5 (Figure 4.13(b)).

With the first patch providing the bulk of the transition, we will use the second patch to add subtle detail and shape to the transition contour. We achieve this by using a far smaller patch size of 12.5 units but with a slightly higher P value of 7.1 to extend the range of this patch slightly beyond the first patch (Figure 4.14(a)). We add more definition to this patch


(a)

(b)

Figure. 4.12 An example weight (a) and underlay texture (b).

layer by increasing its W value to 3 to sharpen the patch shapes. This combination of sharp, mid frequency detail from this patch with the softer, low frequency detail of the first patch results in a transition that has far more contour detail and variation from the low resolution weight texture that can be obtained from standard texture splatting.

This method of using smaller patches to modulate the contour shape of larger patches is sufficient for most material types, although for materials composed of high frequency detail one patch may be sufficient (Figure 4.15(a)). However, excessive W values can produce geometric artefacts as a by-product of the bilinear interpolation stage of DPT where adjacent patches cause thin spines (4.15(b)). This is because the resulting patch shape from high W values is too small compared to the unmodulated patch shape. In such cases, one can either instead use smaller patch sizes, reduce the strength of W modulation or ensure that the proliferation of such patches does not greatly exceed that of any other larger patches.

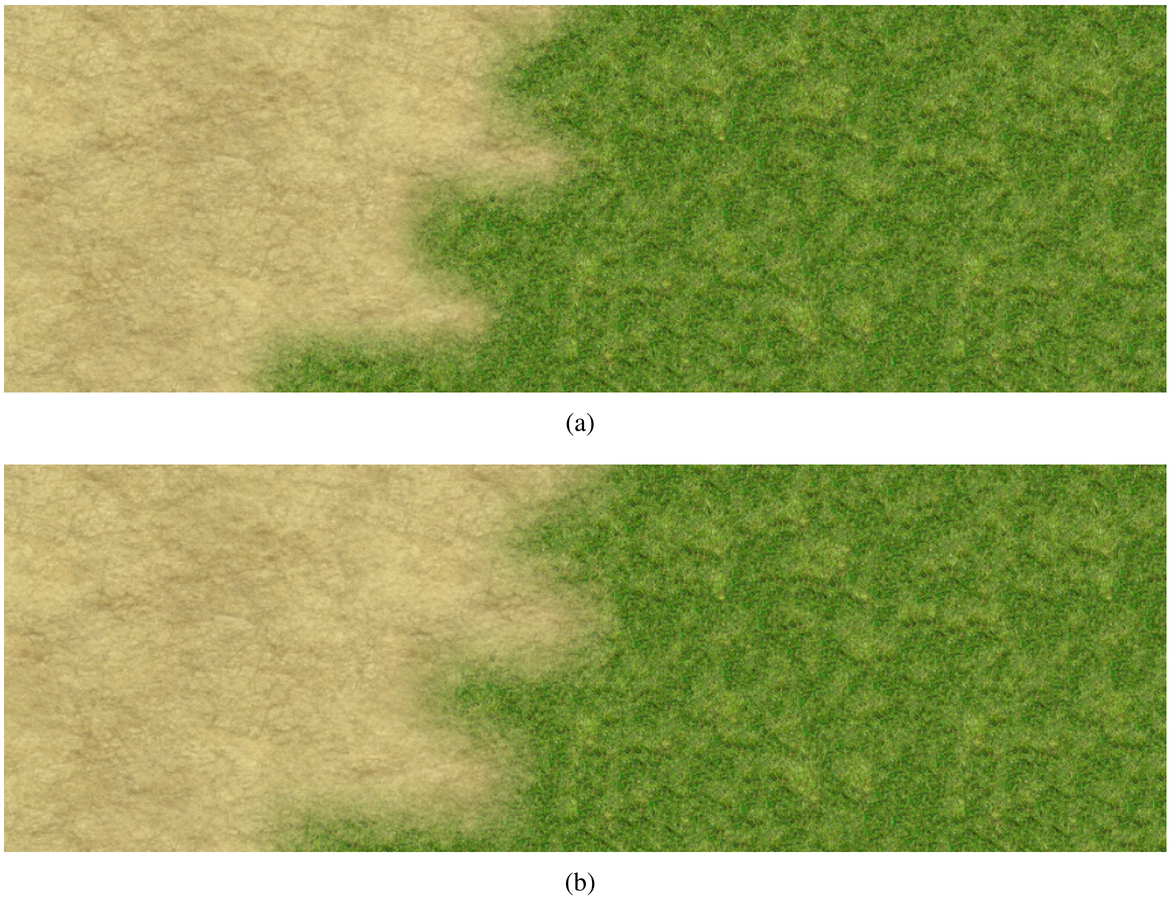
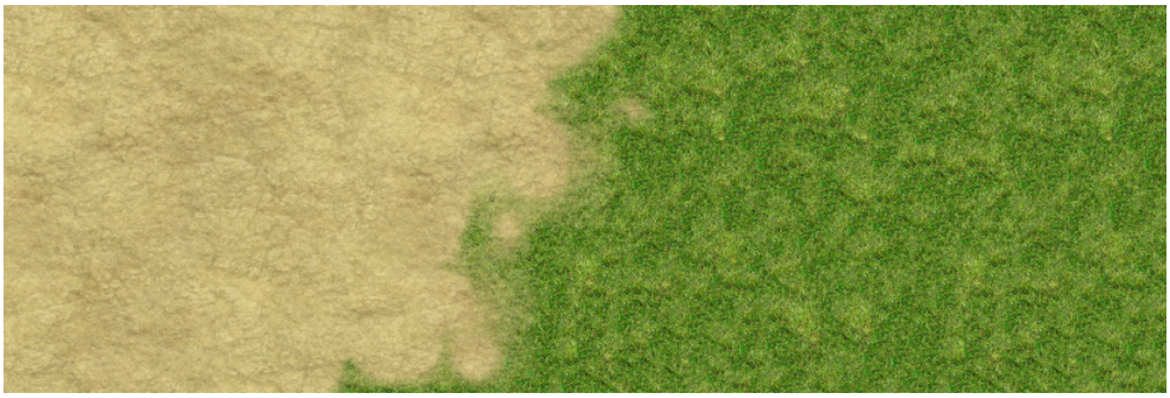


Figure. 4.13 A large initial patch size (a) further softened with W modulation (b).

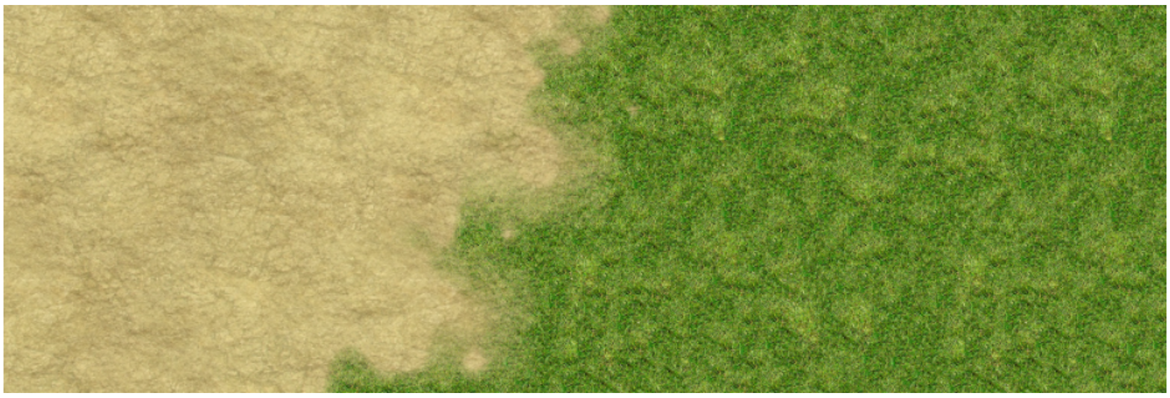
4.5 Results and Analysis

The figures in this (and the following) section demonstrate a single terrain material being blended over an underlay material using DPT and blend maps. For DPT, two patch masks were accumulated to synthesize the final blend mask. In Figures 4.16 to 4.21 a low resolution texture was used as the blend weight source (where a grid of 4x4 weight texels spanned one unit of the material's texture space) whereas Figure 4.1 instead used the slope information of the underlying geometry.

In all Figures it can be seen that whilst blend maps help to add high frequency detail to the transition, the contour of the transition remains unchanged. On the other hand, DPT utilizes blend maps for finer details but also adds stochastic detail and definition to the broader shape of the transition. The flexibility of DPT by using different patch and modulation settings allows for a wide range of materials to be represented. In Figure 4.16, DPT adds subtle definition and variation to the sand's transition, helping break up the uniformity of the



(a)



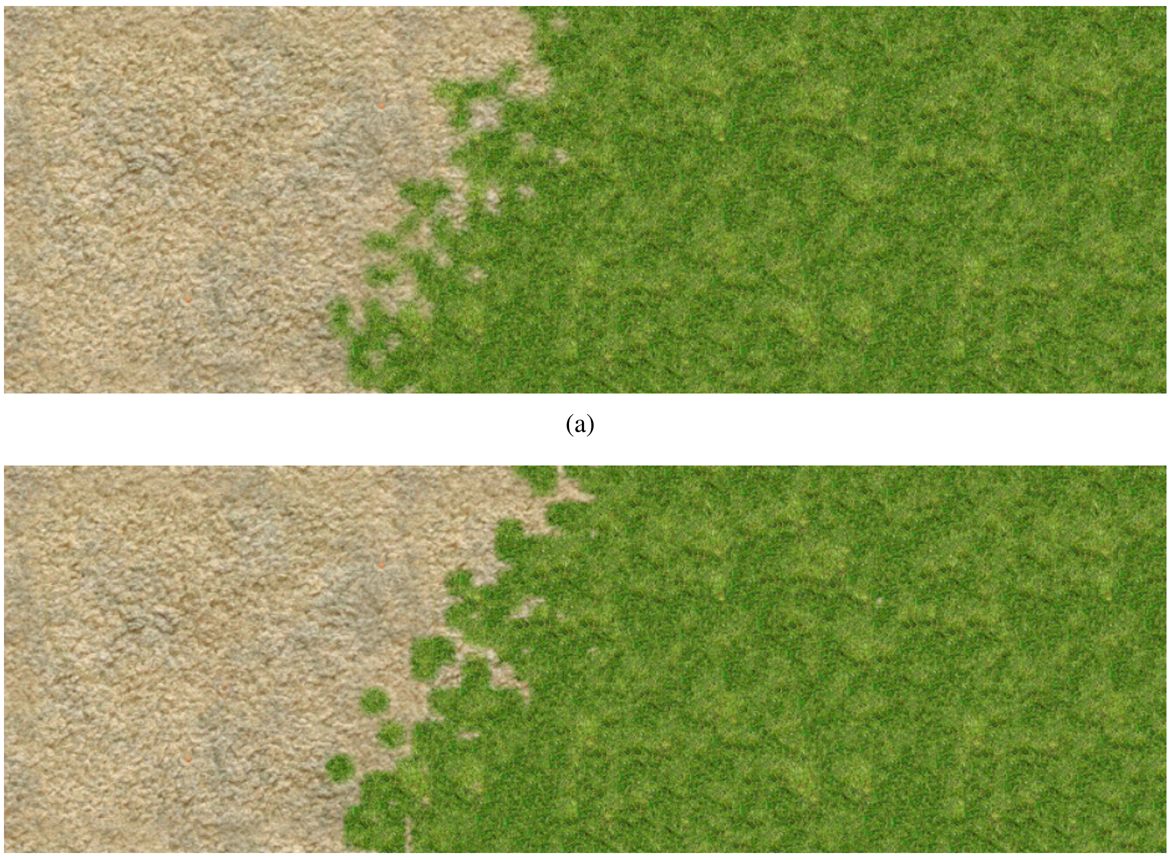
(b)

Figure. 4.14 A smaller size for the second patch (a) further sharpened with W modulation (b).

blend. The sharper contour of the DPT transition in Figure 4.17 is achieved by using higher W values for each patch mask with the different patch masks adding detail and irregularity to the transition. In contrast, blend maps produce the same uniform and monotonous blend contour shape regardless of material.

Figures 4.18 and 4.19 demonstrate DPT's effectiveness with materials containing non-salient details. Whereas these details would be too ambiguous and indistinct for techniques such as FBPB (Ferraris et al., 2012), they contain enough contour information to exhibit the translucency artefacts demonstrated by blend maps transitions. DPT masks these artefacts by creating the illusion of contour preservation to produce more sporadic, intermittent transition rather than the faded, uniform transitions of blend maps.

The uniformity of a blend's contour becomes apparent when geographical features protrude from a ground plane, such as in Figures 4.22 and 4.1. In Figure 4.22, blend maps exhibit strong banding artefacts around the hill, causing it to float artificially above the



(a)

(b)

Figure. 4.15 A single patch can be sufficient in some cases (a) although excessive W modulation can produce artefacts (b).

ground. DPT's stochastic and detailed blend contour conveys a greater sense of shape to the hill, allowing it to "sit" more realistically with the grass beneath. In Figure 4.1 the use of height/slope information rather than a texture as the blend source does help to break up the banding of blend maps but the uniformity of the contour shape still looks unnatural. The intermittency of the DPT blend obfuscates the uniformity of the slope information around the base of the hill, conveying far more detail and rooting it naturally to the grassy ground beneath.

In Figure 4.20, the transition uses FBPB for the salient cobble stones and either DPT or blend maps for the grey non-salient mortar. Whereas the uniform contour of blend maps exaggerates the translucency artefacts of the non-salient details, DPT completely reshapes the contour into one that is sharper and more ragged. This helps break the uniformity of the transition giving it a more natural, more realistic look. When used in conjunction with one another, FBPB and DPT complement each other, with the former accommodating the more prominent, salient details of a material and the latter introducing variation to the shape and contour of the non-salience that cannot be easily identified by FBPB.

Figure 4.21 illustrates the limitations of DPT. Here, the cobble features are too large and salient for DPT to handle correctly as DPT does not take into consideration the topography of the material. As such, the patch shapes bear no correlation to that of the cobble features resulting in a contour shape that is nonsensical and unrealistic. In comparison, blend maps handles this particular material with far more plausibility, preserving the feature shapes without disrupting the overall detail of the texture. Instead, such materials would be better handled by an algorithm such as FBPB which can take into consideration the size and shape of salient features and ensure that the integrity of such features is retained at the transitions.

4.6 Performance Analysis

The performance test case consisted of a multi-pass rendering approach where two terrain materials were blended over a base texture to cover a 256x256 vertex terrain mesh completely (with no culling or geometry level of detail optimizations for the mesh itself). The first pass was an early z pre-pass that mapped the base texture onto the mesh with a subsequent blending pass for each terrain material. The material and base textures were mipmapped and sampled with anisotropic filtering. The camera was positioned to fill the viewport (measuring 1920 by 1080 pixels) entirely with a distribution of near and far fragment data. The hardware upon which the tests were performed was a Radeon HD6970M GPU on a 2.2 GHz CPU laptop with 8 GB of RAM running Microsoft Windows 7.

The two algorithms tested were blend maps and DPT blending. In particular, DPT was tested accumulating one and two patch masks for each terrain material. For each algorithm, three blends were performed: a straight blend, where only the blending process is executed, a normal mapped blend and a normal & parallax mapped blend. As the technique for eliminating minification distortion in DPT can be extended to switch over to linear blending for distance fragments, this basic optimization was used for both algorithms.

Referring to Table 4.1, it can be seen that DPT performs only slightly slower than blend maps in all cases which is to be expected considering the difference in complexity and detail of the transitions. As the normal and parallax mapping techniques were applied, the relative performance of DPT to blend maps remained the same, with the single patch mask blend increasing slightly as the overhead of normal and parallax mapping begins to become the bottleneck. After averaging the performance of DPT in relation to blend maps for each test case, DPT typically operates at 98% (one patch) or 95% (two patches) of the performance of blend maps. Nevertheless, DPT has a number of advantages over blend maps. The performance is competitive with blend maps without any extra video memory overhead or asset pre-processing. DPT is capable of producing transitions of far greater variation than blend maps. As the generation of these stochastic transitions is entirely automated at run-time, artists need not manually create transitions for each and every shape and variation for any given material combination.

Algorithm	Straight		Normal		Parallax	
	FPS	MPS[1]	FPS	MPS[1]	FPS	MPS[1]
Base Texture	650	1347.8	-n/a-	-n/a-	-n/a-	-n/a-
Blend Maps	333	690.5	322	667.7	311	644.9
DPT 1 Patch	324	671.8	315	653.2	305	632.4
DPT 2 Patches	316	655.3	305	632.4	297	615.9

[1] Pixel throughput (in megapixels per second)

Table 4.1 Performance results.

4.7 Conclusion and Future Works

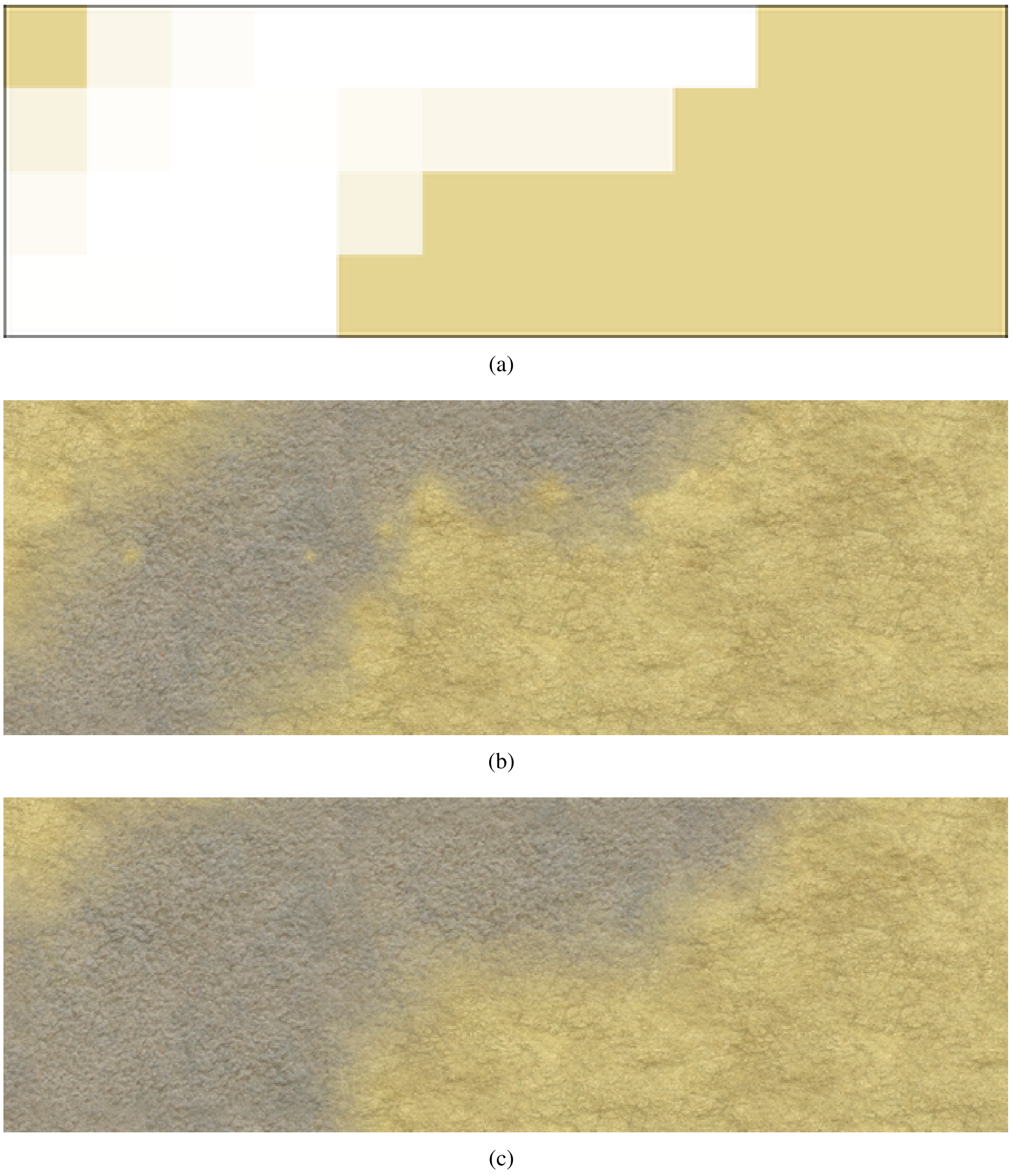
A novel technique DPT has been proposed for creating irregular blend contours of near-endless variation in real-time that offer far greater resolution using low resolution blend weights than conventional techniques. This is achieved by breaking up the contour of the blend to create detailed and stochastic transitions between materials. This process is entirely automated and offers consistent, competitive performance for no additional video memory overhead. Artists have great freedom to create a wide range of material transitions through patch mask accumulation and parameter modulation. As DPT does not require any pre-processing of assets, any non-salient textures can be used, allowing for instant previewing and appraisal of results.

Performance and Scalability: The performance of DPT is 98% and 95% that of blend maps for one and two iterations accordingly so, the technique performs well given the extra detail added to the transition contours. As DPT does not require any additional assets for the materials used it can scale adequately to terrains of any size. However, the algorithm operates on a per-material basis so if many materials are used the technique can become fill-rate bound, especially if a large number of patch masks are used per material. In practice, the number of fragment shader cycles dedicated to DPT can be reduced through dynamically reducing the patch mask count on a distance basis and/or minimizing the fragment overdraw through masking the terrain on a per-material basis as with the work of Andersson (2007).

Fixed Costs: As DPT does not require any additional per-material assets, the only additional resource is the noise function. This can be either a texture or an algorithm, with the former being favoured for simplicity and efficiency to free up algorithmic logic units for other duties. As with most of the techniques surveyed in the Literature Review chapter, DPT requires a GPU with programmable stages, although considering the proliferation of such units in the consumer market this should rarely be an issue. The technique makes heavy use of the fragment shader stage of the GPU pipeline so the performance will be a reflection of the sophistication of this stage. For dynamically controlled patch mask iterations, the GPU hardware will require hardware paths for dynamic flow control as specified in Shader Model 3.0 (Rege, 2004).

Transition Variation and Detail: DPT works well with a variety of materials without defined features, even those with features that are smaller than the patch sizes. The diversity of the results is limited only by the randomness of the noise function as textures or algorithms that are biased to a particular range of values (either due to insufficient texture resolution or probability distribution) will result in noticeable patterns emerging. This effect can be reduced by increasing the number of patch masks generated for a particular material but if greater shape and control is required for contour shape then quilted decal patches can be used instead of the bilinearly interpolated patches used for the majority of the results.

Future Works: Future work will look into incorporating elements of FBPB's feature awareness to preserve salient contours of the materials themselves to further minimize artefacts caused by abrupt changes in blend weighting mid-way through such contours. Finally, research will be conducted into re-synthesizing the tiled material textures in order to address the inherent regularity of tiled textures to produce truly rich and diverse surfaces at the material level. Furthermore, a user study will be performed as a priority to quantitatively show the improvement of DPT over existing techniques in terms of the resulting transitions and artist work flow.

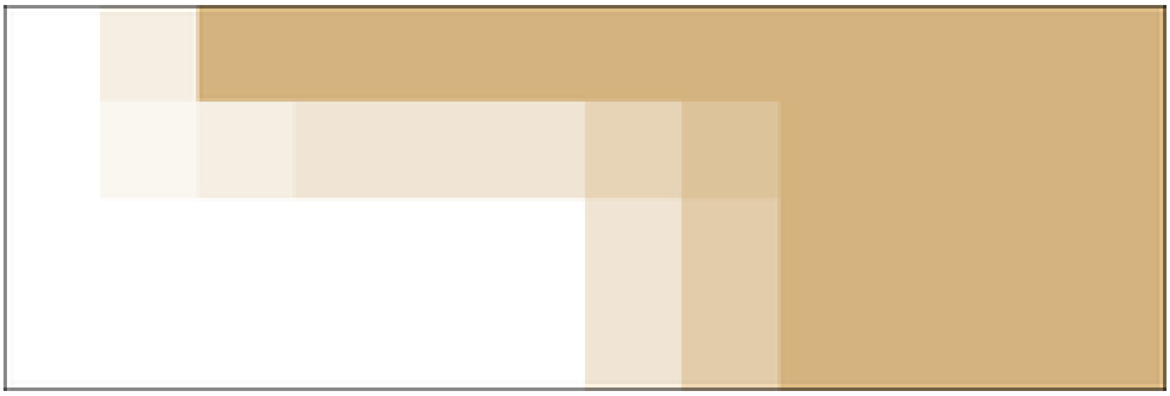


(a)

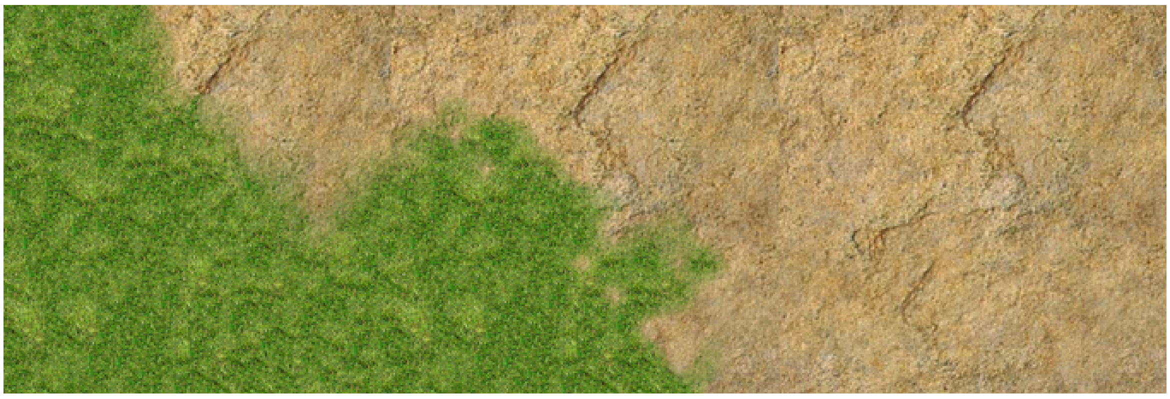
(b)

(c)

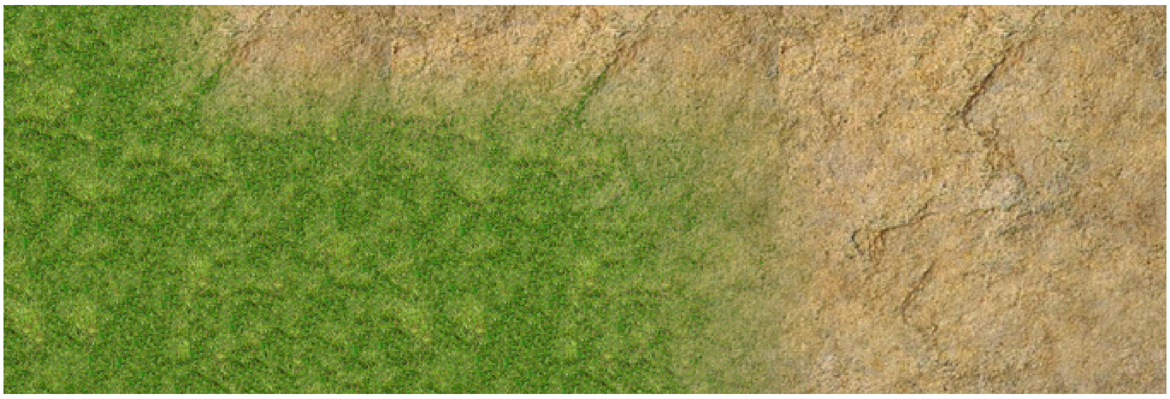
Figure. 4.16 A blend between a sand material and concrete underlay with a low resolution weight texture (a) using DPT (b) and blend maps (c).



(a)

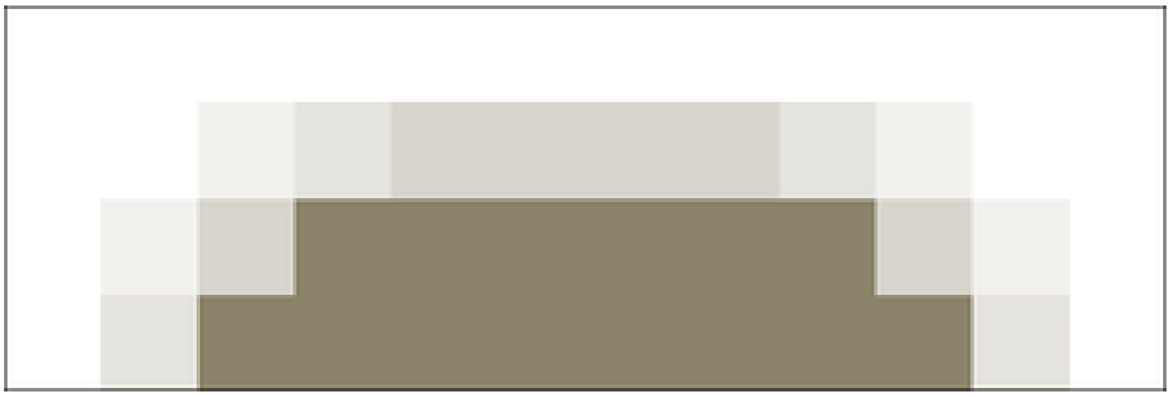


(b)

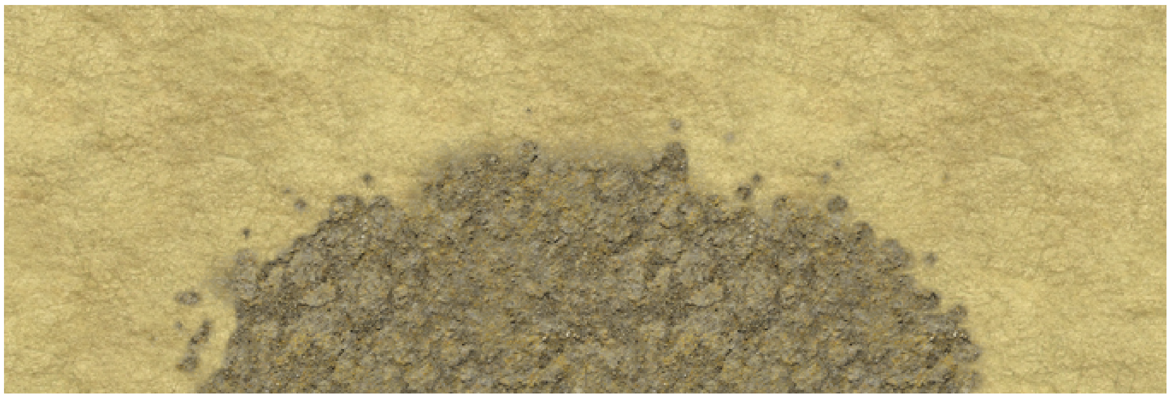


(c)

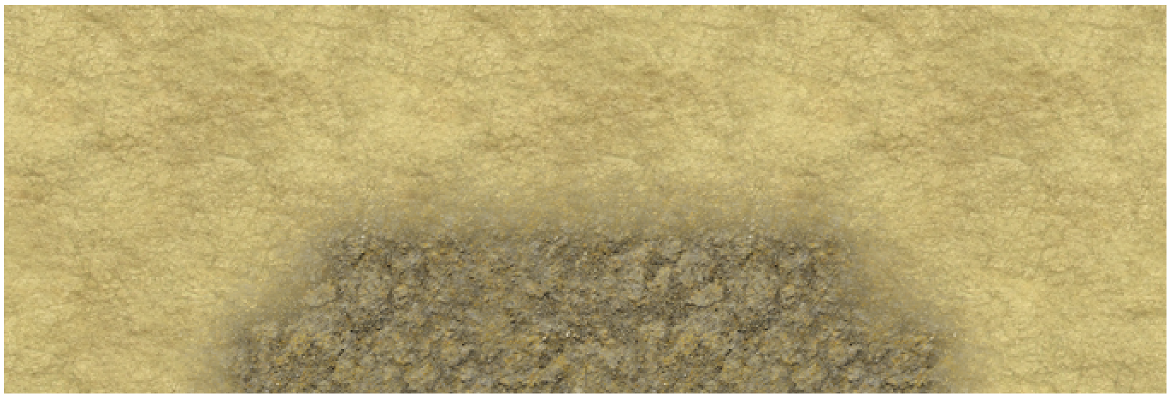
Figure. 4.17 A blend between a rock material and grass underlay with a low resolution weight texture (a) using DPT (b) and blend maps (c).



(a)

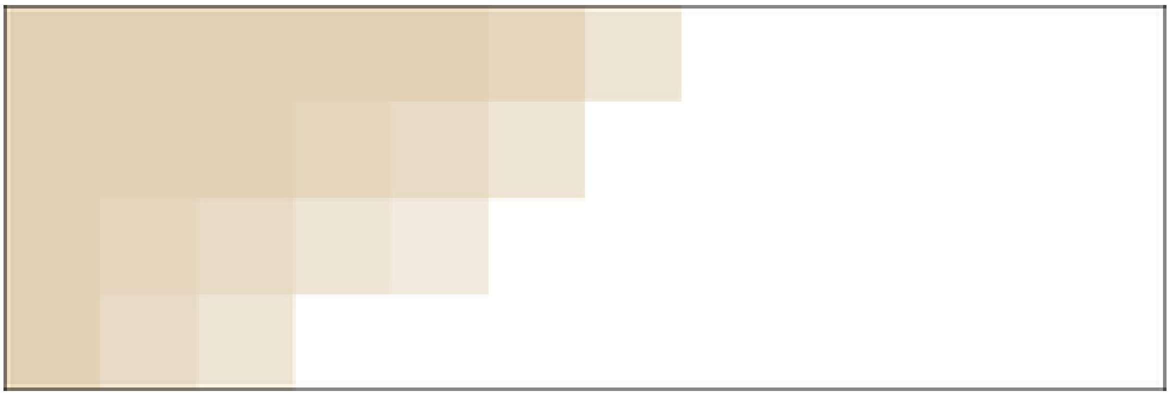


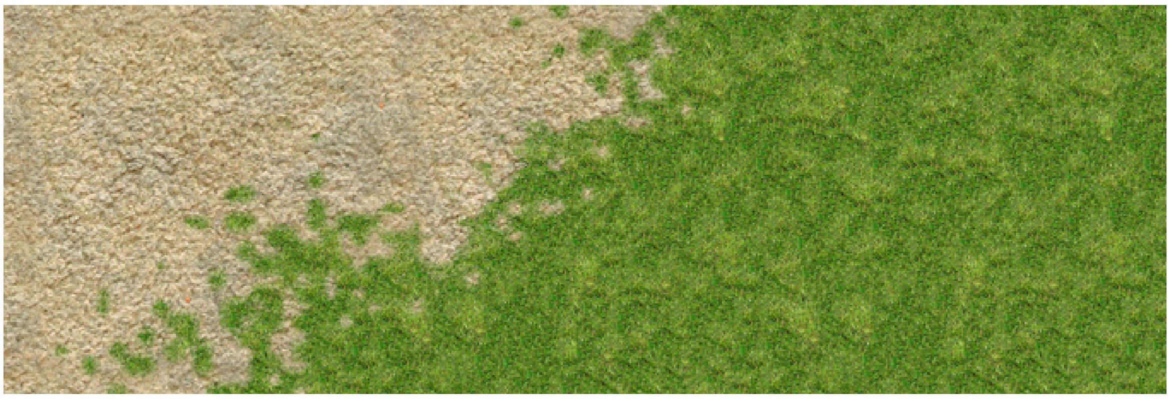
(b)



(c)

Figure. 4.18 A blend between a mud material and sand underlay with a low resolution weight texture (a) using DPT (b) and blend maps (c).


(a)


(b)

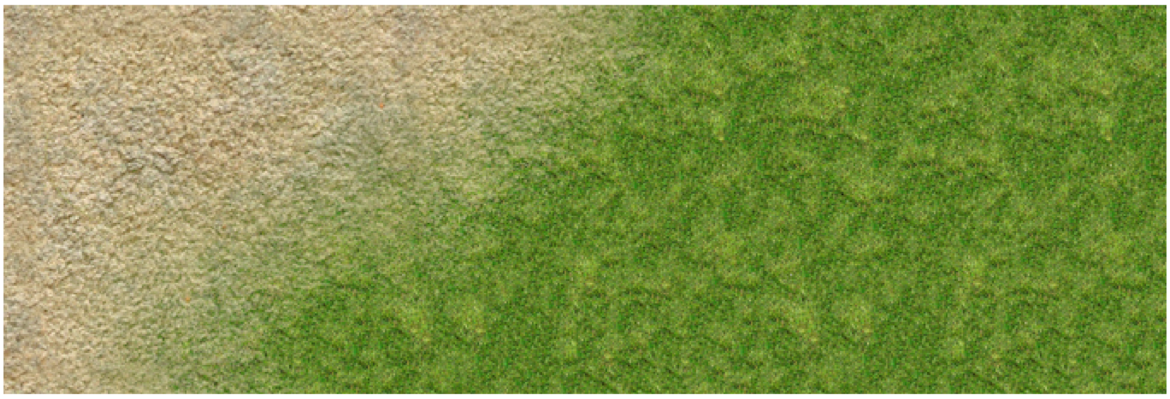

(c)

Figure. 4.19 A blend between a pebble material and grass underlay with a low resolution weight texture (a) using DPT (b) and blend maps (c).

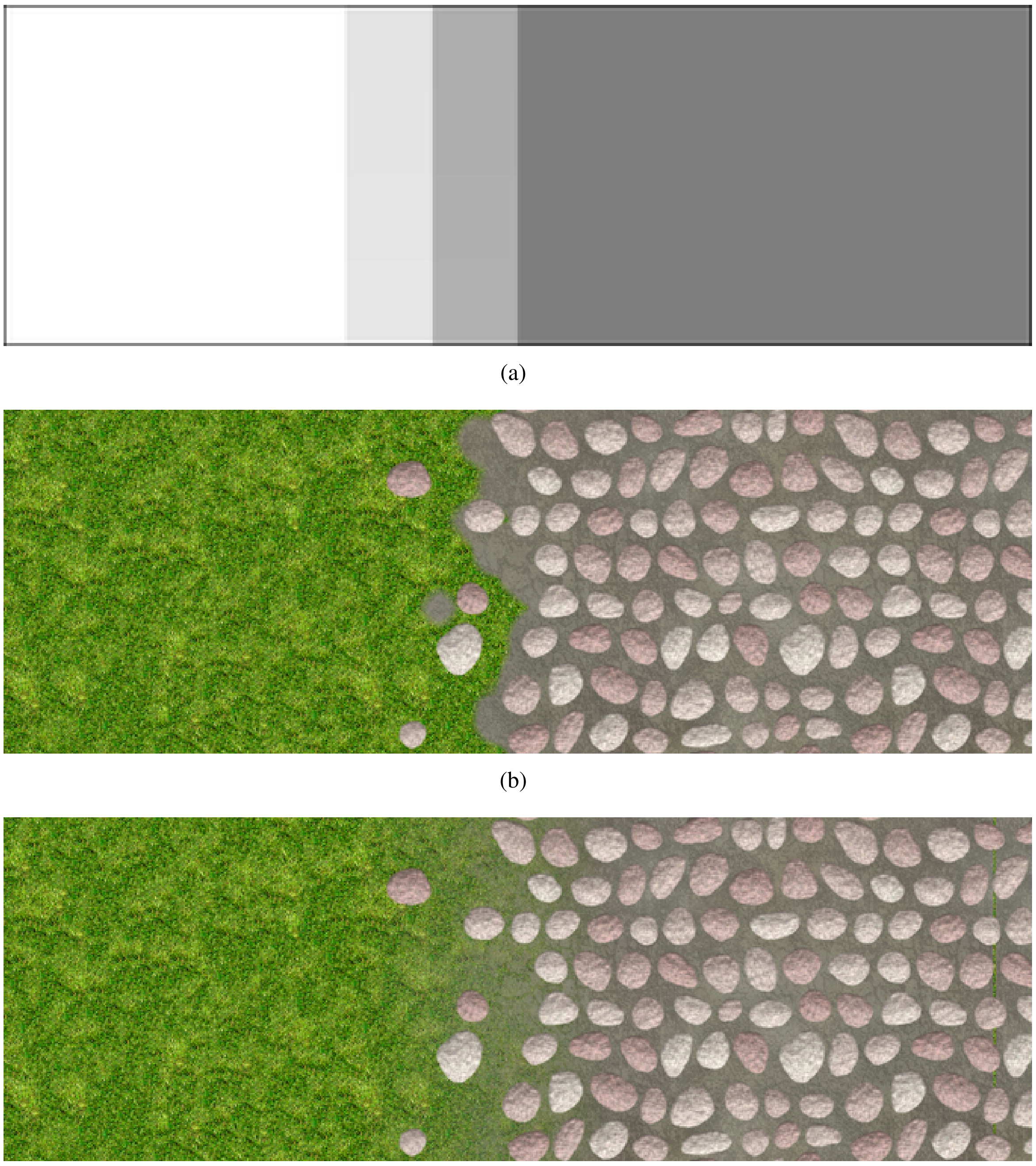
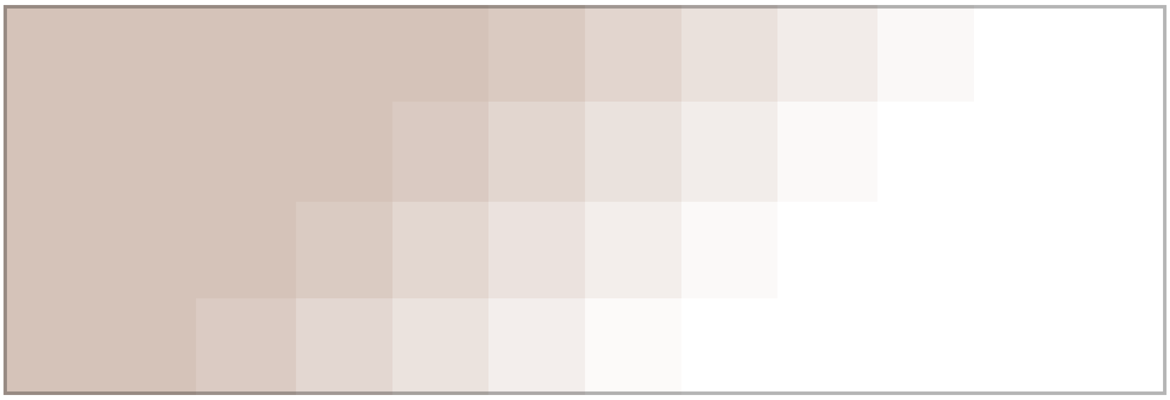
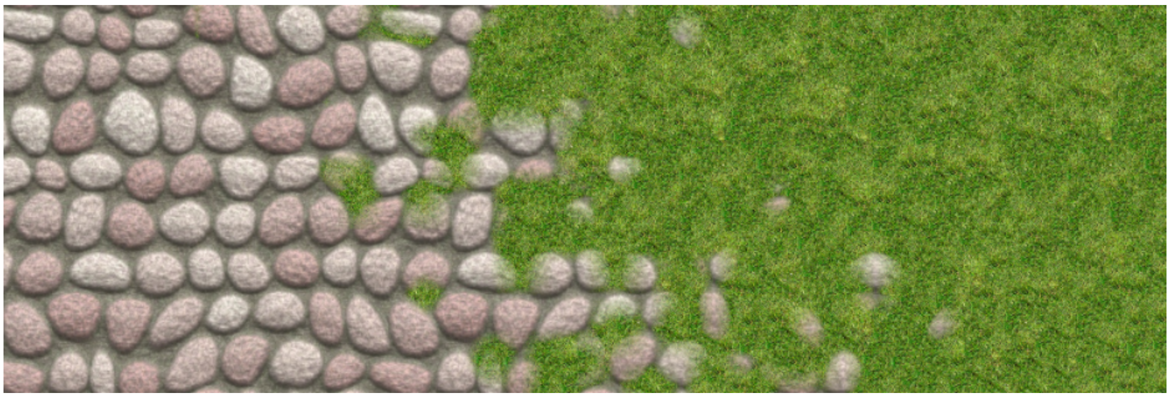


Figure. 4.20 A FBPB blend between a cobble material and grass underlay with a low resolution weight texture (a) using DPT (b) and blend maps (c).


(a)


(b)

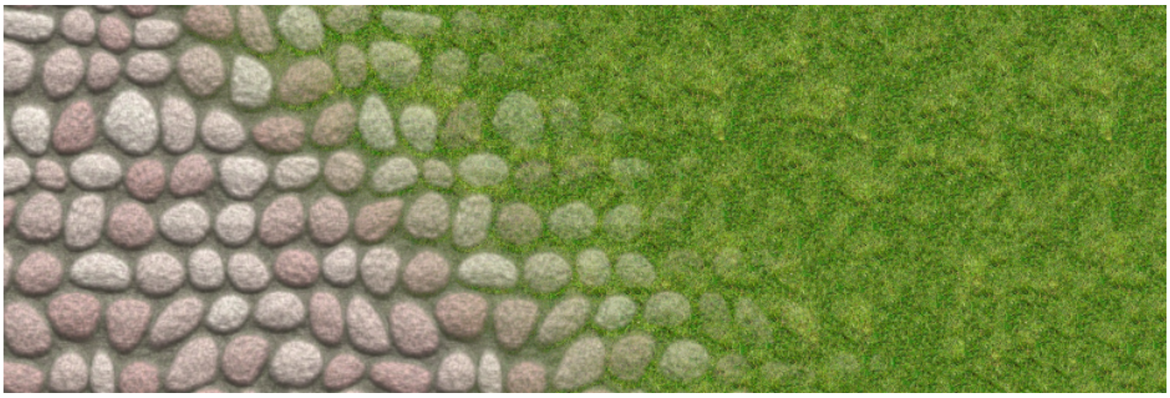

(c)

Figure. 4.21 A blend between a cobble material and grass underlay with a low resolution weight texture (a) using DPT (b) and blend maps (c).

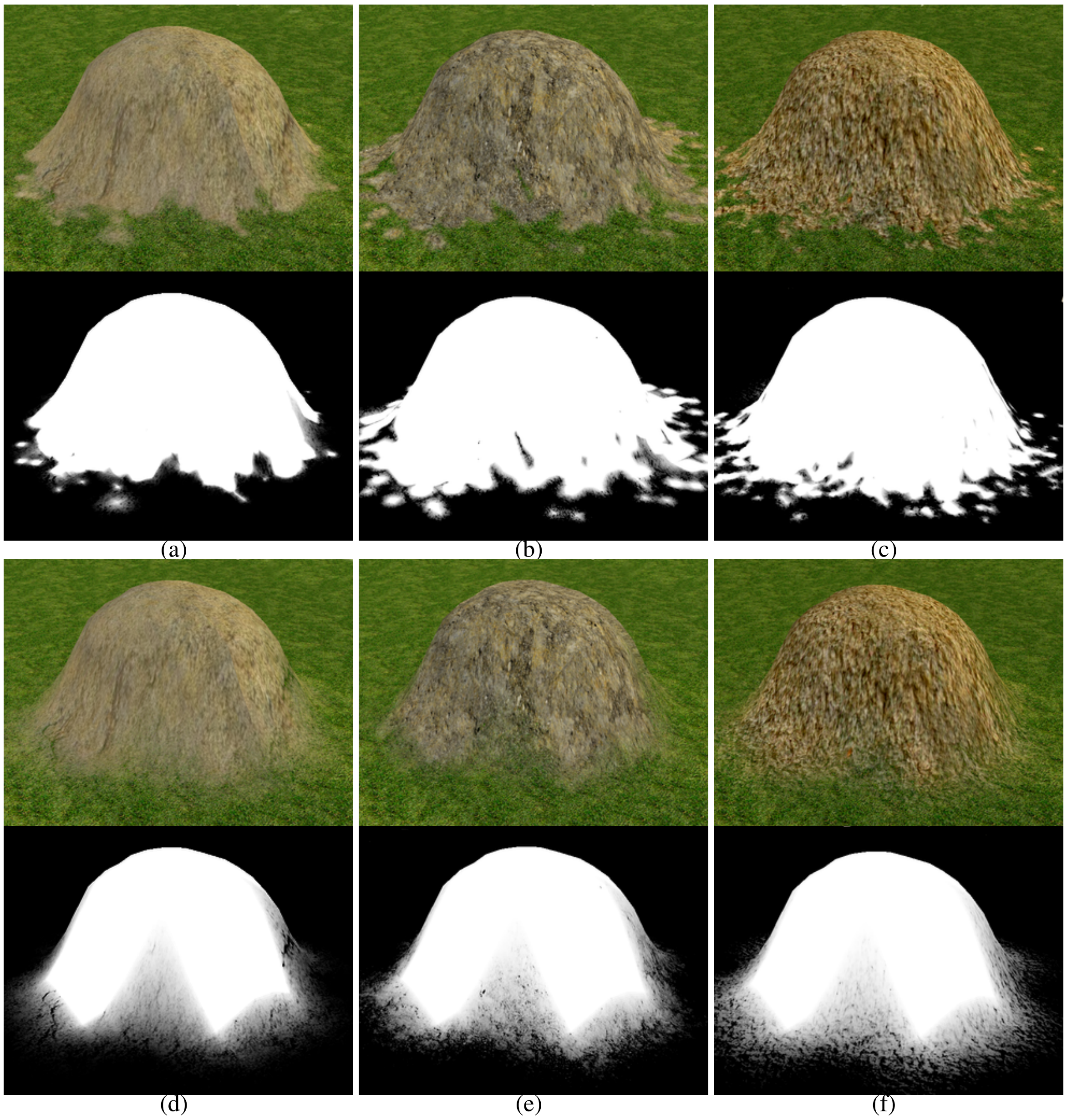


Figure. 4.22 A small hill blended with three different materials over a grass underlay using DPT (a, b & c) and blend maps (d, e & f) along with the generated blend masks (bottom row).

Chapter 5

Conclusions

With the steady advancement of consumer graphics and computing hardware, virtual landscapes are becoming ever richer in detail and scope. The latest generations of gaming consoles and proliferation of cheap, powerful desktop hardware have pushed the boundaries and expectations of terrain rendering towards the ever closer goal of photo-realism. Finally, after decades of research, the approach to terrain texturing has come full circle with the adoption of virtual texturing. Although certainly not the final word in terrain texturing, the circumvention of hardware limits on texture size is giving artists unprecedented control and freedom on how the landscapes are depicted to the user.

In recent years, the adoption of rapid mobile, tablet and other computationally lightweight platforms have sustained the need for lightweight and efficient algorithms for the rendering outdoor environments. Techniques such as texture splatting, previously the staple of terrain texturing for nearly a decade, have continued to find success where hardware resources are limited. Although simple to conceptualize and implement, the technique and its derivatives have matured to offer a wide suite of algorithms for texturing terrains without the need for cutting edge hardware.

5.1 FBPB

Texture splatting and other alpha blending techniques typically exhibit translucency artefacts for salient features where the blending process has no regard for the topology of the material texture. Furthermore, this lack of contextual awareness can also result in contours that are limited in shape and variation to the blend mask texture, which itself is often of low resolution. This can lead to transitions between materials containing such salient features that have unrealistic translucency artefacts and uniform, unvaried contour shapes.

FBPB is a novel approach for eliminating these translucency artefacts for salient features and introducing variation to the transition contour by ensuring that such features are drawn in a probabilistic fashion. By using the blend weight as the chance of such features appearing, the features can be rendered with full opacity or discarded entirely to produce stochastic transitions. Furthermore, the features themselves have their albedo, normal and height modulated by dynamic wear and tear to give the appearance of the endless stochastic detail found in the natural world.

The advantage of FBPB over existing approaches is its awareness of the material topography. This allows it to introduce intermittency and irregularity at the transition and along the contour for materials containing distinct features. The translucency artefacts are eliminated completely and the process is entirely automated in real-time. Compared to tile-based texture mapping, this detail and variation comes at no extra memory cost other than that of the meta-texture itself. The performance of FBPB is constant regardless of the number of features and is capable of scaling up to terrains and materials of any size as the resources required are on a per-material basis rather than being tied to the geometry of the mesh.

5.2 DPT

As texture splatting and other alpha blending techniques operate at the fragment level no accommodation is taken into consideration for the overall contour of the transition. Techniques such as blend maps Hardy and Mc Roberts (2006) add much needed high frequency detail to the resulting transitions but the overall shape of the transition remains the same. This can result in transitions that have little variation or richness to them on areas of the terrain that receive similar blend weight values.

DPT is a novel approach for modulating the contour of the transition by dividing texture space into patches. These patches are then drawn or discarded on a probabilistic fashion in a manner derived from FBPB. By accumulating the weights of multiple different patch sizes and filtering them to produce a smooth result, the contour can be greatly enhanced in both shape and detail. When combined with blend mapping, the result is a transition that is far richer and more detailed than can be achieved with texture mapping or blend mapping alone.

The advantage of DPT over blend mapping is that DPT operates on the contour rather than the material, creating irregular blend contours of near-endless variation in real-time that offer far greater resolution using low resolution blend weights than conventional techniques. This process is entirely automated and offers consistent, competitive performance for no additional video memory overhead. Artists have great freedom to create a wide range of material transitions through patch mask accumulation and parameter modulation. As DPT does not require any pre-processing of assets, any non-salient textures can be used, allowing for instant previewing and appraisal of results.

5.3 Future Works

FPBP was designed to work with materials featuring salient features but this general probabilistic approach can be extended to non-salient materials. This has partially been accomplished with DPT but could be modified to create variation and irregularity within the material itself. Much in the way that the variation textures of FBPB work, materials could be synthesized in real-time by probabilistically combining different surface parameters to create truly stochastic and detailed materials.

In order to achieve this, the terrain material can be broken up into two layers, the feature and non-feature layer. These two layers can then be composited together in the fragment shader to produce the final material texture. The non-feature layer will contain non-salient details and can be tiled across the terrain without any obvious repetition as the contribution of this layer is low frequency detail devoid of any distinct information. The feature layer instead would contain various salient details such as cracks, features, debris and any other relevant information that looks unnatural when tiled across the terrain due to the repetition of such details from the tiling process.

A naive solution would be to break up texture space into grids in the same manner as DPT and render a feature detail per cell of the grid from a list of features (perhaps a texture array) in a probabilistic manner. However, any regularity of possible feature placement could become apparent when large enough repetitions of the material texture are in view so, instead, the placement of each feature could be offset from the cell by a random amount to minimise the possibility of the same feature occupying the same cell position in a repetition of the texture close by.

An alternative to offsetting the position of features by random amounts would be to adopt a process similar to DPT through the use of layering features using different grid sizes and compositing the results. If a high resolution feature list was used as the exemplars then the features could be stretched to fit the cell sizes, allowing for far greater variation in feature size and layout without a noticeable loss in resolution. Additional processing could be performed in the fragment shader to composite multiple features together in more elaborate ways than simply summing the albedo of each feature, such as features from higher precedence layers modulating the normal maps of lower precedence features along the contours of said features.

DPT was designed to work with materials that contain little salient surface information. However, in practice many materials contain intermittent details such as the odd pebble, crack or any other topographical information. Although the translucency artefacts of blending such details with DPT are minimal, preserving these details and extending them beyond the transition contour will add more detail and richness to the transition. This could be achieved by integrating some of the feature-aware principles of FBPB to ensure that such details are rendered more intelligently by the algorithm.

In order to ensure that such intermittent details are not prematurely truncated by the sort of sharp transitions that DPT is capable of producing, said details could either be suppressed at the contour of transitions (ensuring that they do not reach beyond the transition boundary of the material in question) or the transition could be sustained for long enough to ensure that transition always fully encapsulates a given detail. In either case, the idea would be to ensure that the details in question are rendered within the boundary of the material's transition so that this truncation does not occur.

As such details are often indistinct enough from the material to make isolation (as per FBPB) difficult, isolating the non-feature and indistinct feature parts of the material into two separate layers (as described above) and compositing them together in real-time would allow for more flexibility when implementing an algorithm to address these truncation issues. Such details could then be probabilistically blended in the same manner to FBPB. If the extension of such details past the boundary of the transition (for example, a crack or pebble that straddles the transition boundary of a material) looks unnatural, the alpha channel of this indistinct feature layer could contain a feature map for that detail that extends beyond the boundary of said detail so that a portion of the non-feature layer along the periphery of that detail is also extended past the transition boundary.

The two techniques described above for extending FBPB and DPT could potentially be unified under one algorithm, allowing for a single solution to be adopted for materials containing details that are either salient, non-salient or both. This would require a modification to the artistic process as, instead of painting details into one material texture, the algorithm would require separate layers for both salient/quasi-salient and non-salient details. This would make adopting stock textures more difficult as the salient feature isolation can be very difficult for materials with ambiguous details. However, it would offer more variation in results by allowing any combination of different salient and non-salient texture sets to be combined together in real-time.

References

Michael Abrash. Quake's lighting model: Surface caching, 1996. URL `http://www.bluesnews.com/abrash/chap68.shtml`.

Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008. ISBN 987-1-56881-424-7.

AMD. AMD Radeon HD 6970 graphics, 2010. URL `http://www.amd.com/en-us/products/graphics/desktop/6000/6970`.

Johan Andersson. Terrain rendering in frostbite using procedural shader splatting. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 38–58, New York, NY, USA, 2007. ACM.

S. Andersson and J. Goransson. Virtual texturing with WebGL. 2012.

Phillip N. Azariadis and Nikos A. Aspragathos. On using planar developments to perform texture mapping on arbitrarily curved surfaces. *Computers & Graphics*, 24(4):539–554, 2000.

Sean Barret. Sparse virtual textures. In *Game Developers Conference*, 2008.

Robert Berger. *Undecidability of the Domino Problem (Memoirs ; No 1/66)*. Amer Mathematical Society, 1966. ISBN 0821812661.

James F. Blinn. Simulation of wrinkled surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '78, pages 286–292, New York, NY, USA, 1978a. ACM.

James Frederick Blinn. Computer display of curved surfaces. *The University of Utah*, 1978b.

Charles Bloom. Terrain texture compositing by blending in the frame-buffer, November 2000. URL `http://goo.gl/xDALP`.

David Blythe, Brad Grantham, Tom Mcreynolds, and Scott R. Nelson. *Advanced Graphics Programming Techniques Using OpenGL*. Number 29 in Course Notes for SIGGRAPH '99. ACM, August 1999.

K. R. Boff, L. Kaufman, and J. P. Thomas, editors. *Handbook of Perception and Human Performance: Sensory Processes and Perception*, volume 1. John Wiley & Sons, New York, 1986.

Kevin Boulanger. *Real-Time Realistic Rendering of Nature Scenes with Dynamic Lighting*. PhD thesis, University of Rennes I, 2008.

Kévin Boulanger, Sumanta Pattanaik, and Kadi Bouatouch. Rendering grass terrains in real-time with dynamic lighting. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6.

Eric Bruneton and Fabrice Neyret. Precomputed atmospheric scattering. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR'08, pages 1079–1086, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

Salvatore A. Catanese, Emilio Ferrara, Giacomo Fiumara, and Francesco Pagano. Rendering of 3d dynamic virtual environments. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, SIMUTools '11, pages 351–358, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-1-936968-00-8. URL `http://dl.acm.org/citation.cfm?id=2151054.2151116`.

Ying-Chieh Chen and Chun-Fa Chang. A prism-free method for silhouette rendering in inverse displacement mapping. *Computer Graphics Forum*, 27(7):1929–1936, 2008. ISSN 1467-8659.

Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 115–122, New York, NY, USA, 1998. ACM. ISBN 0-89791-999-8.

Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 287–294, New York, NY, USA, 2003. ACM. ISBN 1-58113-709-5.

John Conway. Mathematical games. *Scientific American*, 1970.

Robert L. Cook. Shade trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 223–231, New York, NY, USA, 1984. ACM. ISBN 0-89791-138-5.

Daniel Cornel. Texture virtualization for terrain rendering. Technical report, Vienna University of Technology, 2012.

Crytek. Character budgets, 2011a. URL `http://docs.cryengine.com/display/SDKDOC2/Character+Budgets`.

Crytek. Painting terrain, 2011b. URL `http://freesdk.crydev.net/display/SDKDOC2/Painting+Terrain`.

Karel Culik, II. An aperiodic set of 13 wang tiles. *Discrete Math.*, 160(1-3):245–251, November 1996. ISSN 0012-365X.

Christian Dick, Jens Krüger, and Rüdiger Westermann. GPU ray-casting for scalable terrain rendering. In *Proceedings of Eurographics 2009 - Areas Papers*, pages 43–50, 2009a.

Christian Dick, Jens Schneider, and Rüdiger Westermann. Efficient Geometry Compression for GPU-based Decoding in Realtime Terrain Rendering. *Computer Graphics Forum*, 28: 67–83, M 2009b.

Rouslan Dimitrov. Cascaded shadow maps. Technical report, NVIDIA Corporation, 2007. URL `http://developer.download.nvidia.com/SDK/10/opengl/src/cascaded_shadow_maps/doc/cascaded_shadow_maps.pdf`.

M. Duchaineau, M. Wolinsky, D.E. Sigeti, M.C. Miller, C. Aldrich, and M.B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. In *Visualization '97, Proceedings*, pages 81–88, 1997.

William Dungan, Jr., Anthony Stenger, and George Sutty. Texture tile considerations for raster graphics. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '78, pages 130–134, New York, NY, USA, 1978. ACM.

Epic Games. Unreal technology. URL `http://www.unrealengine.com/`.

Epic Games. Setting up terrain in Unreal, 2009a. URL `http://udn.epicgames.com/Three/SettingUpTerrain.html`.

Epic Games. Terrain advanced textures, 2009b. URL `http://udn.epicgames.com/Three/TerrainAdvancedTextures.html`.

John Ferraris, Christos Gatzidis, and Feng Tian. Automating terrain texturing in real-time using a rule-based approach. *The International Journal of Virtual Worlds*, 9(4), December 2010a.

John Ferraris, Feng Tian, and Christos Gatzidis. Feature-based probability blending. In *ACM SIGGRAPH ASIA 2010 Posters*, SA '10, pages 51:1–51:1, New York, NY, USA, 2010b. ACM. ISBN 978-1-4503-0524-2.

John Ferraris, Feng Tian, and Christos Gatzidis. Feature-based probabilistic texture blending with feature variations for terrains. *Computer Animation and Virtual Worlds*, 23(3-4): 435–445, 2012. ISSN 1546-427X.

John Ferraris, Feng Tian, and Christos Gatzidis. Automatic terrain texturing with dynamic patch transitions. *Computers in Entertainment*, ahead of print, 2015.

Jonathan Ferraris and Christos Gatzidis. A rule-based approach to 3D terrain generation via texture splatting. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE '09, pages 407–408, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-864-3.

Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 213–222, New York, NY, USA, 1984. ACM. ISBN 0-89791-138-5.

Khronos Group. Khronos releases Opengl 4.4 specification. July 2013. URL `https://www.khronos.org/news/press/khronos-releases-opengl-4.4-specification`.

Pascal Guitton, Jean Roman, and Gilles Subrenat. Implementation results and analysis of a parallel progressive radiosity. In *Proceedings of the IEEE symposium on Parallel rendering*, PRS '95, pages 31–38, New York, NY, USA, 1995. ACM. ISBN 0-89791-774-X.

Chris Hall. Virtual textures texture management in silicon. Technical report, 3Dlabs Inc., 1999.

Charles Han, Eric Risser, Ravi Ramamoorthi, and Eitan Grinspun. Multiscale texture synthesis. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 51:1–51:8, New York, NY, USA, 2008. ACM. ISBN 978-1-4503-0112-1.

Alexandre Hardy and Duncan Andrew Keith Mc Roberts. Blend maps: enhanced terrain texturing. In *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, SAICSIT '06, pages 61–70, Republic of South Africa, 2006. South African Institute for Computer Scientists and Information Technologists. ISBN 1-59593-567-3.

Paul S Heckbert. Survey of texture mapping. *IEEE Comput. Graph. Appl.*, 6:56–67, November 1986. ISSN 0272-1716.

Paul S. Heckbert. Fundamentals of texture mapping and image warping. Technical Report UCB/CSD-89-516, EECS Department, University of California, Berkeley, Jun 1989. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/1989/5504.html`.

Alex Herrera. Ending the tradeoff of time vs. quality when creating 3D computer graphics content - stepping up the production workflow with real-time rendering software. *ACM Computer Graphics*, 44(1), February 2010.

Naty Hoffman and Arcot J. Preetham. Graphics programming methods. chapter Real-time Light-atmosphere Interactions for Outdoor Scenes, pages 337–352. Charles River Media, Inc., Rockland, MA, USA, 2003. ISBN 1-58450-299-1.

Charles-Frederik Hollemeersch, Bart Pieters, Peter Lambert, and Rik Van de Walle. Accelerating virtual texturing using CUDA. In Wolfgang Engel, editor, *GPU Pro*, pages 623–642. A K Peters, 2010.

Id Software. Ultimate doom, a. URL `http://www.idsoftware.com/games/doom/doom-ultimate/`.

Id Software. Quake, b. URL `http://www.idsoftware.com/games/quake/quake/`.

Id Software. Wolfensten 3d and spear of destiny, c. URL `http://www.idsoftware.com/games/wolfenstein/wolf3d/`.

David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. *SIGGRAPH Comput. Graph.*, 20(4):133–142, August 1986. ISSN 0097-8930.

Intel. Fast CPU DXT compression. Technical report, Intel, 2012a.

Intel. Pre-compositing textures for terrain rendering, 2012b. URL http://software.intel.com/en-us/articles/pre-compositing-textures-for-terrain-rendering.

Deane B. Judd. Color in business science and industry. *Appl. Spectrosc.*, 7(2):90–91, May 1953.

James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM. ISBN 0-89791-196-2.

Khronos Group. WebGL specification. URL http://www.khronos.org/registry/webgl/specs/latest/.

R. Kooima, J. Leigh, A. Johnson, D. Roberts, M. SubbaRao, and T.A. DeFanti. Planetary-scale terrain composition. *Visualization and Computer Graphics, IEEE Transactions on*, 15(5):719 –733, sept.-oct. 2009. ISSN 1077-2626.

Yueh-Yi Lai and Wen-Kai Tai. Transition texture synthesis. *J. Comput. Sci. Technol.*, 23(2): 280–289, March 2008. ISSN 1000-9000.

Yueh-Yi Lai, Wen-Kai Tai, Chin-Chen Chang, and Chen-Duo Liu. Synthesizing transition textures on succession patterns. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '05, pages 273–276, New York, NY, USA, 2005. ACM. ISBN 1-59593-201-1.

G.W. Larson, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *Visualization and Computer Graphics, IEEE Transactions on*, 3(4):291–306, 1997.

Sylvain Lefebvre and Fabrice Neyret. Pattern based procedural textures. In *Proceedings of the 2003 symposium on interactive 3D graphics*, I3D '03, pages 203–212, New York, NY, USA, 2003. ACM. ISBN 1-58113-645-5.

Sylvain Lefebvre, Jérome Darbon, and Fabrice Neyret. Unified Texture Management for Arbitrary Meshes. Rapport de recherche RR-5210, INRIA, 2004. URL http://hal.inria.fr/inria-00070783.

J. P. Lewis. Algorithms for solid noise synthesis. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '89, pages 263–270, New York, NY, USA, 1989. ACM. ISBN 0-89791-312-4.

Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, July 2001. ISSN 0730-0301.

Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Transactions on Graphics*, 23:769–776, 2004.

Shang Ma, Xiaohui Liang, Zhuo Yu, and Wei Ren. Light space cascaded shadow maps for large scale dynamic environments. In *Proceedings of the 2nd International Workshop on Motion in Games*, MIG '09, pages 243–255, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10346-9.

Benoit Mandelbrot. How long is the coast of britain? statistical self-similarity and fractional dimension. *Science*, 156(3775):636–638, 1967.

Mircea Marghidanu. Fast computation of terrain shadow maps. Technical report, nervus.org, 2002. URL www.nervus.org/files/fctsm.pdf.

Nelson L. Max and Barry G. Becker. Bump shading for volume textures. *IEEE Comput. Graph. Appl.*, 14:18–20, July 1994. ISSN 0272-1716.

Albert Julian Mayer. *Virtual Texturing*. PhD thesis, University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, October 2010.

Morgan McGuire and Max McGuire. Steep parallax mapping. *I3D 2005 Poster*, 2005. URL http://www.cs.brown.edu/research/graphics/games/SteepParallax/index.html.

Microsoft. *Programming Guide for Direct3D 11: Tessellation Overview*. Microsoft, 2012.

Microsoft. DirectX tiled resources. Technical report, Microsoft Corporation, 2013. URL http://msdn.microsoft.com/en-us/library/windows/apps/bg182880.aspx.

Martin Mittring and Crytek GmbH. Advanced virtual texture topics. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pages 23–51, New York, NY, USA, 2008. ACM.

Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, pages 41–50, New York, NY, USA, 1989. ACM. ISBN 0-89791-312-4.

M.E. Newell, R.G. Newell, and T.L. Sancha. A new approach to the shaded picture problem. In *Proc. ACM National Conference,*, pages 443–450. ACM, 1974.

Tomoyuki Nishita, Isao Okamura, and Eihachiro Nakamae. Shading models for point and linear sources. *ACM Trans. Graph.*, 4:124–146, April 1985. ISSN 0730-0301.

Alan Norton, Alyn P. Rockwood, and Philip T. Skolmoski. Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '82, pages 1–8, New York, NY, USA, 1982. ACM. ISBN 0-89791-076-1.

nVidia. GPU gems - chapter 12. tile-based texture mapping, 2004. URL http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter12.html.

nVidia. Geforce GTX 660 specifications. Technical report, 2012. URL http://www.geforce.co.uk/hardware/desktop-gpus/geforce-gtx-660.

nVidia. Corporate timeline, 2013. URL http://www.nvidia.com/page/corporate_timeline.html.

Renato M. Okamoto, Flávio L. de Mello, and Claudio Esperança. Texture management in view dependent application for large 3D terrain visualization. In *Proceedings of the 2008 Spring simulation multiconference*, SpringSim '08, pages 641–647, San Diego, CA, USA, 2008. Society for Computer Simulation International. ISBN 1-56555-319-5. URL `http://dl.acm.org/citation.cfm?id=1400549.1400652`.

OpenGL Specification. ARB_tessellation_shader, 2012. URL `http://www.opengl.org/registry/specs/ARB/tessellation_shader.txt`.

James P. O'Shea, Maneesh Agrawala, and Martin S. Banks. The influence of shape cues on the perception of lighting direction. *Journal of Vision*, 10, December 2010.

Vincent Pegoraro, Mathias Schott, and Philipp Slusallek. A mathematical framework for efficient closed-form single scattering. In *Proceedings of Graphics Interface 2011*, GI '11, pages 151–158, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2011. Canadian Human-Computer Communications Society. ISBN 978-1-4503-0693-5.

Ken Perlin. An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 287–296, New York, NY, USA, 1985. ACM. ISBN 0-89791-166-0.

Ken Perlin. Improving noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 681–682, New York, NY, USA, 2002. ACM. ISBN 1-58113-521-1.

Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18:311–317, June 1975. ISSN 0001-0782.

Phil Plait. How far away is the horizon? 2009. URL `http://blogs.discovermagazine.com/badastronomy/2009/01/15/how-far-away-is-the-horizon/#.Uz55wPldXmt`.

Fábio Policarpo, Manuel M. Oliveira, and João L. D. Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 155–162, New York, NY, USA, 2005. ACM. ISBN 1-59593-013-2.

Ashu Rege. Shader model 3.0. Technical report, nVidia Corporation, 2004.

John Rhoades, Greg Turk, Andrew Bell, Andrei State, Ulrich Neumann, and Amitabh Varshney. Real-time procedural textures. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, I3D '92, pages 95–100, New York, NY, USA, 1992. ACM. ISBN 0-89791-467-8.

Tobias Ritschel, Thorsten Grosch, Jan Kautz, and Hans-Peter Seidel. Interactive global illumination based on coherent surface shadow maps. In *Proceedings of graphics interface 2008*, GI '08, pages 185–192, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society. ISBN 978-1-56881-423-0.

Andrew Rollings and David Morris. *Game Architecture and Design: A New Edition*. New Riders Games, 2003. ISBN 0735713634.

Zhang Rong-hua. Real-time optimization technology and its application in terrain rendering. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, volume 3, pages 1349–1352, 2011.

Mattias Roupé and Mikael Johansson. Visual quality of the ground in 3D models: using color-coded images to blend aerial photos with tiled detail-textures. In *Proceedings of the 6th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '09, pages 73–79, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-428-7.

Daniel L. Schacter, Daniel T. Gilbert, and Daniel M. Wegner. *Psychology*. Worth Publishers, 2010. ISBN 1429237198.

Musawir A. Shah, Jaakko Kontinnen, and Sumanta Pattanaik. Real-time rendering of realistic-looking grass. In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '05, pages 77–82, New York, NY, USA, 2005. ACM. ISBN 1-59593-201-1.

John Snydre and Derek Nowrouzezahrai. Fast soft self-shadowing on dynamic height fields. *Computer Graphics Forum*, 27(4):1275–1283, 2008. ISSN 1467-8659.

Jos Stam. Aperiodic texture mapping. Technical report, European Research Consortium for Informatics and Mathematics (ERCIM), 1997.

A. James Stewart. Fast horizon computation at all points of a terrain with visibility and shading applications. *IEEE Transactions on Visualization and Computer Graphics*, 4(1): 82–93, March 1998.

Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. The clipmap: a virtual mipmap. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 151–158, New York, NY, USA, 1998. ACM. ISBN 0-89791-999-8.

Seth Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and ordering large radiosity computations. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 443–450, New York, NY, USA, 1994. ACM. ISBN 0-89791-667-0.

J.M. van Verth and L.M. Bishop. *Essential Mathematics for Games & Interactive Applications: A Programmer's Guide*. Morgan Kaufmann series in interactive 3D technology. Elsevier, Morgan Kaufmann Publ., 2008. ISBN 9780123742971. URL `http://books.google.co.uk/books?id=zkEY9RIm4WkC`.

J. van Waveren. id tech 5 challenges: From texture virtualization to massive parallelization. In *ACM Annual SIGGRAPH Conference 2009: Beyond Programmable Shading*, SIGGRAPH '09, New York, NY, USA, 2009. ACM.

Hao Wang. Proving theorems by pattern recognition ii. *Bell System Technical Journal*, 40: 1–42, 1961. ISSN 0001-0782.

David R. Warn. Lighting controls for synthetic images. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '83, pages 13–21, New York, NY, USA, 1983. ACM. ISBN 0-89791-109-1.

Li-Yi Wei. Tile-based texture mapping on graphics hardware. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, pages 67–, New York, NY, USA, 2004. ACM. ISBN 1-58113-896-2.

Mattias Widmark. Terrain in battlefield 3: A modern, complete and scalable system. In *Game Developers Conference 2012*, GDC '12, 2012.

Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 364–367, New York, NY, USA, 2004. ACM.

Steve Zelinka and Michael Garland. Jump map-based interactive texture synthesis. *ACM Trans. Graph.*, 23:930–962, October 2004. ISSN 0730-0301.

Fan Zhang, Hanqiu Sun, Leilei Xu, and Lee Kit Lun. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, VRCIA '06, pages 311–318, New York, NY, USA, 2006. ACM. ISBN 1-59593-324-7.

Huijie Zhang, Dantong Ouyang, Heping Lin, and Weizhou Guan. Texture synthesis based on terrain feature recognition. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02*, pages 1158–1161, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3336-0.