Player-Driven Procedural Texturing (sketches_0311)

David (grue) DeBry Henry Goffin Chris Hecker Ocean Quigley Shalin Shodhan Andrew Willmott[*]
Maxis, Electronic Arts

1 Introduction

In the video game Spore we make heavy use of player-driven procedural texturing. Our aim is to amplify the user's creativity by allowing them to control a powerful procedural texturing system. We attempt to strike a balance between giving the player a full painting interface (desirable for highly skilled artists, but tedious for others), and limiting them to simple compositions of preauthored textures. We solve this problem differently for our two model kinds: creatures, and buildings/vehicles.

2 Creature Texturing

Our creatures consist of a main body mesh, with attached detail parts. To texture it, we use our effects system, Swarm, to run particle-driven brushes over the mesh surfaces, and to distribute brushes according to creature features. The brushes paint directly into diffuse, specular, and bump map textures. The features of this system include the ability to run particles over a dynamic mesh with both 2D (tangent space) and 3D control, and fast UV generation for the body mesh, using cubemap partitioning.

Although in initial prototyping all paint scripts used vanilla particle control systems, we later developed a number of custom components to optimize common operations or provide advanced features, such as running paint up and down a creature's spine and limbs, and covering the skin with oriented strokes.

The player controls this system by selecting from sets of base, coat, and detail paint scripts. The scripts are also parameterized by player-selectable colour . Finally, for one-click texturing, we let the player select themes with pre-set combinations of these things.

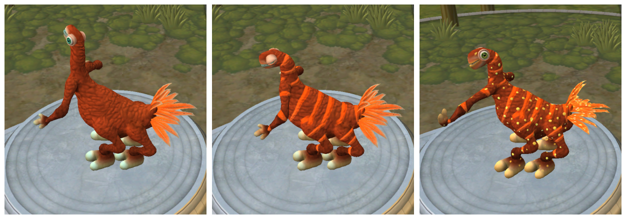
Figure 1: *Texturing a creature.*

3 Building and Vehicle Texturing

Buildings and vehicles are wholly assembled from deformable parts called Rigblocks. The texturing appropriate for these parts is stylistically different from creatures. We prefer tiled swatches of texture, more akin to wallpaper. Because these rigblocks are deformable, often in drastic ways, their uvs cannot be static, or the textures will be stretched and warped unpleasantly. We solve this problem by means of a procedural UVing system. Rigblock regions can be marked as belonging to one of a set of UV types, including planar and cylindrical projections, and most usefully a boxmap projection. These projections are then applied at runtime, in the vertex shader.

Regions are also used to give the player finer-grain control over the look of the model; they are assigned to groups that the player can texture individually. Finally, The textures are parameterized by two palette colours, to allow a large number of combinations.

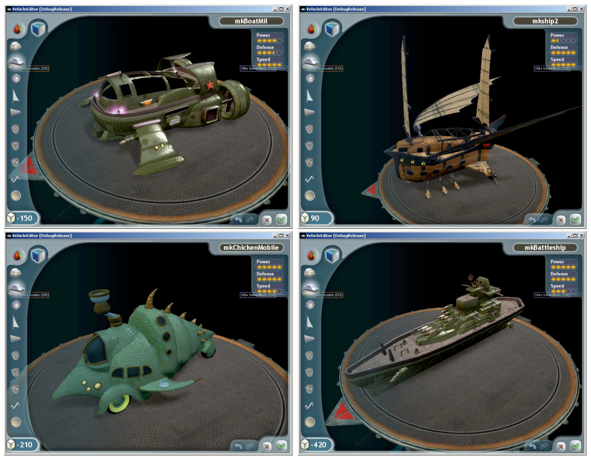

Figure 2: *A number of textured Vehicles.*

4 Final Touches

Once the building or vehicle is ready, a single texture page is assembled by generating a second set of unique UVs, and then using a "splatting" shader to downsample all individually textured parts into the texture page. This allows us both to avoid having draw complexity affected by user material choices, and also to have tight control over final texture resolution. Figure **??** shows the resulting models in game.

We also generate an ambient occlusion map by using a shadow/splat shader to average together visibility samples over the sphere. This provides the visual glue that ties the final model together and makes it look solid.

Both in the editor and in the game we use a spherical harmonics lighting model, interpolating between diffuse and fully glossy terms, to allow a range of diffuse, glossy, and specular finishes. This is particularly important to capture more metallic looks for some parts.

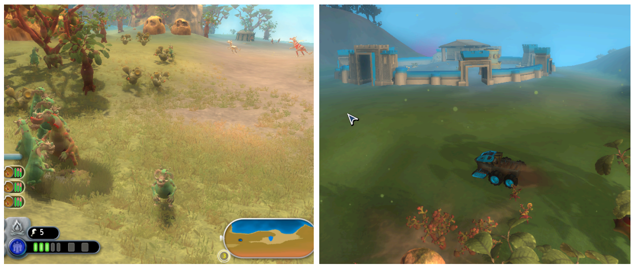
Figure 3: *Player-created models in-game.*

*e-mail: awillmott@maxis.com