Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches

CHARLES LOOP
Microsoft Research
and
SCOTT SCHAEFER
Texas A&M University

We present a simple and computationally efficient algorithm for approximating Catmull-Clark subdivision surfaces using a minimal set of bicubic patches. For each quadrilateral face of the control mesh, we construct a geometry patch and a pair of tangent patches. The geometry patches approximate the shape and silhouette of the Catmull-Clark surface and are smooth everywhere except along patch edges containing an extraordinary vertex where the patches are C^0. To make the patch surface appear smooth, we provide a pair of tangent patches that approximate the tangent fields of the Catmull-Clark surface. These tangent patches are used to construct a continuous normal field (through their cross-product) for shading and displacement mapping. Using this bifurcated representation, we are able to define an accurate proxy for Catmull-Clark surfaces that is efficient to evaluate on next-generation GPU architectures that expose a programmable tessellation unit.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

General Terms: Performance

Additional Key Words and Phrases: Catmull-Clark subdivision, GPU tessellation, subdivision surfaces

ACM Reference Format:

Loop, C. and Schaefer, S. 2008. Approximating Catmull-Clark subdivision surfaces with bicubic patches. ACM Trans. Graph. 27, 1, Article 8 (March 2008), 11 pages. DOI = 10.1145/1330511.1330519 http://doi.acm.org/10.1145/1330511.1330519

1. INTRODUCTION

Catmull-Clark subdivision surfaces [Catmull and Clark 1978] have become a standard modeling primitive in computer generated motion pictures and 3D games. To create a subdivision surface, an artist constructs a coarse polygon mesh that approximates the shape of the desired surface. A subdivision algorithm recursively refines this base mesh to produce a sequence of finer meshes that converge to a smooth limit surface. In practice, developers perform up to 5 off-line subdivision steps to generate a dense mesh suitable for rendering a smooth surface.

This off-line refinement process leads to a number of difficulties when dealing with Catmull-Clark surfaces in real-time applications like games. The large, dense model produced by subdivision consumes limited removable disk space and GPU memory, and requires significant bus bandwidth to be transferred to and from memory. For large numbers of models the amount of resources required can degrade system performance. However, the most serious issue encountered by this off-line refinement strategy is the expense of animation; every vertex of the dense model may need to be modified independently. To minimize the computational overhead needed by these shapes, developers often use a courser mesh, which leads to visible faceting artifacts. All these factors could be mitigated if subdivision was deferred until after base mesh vertex animation in the GPU.

In fact, support for higher order tessellation directly in hardware is becoming a reality as exemplified by both the Microsoft Xbox 360 and the ATI Radeon HD 2900 series graphics cards [Lee 2006; ATI/AMD 2007]. The tessellator unit in these GPUs provide hardware support for adaptive tessellation of parametric surfaces. Based on user-provided tessellation factors, the tessellator adaptively creates a sampling pattern of the underlying parametric domain and automatically generates a set of triangles connecting these samples. The programmer then provides a special shader program that the tessellator calls with the parametric coordinates (u, v) for each sample in the parametric patch; the shader then emits a vertex that corresponds to the patch evaluated at those coordinates. This approach allows the GPU to triangulate arbitrary parametric surfaces because the evaluation details are provided by the programmer in the form of a shader. Furthermore, the GPU is able to exploit parallelism because multiple arithmetic units can be running the same evaluation shader in lockstep. We expect tessellation hardware to become standard in the near future [Blythe 2006; Boyd 2007].

Authors' address: C. Loop, email: cloop@microsoft.com

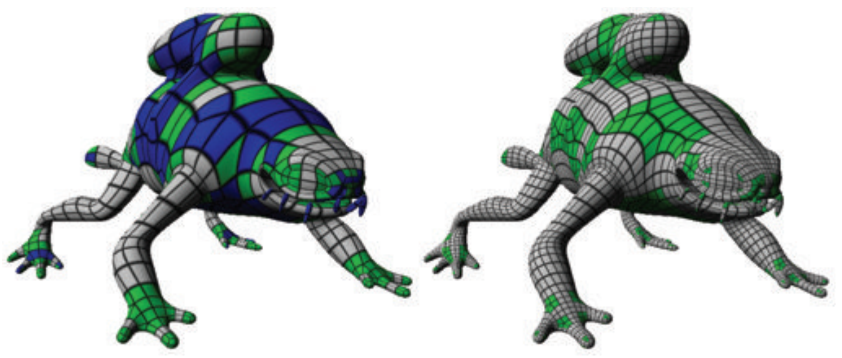

Fig. 1. Blue patches (left) contain more than one extraordinary vertex and cannot be evaluated using Stam's method. The subdivided shape (right) contains patches with one or less extraordinary vertex but increases the number of patches by a factor of 4.

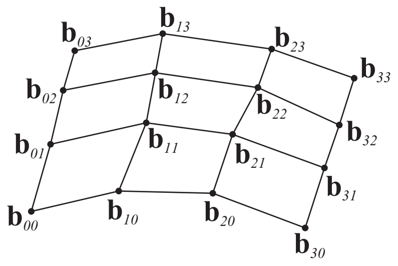

Fig. 2. Control point labeling for a bicubic Bézier patch.

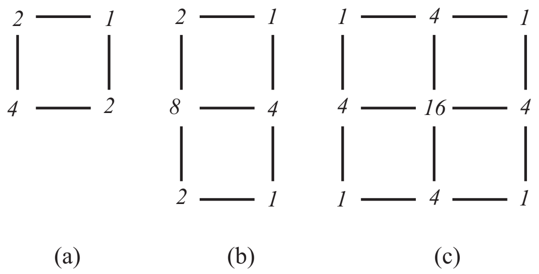

Fig. 3. Masks for determining Bézier control points from a uniform bicubic B-spline surface.

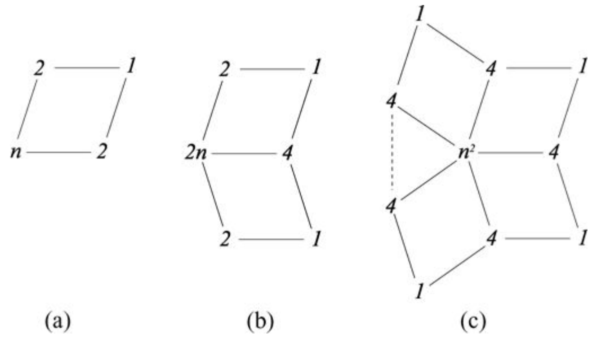

Fig. 4. Generalized masks for interior, edge and corner points.

1.1 Catmull-Clark Surfaces on Tessellation Hardware

Catmull-Clark subdivision surfaces are in fact piecewise parametric and therefore amenable to hardware tessellation. Each quadrilateral face in a Catmull-Clark control mesh corresponds to a single bicubic patch except for quadrilaterals that contain an extraordinary vertex. These *extraordinary patches*, patches containing one or more extraordinary vertices, are actually composed of an infinite collection of bicubic patches. Using this polynomial structure, Stam [1998] developed an algorithm for directly evaluating the parametric form of Catmull-Clark surfaces.

While programmable tessellation hardware will be capable of running Stam's algorithm, there are a number of issues that indicate performance will be poor. Stam's method requires branching, which reduces SIMD efficiency. Stam also factors his computation into a sequence of matrix multiplications. Counting the number of multiplies needed in this evaluation shows that, even in the regular case ($n = 4$), Stam's evaluation will be over an order of magnitude more expensive than normal bicubic evaluation. Finally, Stam's method requires that extraordinary patches contain only one extraordinary vertex. If there are patches that contain more than one extraordinary vertex, one level of subdivision must be performed resulting in 4 times as many patches to evaluate (see Figure 1). If this subdivision step is performed off-line, then even more disk space, memory, computational resources for animation and bandwidth to transfer the data to the GPU are required. Furthermore, this subdivided mesh becomes the coarsest level of resolution, reducing the effectiveness of level of detail management. Therefore, alternatives to exact evaluation are needed to improve performance on a GPU tessellator pipeline.

1.2 Previous Work

Some of the early work in this area used Gregory patches [Chiyokura and Kimura 1983] to create surfaces that interpolate networks of curves and allow the user to specify cross-boundary derivatives. While these patches could be used to approximate Catmull-Clark surfaces, the patches are rational polynomials whose denominators vanish at patch corners complicating evaluation. Furthermore, these patches contain few degrees of freedom that can be used to approximate Catmull-Clark surfaces.

Peters [2000] describes an algorithm that converts Catmull-Clark surfaces into a NURBS approximation of the subdivision surface. This method creates one bicubic patch for each face of a quad mesh. The surfaces produced are C^2 everywhere except near extraordinary vertices where they are C^1. However, this method requires that the

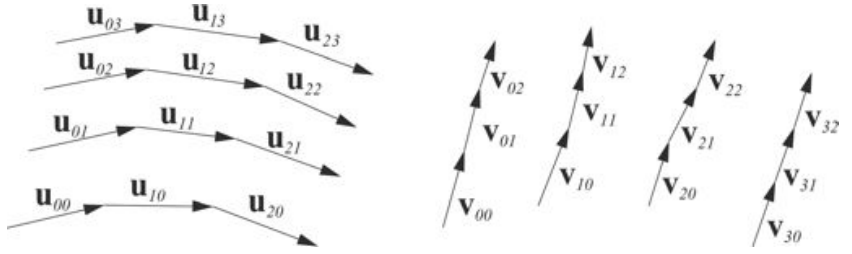

Fig. 5. Control *vectors* for tangent patches ∂u and ∂v.

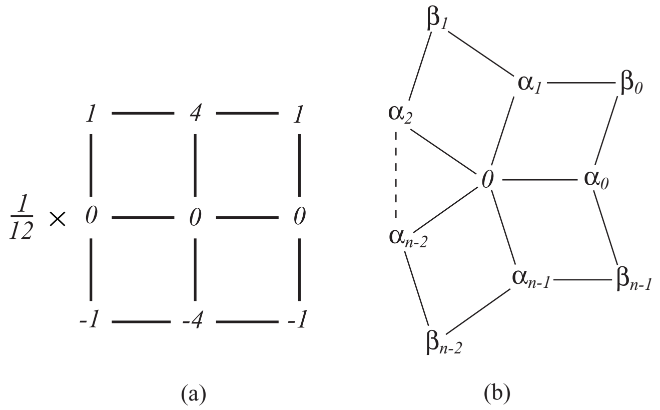

Fig. 6. (a) Tangent mask for uniform bicubic B-splines surfaces (b) Mask for Catmull-Clark limit tangent.

base quad mesh be subdivided at least once (twice if there are extraordinary vertices of even valence) to create sufficient separation of extraordinary vertices resulting in 4–16 times as many patches as the base subdivision surface.

Recently, several researchers have considered techniques for performing subdivision directly on the GPU. Bolz and Schröder [2002], Shiue et al. [2005] and Bunnell [2005] have used the GPU to dynamically tessellate Catmull-Clark surfaces. Bolz and Schröder [2002] and Shiue et al. [2005] require that extraordinary vertices (a vertex not touched by exactly four quadrilaterals) be sufficiently separated, which necessitates at least one level of subdivision in software before GPU acceleration. Bunnell [2005] does not require separation of extraordinary vertices but is a multipass scheme requiring significant CPU intervention. Also, all of these techniques can only produce sampling patterns compatible with binary subdivision, which may introduce visible popping artifacts when patch resolution changes or require blending vertices between different resolutions. None of these methods can produce sampling patterns compatible with future tessellation hardware [Blythe 2006; Lee 2006; ATI/AMD 2007; Boyd 2007].

Finally, Curved PN Triangles (sometimes known as N-Patches) [Vlachos et al. 2001] bear the most similarity to our work. This method takes as input a set of triangles with normals specified at the vertices and attempts to build an interpolating, smooth surface consisting of cubic Bézier triangles. Unfortunately, the patches are not smooth across their edges. To combat this effect, the authors create a separate normal field that gives the surface the appearance of being smooth. The advantage of this method is that the computations are local and a patch can be constructed using only the information present in a single triangle. The disadvantage is that the surfaces suffer from various shading artifacts and the lack of smoothness can typically be seen in the silhouette of the object.

—*Contributions.* We propose an algorithm for visually approximating Catmull-Clark subdivision surfaces, possibly with boundaries, using a collection of bicubic patches (one for each face of a quad-mesh). We contend approximating the surface with patches that are in one-to-one correspondence with the faces of the coarsest base mesh is best. Further subdividing mesh faces may improve the quality of the approximation but diminishes tessellator utilization, requires increased bandwidth to the GPU and limits the minimal level of tessellation leading to over sampling. Our patches are also smooth everywhere except along edges leading to an extraordinary vertex where they are only C^0; therefore shading discontinuities may result. We overcome this difficulty by creating independent tangent patches that conspire to produce a continuous normal field and, hence, the appearance of a smooth surface. When each vertex of the patch has valence 4, our geometry and tangent patches are identical to the Catmull-Clark subdivision surface.

2. GEOMETRY PATCHES

For each face in a quad-mesh, we construct a bicubic patch to approximate the Catmull-Clark surface over the corresponding region. We represent these bicubic patches in Bézier form with the labeling scheme illustrated in Figure 2.

Our geometry patch construction is a generalization of B-spline knot insertion, used to convert from the B-spline to Bézier basis. If all four vertices of a quad-mesh face have valence 4, then the construction reproduces the standard uniform B-spline patch in Bézier form. There are three types of masks needed to construct the control points of a Bézier patch from a uniform B-spline control mesh as shown in Figure 3. These masks encode a set of coefficients that are applied by summing the products of these coefficients and the corresponding points. For masks that generate points (such as the masks in this figure) there is an implied normalization that these masks sum to 1. However, masks that generate vectors must sum to 0 and, thus, do not have an implied normalization.

Referring to Figure 3, mask (a) is used (in four orientations) to create the four *interior points* $\mathbf{b}_{11}$, $\mathbf{b}_{21}$, $\mathbf{b}_{12}$, and $\mathbf{b}_{22}$, corresponding to each quad face; mask (b) is used to create the *edge points* $\mathbf{b}_{10}$, $\mathbf{b}_{20}$, $\mathbf{b}_{01}$, $\mathbf{b}_{02}$, $\mathbf{b}_{31}$, $\mathbf{b}_{32}$, $\mathbf{b}_{13}$, and $\mathbf{b}_{23}$ corresponding to the edges of the quad-mesh; finally, mask (c) is used to create the *corner points* $\mathbf{b}_{00}$, $\mathbf{b}_{30}$, $\mathbf{b}_{03}$, and $\mathbf{b}_{33}$, corresponding to the vertices of the quad-mesh. Note that each edge point lies at the midpoint of two interior points, belonging to adjacent faces; and each corner point lies at the centroid of the 4 interior points that surround that vertex. Our general quad-mesh patch construction is inspired by these geometric relationships.

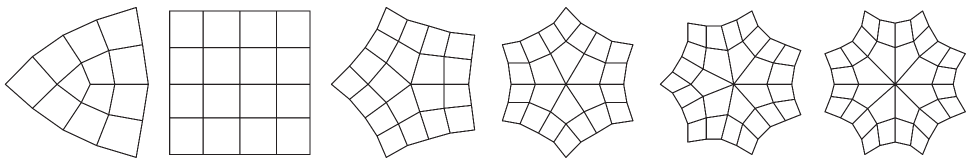
Fig. 7. The characteristic map of a Catmull-Clark surface for various valences.

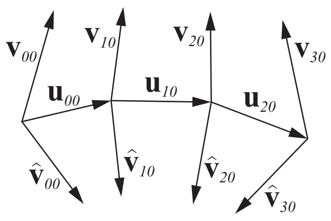

Fig. 8. Control vectors involved in smooth edge conditions.

In the ordinary case (all vertices of the patch have valence 4), the corner points $\mathbf{b}_{00}$, $\mathbf{b}_{30}$, $\mathbf{b}_{03}$, and $\mathbf{b}_{33}$ interpolate the limit position of the Catmull-Clark surface. Therefore, in the extraordinary case, we also choose these control points to interpolate the limit position of the Catmull-Clark surface. Halstead et al. [1993] showed that the left eigenvector corresponding to the dominant eigenvalue of the Catmull-Clark subdivision matrix corresponds to the mask that generates the limit position of an extraordinary vertex. Figure 4(c) illustrates this limit position mask.

If the centroid of the surrounding interior Bézier points creates the corner point with the mask shown in Figure 4(c), then we can infer the mask for the interior points (shown in Figure 4(a). Note that the value n in the generalized interior point mask corresponds to the valence of the vertex whose weight is n. Furthermore, this valence may differ for each interior point $\mathbf{b}_{11}$, $\mathbf{b}_{21}$, $\mathbf{b}_{12}$, and $\mathbf{b}_{22}$. Finally, the edge points are found as midpoints of the adjacent interior points leading to the mask shown in Figure 4(b). Notice that, if $n = 4$, the masks in Figure 4 reproduce the knot insertion masks in the uniform case shown in Figure 3.

3. TANGENT PATCHES

In general, replacing the Catmull-Clark surface with the geometry patches from the previous section results in a surface that is smooth everywhere except along edges containing extraordinary vertices. For some applications, this lack of smoothness may be acceptable. However, for smooth shading, we need a surface that has a continuous normal field over the entire surface. The normal field of a bicubic surface is biquintic, which produces a large number of control vectors in Bézier form to exactly represent the biquintic polynomial (36 control vectors). Furthermore, the control vectors do not depend linearly on the underlying control mesh complicating animation. Therefore, our approach uses a pair of tangent patches denoted by ∂u, ∂v whose cross-product approximates the normal field of the Catmull-Clark surface. These tangent patches have fewer control vectors (they are degree 3×2) and depend linearly on the control mesh.

Consider the tangent patch ∂u. This patch will be bidegree 2×3, since differentiating the bidegree 3×3 geometry patch with respect to u will lower the degree by one in the u-direction. Similarly, the ∂v patch will be bidegree 3×2. Since the ∂u and ∂v patches represent vector fields, their coefficients are control vectors, as illustrated in Figure 5. The construction of tangent patches is symmetric; that is, the constructions are identical up to an interchange of principle directions, with appropriate change of signs. Therefore, we limit our discussion to the ∂v patch.

For Bézier patches, the ∂v patch can be found using differences of the control points. If $\mathbf{b}_{i,j}$ are the coefficients of the geometry patch and $\mathbf{v}_{k,l}$ are the coefficients of the tangent patch, then

$$\mathbf{v}_{i,j} = 3\left(\mathbf{b}_{i,j+1} - \mathbf{b}_{i,j}\right), \quad i = 0, \ldots, 3 \quad j = 0, \ldots, 2. \tag{1}$$

These control vectors represent a Bézier patch that exactly encodes the tangent field in the v-direction of the corresponding geometry patch. However, since the geometry patches do not meet with C^1 continuity everywhere, the tangent patches ∂u and ∂v will not create a continuous normal field. In particular, the tangent field must create a unique normal at the corners of the patch (shared by multiple patches) and along the edges of the patch (shared by two patches). Therefore, we must modify the control points along the edges of ∂v such that we create a continuous normal field over the entire surface.

3.1 Tangent Patch Corners

To modify our tangent patch ∂v, we begin with the corner vertices $\mathbf{v}_{00}$, $\mathbf{v}_{02}$, $\mathbf{v}_{30}$, and $\mathbf{v}_{32}$, which should produce a unique tangent plane among all patches sharing this corner. Unfortunately, our geometry patches are not smooth for an arbitrary valence vertex so our construction from Equation (1) does not produce a unique tangent plane. Given that the geometry patches are meant to approximate the Catmull-Clark surface, we use the limit tangent mask of the Catmull-Clark surface to create a unique tangent plane at the corners of the patch.

Halstead et al. [1993] showed that the tangent limit masks for Catmull-Clark surfaces correspond to the left eigenvectors of the subdivision matrix associated with the subdominant eigenvalue pair. Using these eigenvectors, we arrive at a tangent mask

$$\begin{aligned}\alpha_i^L &= \cos\left(\tfrac{2\pi i}{n}\right), \\ \beta_i^L &= \left(\frac{\sqrt{4+\cos\left(\frac{\pi}{n}\right)^2}-\cos\left(\frac{\pi}{n}\right)}{4}\right)\cos\left(\tfrac{2\pi i+\pi}{n}\right).\end{aligned}$$

where α^L and β^L are the coefficients for the left eigenvector and use the labeling shown in Figure 6(b). Unfortunately, this relationship between the left eigenvectors of the subdivision matrix and the tangent mask only generates a mask that determines the direction of the tangent vector and not its length (eigenvectors are independent of scale). Therefore, we must find an appropriate scale for this vector to ensure a well behaved tangent field.

Our approach conceptually uses the *characteristic map* of the subdivision scheme as a local parameterization of the surface [Reif 1995]. Similar to the tangent mask, the coordinates of the characteristic map are given by the pair of right eigenvectors corresponding to the subdominant eigenvalues. If we allow α^R, β^R to be points

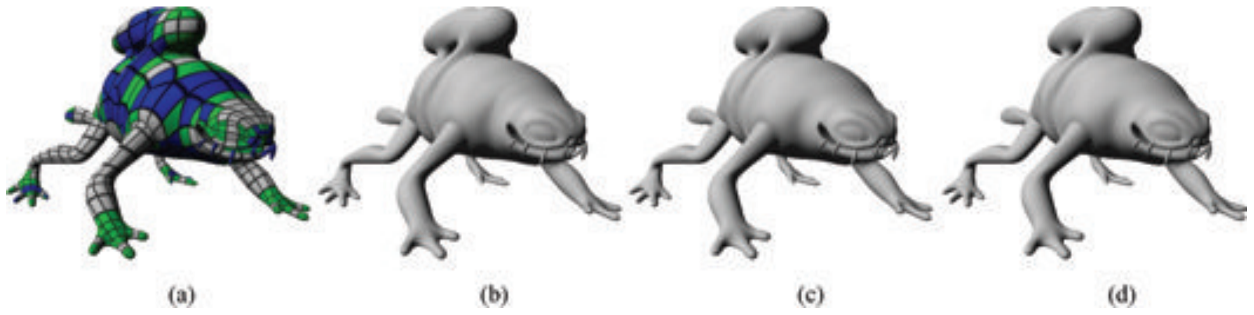

Fig. 9. a) The patch structure we associate with a Catmull-Clark subdivision surface. The grey patches contain only valence 4 vertices, green have one extraordinary vertex and blue have more than one extraordinary vertex. b) Our approximation to the Catmull-Clark subdivision surface using geometry patches and c) our final approximation using geometry and tangent patches compared with d) the actual Catmull-Clark limit surface.

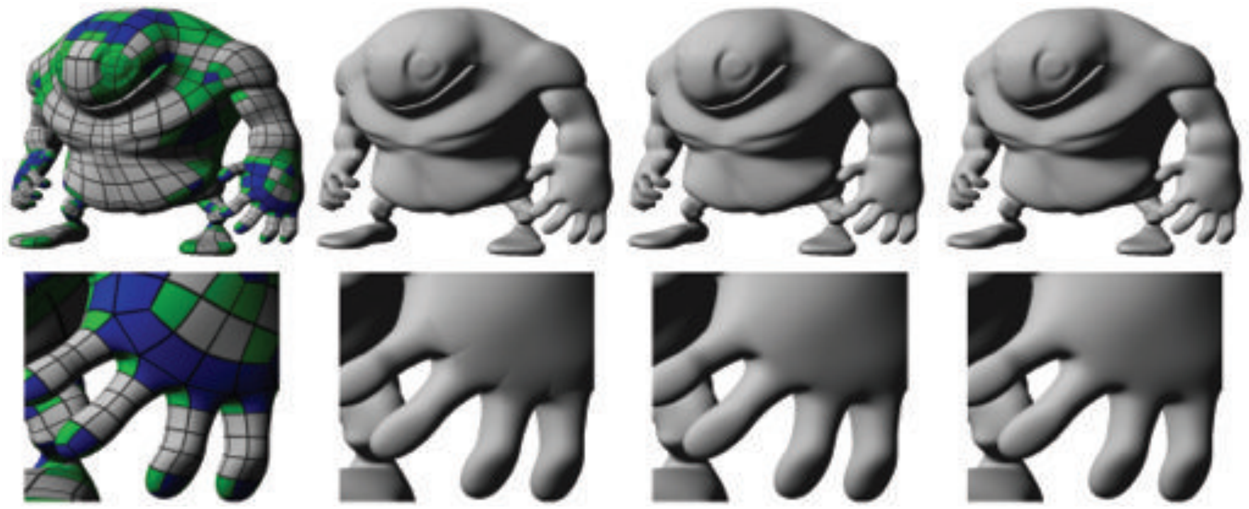

Fig. 10. An example mesh (top) and a zoomed in region of a complex patch structure (bottom). From left to right: Catmull-Clark patch structure, Geometry patch approximation, Geometry/Tangent patch approximation and Catmull-Clark limit mesh.

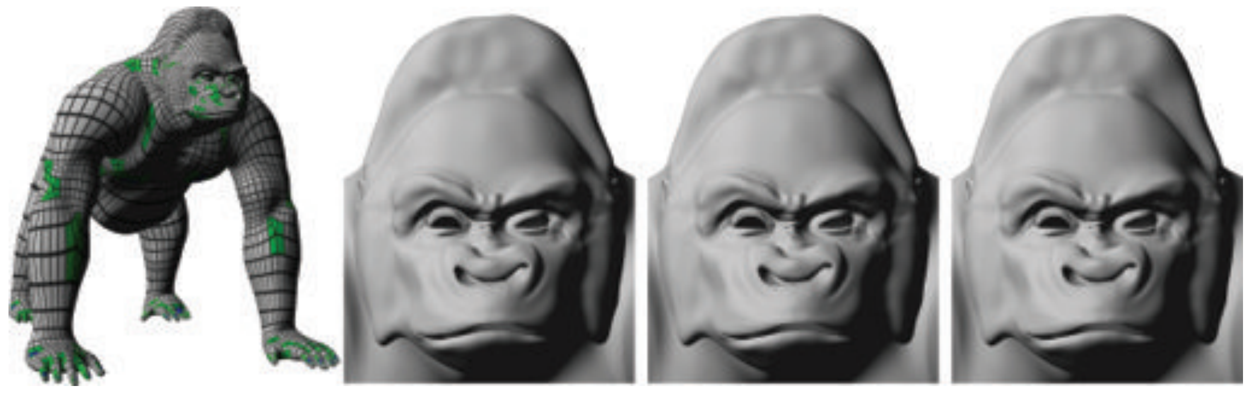

Fig. 11. A complex mesh with boundary. From left to right: Catmull-Clark patch structure, Geometry patch approximation, Geometry/Tangent patch approximation and Catmull-Clark limit surface.

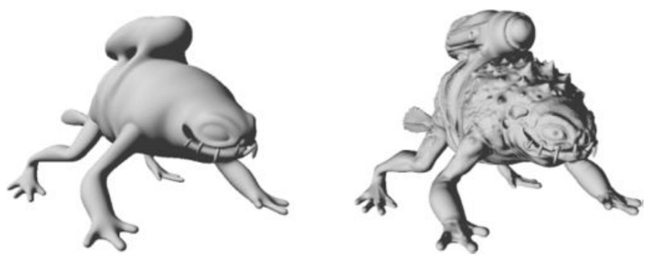

Fig. 12. Our Geometry/Tangent patch model (left) with displacement mapping (right).

in the plane, then the one-ring control points of the characteristic map are

$$\alpha_i^R = \left(\cos\left(\tfrac{2\pi i}{n}\right), \sin\left(\tfrac{2\pi i}{n}\right)\right)$$

$$\beta_i^R = \frac{4}{\sqrt{4+\cos\left(\frac{\pi}{n}\right)^2+\cos\left(\frac{\pi}{n}\right)}}\left(\cos\left(\tfrac{2\pi i+\pi}{n}\right), \sin\left(\tfrac{2\pi i+\pi}{n}\right)\right).$$

The characteristic map is also independent of scale, which we are free to choose. We pick a scale for the map such that $\alpha_0^R = (1, 0)$. Figure 7 shows examples of the characteristic map for Catmull-Clark surfaces for various valences.

If we apply the limit tangent mask to the control points of the characteristic map, the result will be a vector with nonunit length. We then find a scalar σ such that

$$\sigma \sum_{i=0}^{n-1} \alpha_i^L \alpha_i^R + \beta_i^L \beta_i^R = (1, 0).$$

Solving for σ yields

$$\sigma = \frac{1}{n} + \frac{\cos\left(\frac{\pi}{n}\right)}{n\sqrt{4+\left(\cos\left(\frac{\pi}{n}\right)\right)^2}}.$$

Multiplying the previous tangent mask by σ produces the final tangent mask.

$$\alpha_i = \left(\frac{1}{n} + \frac{\cos\left(\frac{\pi}{n}\right)}{n\sqrt{4+\left(\cos\left(\frac{\pi}{n}\right)\right)^2}}\right)\cos\left(\tfrac{2\pi i}{n}\right),$$
$$\beta_i = \left(\frac{1}{n\sqrt{4+\left(\cos\left(\frac{\pi}{n}\right)\right)^2}}\right)\cos\left(\tfrac{2\pi i+\pi}{n}\right). \quad (2)$$

This tangent mask is used to construct the vectors $\mathbf{v}_{00}$, $\mathbf{v}_{02}$, $\mathbf{v}_{30}$, and $\mathbf{v}_{32}$ resulting in a unique tangent plane at each of the patch corners. Also, note that all these vectors must be consistently aligned. In particular, the tangent vector directions must be reversed (multiplied by -1) for $\mathbf{v}_{02}$ and $\mathbf{v}_{32}$. The construction of tangent field vectors $\mathbf{u}_{00}, \mathbf{u}_{20}, \mathbf{u}_{03}$, and $\mathbf{u}_{23}$ is identical. Finally, notice that if $n = 4$, this tangent mask exactly reproduces the tangent mask for bicubic B-splines in Figure 6(a) including scale.

3.2 Tangent Patch Edges

Given the tangent patch from Equation (1) with corner vertices specified by Equation (2), the tangent patches create a unique tangent plane everywhere except along the edges of a patch. Therefore, we must modify the control vectors along the edges of the patch as well.

Consider the patch edge in Figure 8 shared by two patches. $v_{i,j}$ are the control vectors for the top patch along the shared patch boundary in the v-direction, $u_{i,j}$ are the control vectors for the tangent in the u-direction shared by both patches and $\hat{v}_{i,j}$ are the control vectors of the tangent in the v-direction for the bottom patch along the shared edge. These control vectors define two cubic functions $v(t)$, $\hat{v}(t)$ and one quadratic function $u(t)$. These three vector fields will be linearly dependent if

$$((1-t)c_0 - tc_1)\,\mathbf{u}(t) = \frac{1}{2}(\mathbf{v}(t) + \hat{\mathbf{v}}(t)) \quad \forall t \in [0, 1],$$

where $c_i = \cos(\frac{2\pi}{n_i})$ and n_0, n_1 are the valence of the left and right endpoints. Solving for the Bézier coefficients results in four conditions:

$$c_0\,\mathbf{u}_{00} = \tfrac{1}{2}(\mathbf{v}_{00} + \hat{\mathbf{v}}_{00}), \quad (3)$$
$$\tfrac{1}{3}(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}) = \tfrac{1}{2}(\mathbf{v}_{10} + \hat{\mathbf{v}}_{10}), \quad (4)$$
$$\tfrac{1}{3}(c_0\mathbf{u}_{20} - 2\,c_1\mathbf{u}_{10}) = \tfrac{1}{2}(\mathbf{v}_{20} + \hat{\mathbf{v}}_{20}), \quad (5)$$
$$-c_1\,\mathbf{u}_{20} = \tfrac{1}{2}(\mathbf{v}_{30} + \hat{\mathbf{v}}_{30}). \quad (6)$$

Conditions (3) and (6) are satisfied by construction using Equation (2). Condition (4) will be satisfied if

$$\hat{\mathbf{v}}_{10} = \tfrac{1}{3}(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}) + \mathbf{x},$$
$$\hat{\mathbf{v}}_{10} = \tfrac{1}{3}(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}) - \mathbf{x}$$

for any choice of $\mathbf{x}$. We choose $\mathbf{x} = 3\,(\mathbf{b}_{11} - \mathbf{b}_{10})$ since, by construction, we get the same vector $\mathbf{x}$ (up to sign) when processing either patch sharing an edge. Furthermore, this construction reproduces the regular case ($n = 4$). The control vector $\mathbf{v}_{20}$ can be found in a similar fashion. To summarize, we set

$$\mathbf{v}_{10} = \tfrac{1}{3}(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}) + 3\,(\mathbf{b}_{11} - \mathbf{b}_{10}),$$
$$\mathbf{v}_{20} = \tfrac{1}{3}(c_0\mathbf{u}_{20} - 2\,c_1\mathbf{u}_{10}) + 3\,(\mathbf{b}_{21} - \mathbf{b}_{20}).$$

The construction for $\hat{\mathbf{v}}_{10}$, and $\hat{\mathbf{v}}_{20}$ follows in a similar manner.

4. RESULTS

Over the ordinary patches in the mesh (no extraordinary vertices), our construction for the geometry and tangent patches exactly reproduces the surface and tangent field of the Catmull-Clark surface. Therefore, the only regions that our surfaces differ from the actual Catmull-Clark surface are those patches containing one or more extraordinary vertices. Furthermore, our method interpolates the limit position and normal of the Catmull-Clark surface at each vertex of the mesh.

Figures 9, 10, and 11, show examples of subdivision surfaces containing patches with one or more extraordinary vertices. The approximation using only Geometry patches matches the Catmull-Clark surface very well, but is noticeably not smooth along some of the edges. Adding the tangent field to the model creates a smooth normal field and results in a surface that is nearly identical visually to the original Catmull-Clark surface. Because the tangent patches create a continuous normal field over the surface, we can use these shapes for displacement mapping as well. Figure 12 shows an example of displacement mapping applied to our geometry/tangent patch approximation from Figure 9.

Figure 13 illustrates an example of a mesh with a boundary containing both ordinary and extraordinary vertices (our boundary construction can be found in Appendix A). The Catmull-Clark surface uses the boundary rules of Biermann et al. [2000] to produce a smooth subdivision surface.

Despite the fact that our geometry patch approximation is only C^0, the lack of smoothness is rarely if ever visible in the silhouette

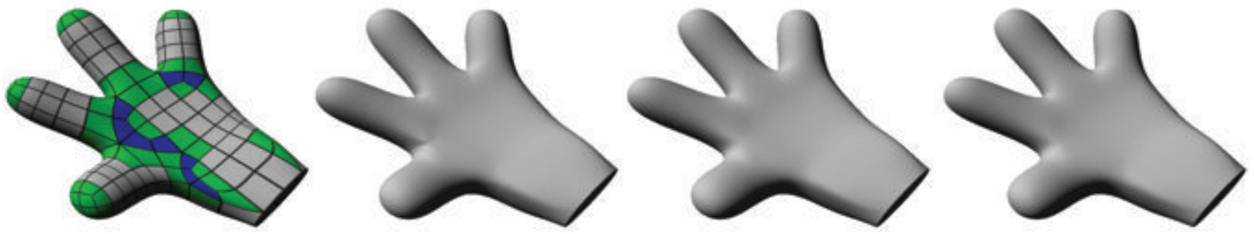
Fig. 13. An example of a mesh with a boundary. From left to right: Catmull-Clark patch structure, Geometry patch approximation, Geometry/Tangent patch approximation and Catmull-Clark limit surface using Biermann et al.'s boundary rules.

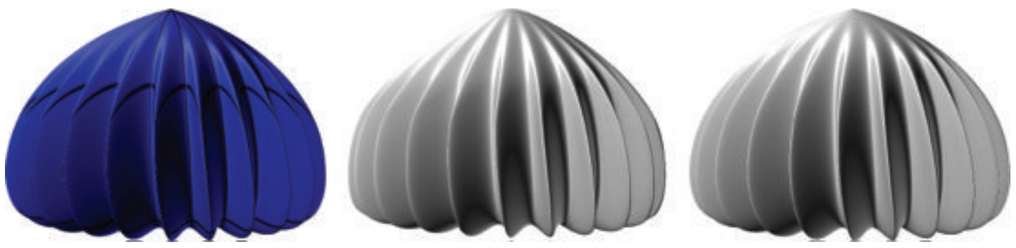
Fig. 14. A model with a valence 24 vertex at the tip composed entirely of patches with more than one extraordinary vertex (left). Our Geometry/Tangent patch approximation (center) actually appears to have a smoother profile at the valence 24 vertex than the Catmull-Clark surface (right).

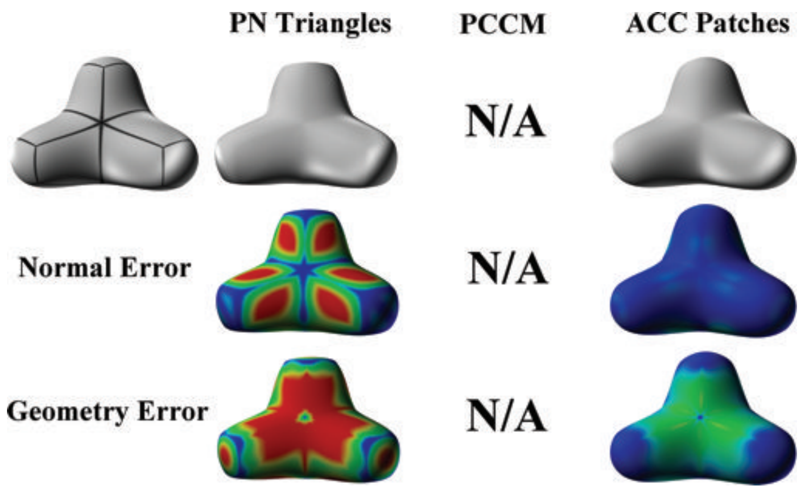

Fig. 15. Comparison of our method with PN-Triangles and PCCM. The figures show the difference between the normal of the Catmull-Clark surface as well as the geometric error. There is not sufficient separation of extraordinary vertices for PCCM in this example.

of the model (we have not been able to discern the C^0 regions of the models along the silhouette in any of our examples). Figure 14 depicts an extreme case that typically is not found in practice. This model consists of a very high valence vertex (24 at the top vertex) and patches that all contain more than one extraordinary vertex. Interestingly, our geometry patch approximation produces a surface that looks smoother than the Catmull-Clark surface at the valence 24 vertex despite the lack of continuity (Catmull-Clark surfaces are known to have unbounded curvature at high valence vertices).

Figures 15–18 show a comparison of our technique with PN-Triangles [Vlachos et al. 2001] and Patching Catmull-Clark Meshes (PCCM) [Peters 2000]. In these pictures, color denotes the error between the given surface and the actual Catmull-Clark surface, where blue is no error and red represents high error. We measure the geometric error as the difference between the given surface and the actual Catmull-Clark surface as a percentage of the length of the bounding box diagonal. The normal error represents the angle between the normals of the two surfaces. In all cases, the correspondence between the two shapes is given by parametric correspondence.

For surfaces composed of triangles, PN-Triangles [Vlachos et al. 2001] can create the illusion of a smooth surface in a similar manner to our Approximate Catmull-Clark Patches for Catmull-Clark surface. We modify PN-Triangles for Catmull-Clark surfaces by

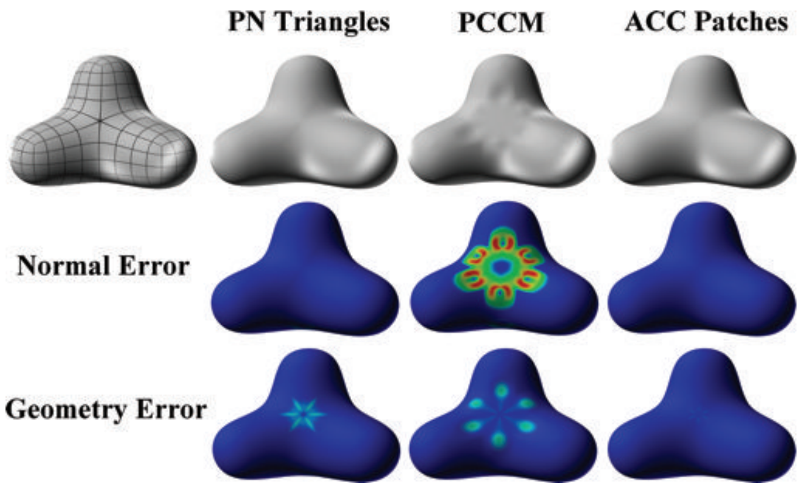

Fig. 16. Comparison of our method with PN-Triangles and PCCM with the surface from Figure 15 subdivided twice. PCCM creates a flat spot with undulations in the surface away from the extraordinary vertex in order to make the surface smooth. ACC Patches produces a surface nearly indistinguishable from the true subdivision surface.

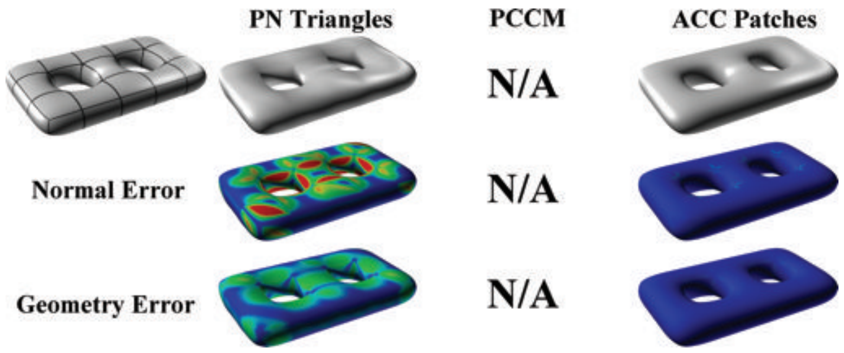

Fig. 17. Comparison of our method with PN-Triangles and PCCM on a two-hole torus containing only odd valence extraordinary vertices. There is not sufficient separation of extraordinary vertices for PCCM in this example.

triangulating the surface and forcing the PN-triangles to interpolate the limit position and normal of the Catmull-Clark surfaces at the vertices. PCCM [Peters 2000] is also similar to our method except that PCCM creates an actual C^1 surface using a finite collection of bicubic patches as opposed to our C^0 surface. However, the extraordinary vertices must be sufficiently separated for PCCM, which requires two-ring separation for odd valence vertices and four-ring separation for even valence vertices.

Figures 15–18 illustrate that, as expected, PN-Triangles does not produce good approximations of the Catmull-Clark surface. This is especially true for the normal field approximation since PN-Triangles only have a quadratic normal field whereas ordinary bicubic patches have a degree 5×5 normal field. Hence, even in ordinary regions of the surface, PN-Triangles do not provide good approximations to the underlying subdivision surface.

Many of our examples (Figures 9, 10, 11, 13, 15, and 17) do not have sufficient separation of extraordinary vertices for PCCM and require one or more levels of subdivision before we can apply this technique. In Figures 15 and 17 this lack of separation is denoted by "N/A." Figures 16 and 18 show the same surfaces subdivided to provide sufficient separation of extraordinary vertices. Surprisingly, ACC patches provide a better approximation to the Catmull-Clark surface both in terms of geometric error as well as normal field error. The reason behind this phenomenon is that PCCM patches are quite constrained by the need to create a C^1 surface using only bicubic patches and can create flat spots or undulations (at even valence vertices) not present in the Catmull-Clark surface. In contrast, ACC patches decouple the geometry and normal field approximation and, therefore, have more degrees of freedom leading to lower error approximations.

5. IMPLEMENTATION

The GPU processing stages needed to support hardware tessellation are illustrated in Figure 19. We should emphasize that no GPU's currently exist that implement all of these stages in hardware and, hence, this section reflects our view of the proposed pipeline necessary for hardware tessellation. Processing begins at the left with the base control mesh as input and proceeds to the right, ending with triangle rasterization. The function of these stages are as follows:

—*Vertex Processing*. In addition to vertex transform, vertices are also animated in this stage.

—*Patch Assembly*. Patch control nets are formed by averaging over local collections of control mesh vertices.

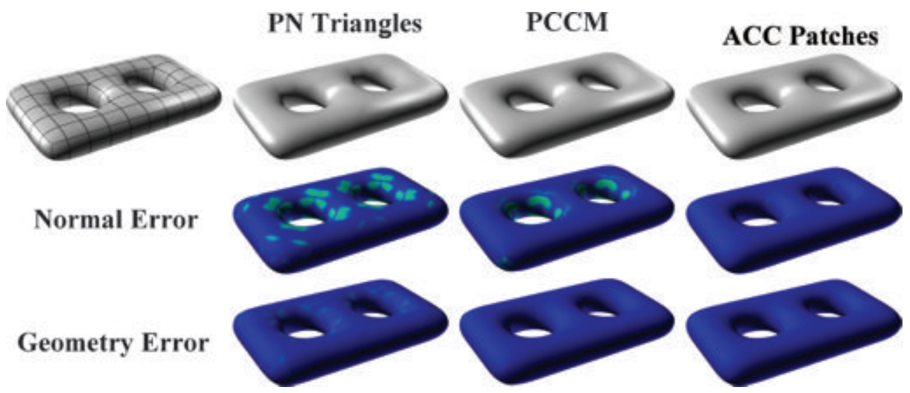

Fig. 18. Comparison of our method with PN-Triangles and PCCM on a two-hole torus containing only odd valence extraordinary vertices after one level of subdivision. Our ACC-Patches provide superior approximation both in terms of the normals of the surface as well as its geometry.

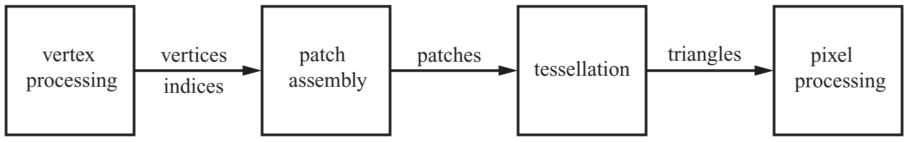

Fig. 19. Proposed future GPU pipeline.

—*Tessellation*. Patches are evaluated at hardware generate domain points.

—*Pixel Processing*. Conventional pixel/fragment shading.

Both the Xbox 360 and the Radeon HD 2900 have vertex and pixel units, as well as tessellation hardware [Lee 2006; ATI/AMD 2007]. Currently, the patch assembly stage on these devices must be implemented as vertex program. In future hardware, we expect patch assembly to be a dedicated hardware stage [Boyd 2007].

The algorithm for patch construction presented in this paper is intended to run as a patch assembly program. Without this specialized hardware stage, our algorithm could be implemented instead as vertex program. In either case, a control mesh would consist of a pair buffers; one that contains control mesh vertices, and a second containing a neighborhood of indices corresponding to the quad faces of the control mesh. The vertices in the first buffer are transformed and animated. Each entry of the second buffer contains the indices of the union of all mesh vertices belonging to faces incident on a quad, together with the valence of each of the four vertices of the quad. These local index neighborhoods are computed as a preprocess by visiting each base mash face. In order to optimize SIMD efficiency and avoid underutilizing GPU buffers, the amount of data that can be packed into an index neighborhood structure must be limited. Therefore, this approach cannot accommodate arbitrarily high valence vertices. The structure for storing these perquad index neighborhoods should only be large enough to handle common cases; larger structures will waste resources in all but extreme situations. When each index neighborhood structure is processed, the shader gathers the needed vertices and averages these to form patch vertices that are written to an output buffer for input to the tessellator stage.

Even though the patches input to the tessellator unit contain 40 control points (16 for geometry and 12 for each tangent), we do not need to transfer this amount of data since many of these control points are redundant. In reality, all we need is the quad of the corresponding patch with its edge-adjacent quads (12 points) as well as the limit positions at the corners of the patch (4 points) and the tangents (8 vectors). Along with the valence at each patch corner we only need 25 points to evaluate the patch on the GPU.

6. FUTURE WORK

While Catmull-Clark surfaces are typically created from quad-meshes, the subdivision rules are general enough to handle meshes with arbitrary sided polygons. Arbitrary polygons are theoretically possible in our framework, but are not practical for adaptive tessellation on current graphics hardware, which support only triangular and quadrilateral domains. However, it is possible to incorporate triangle patches into the tessellation process.

For simplicity, we only operate on meshes consisting entirely of quads. We can, of course, produce an all-quad mesh by performing one step of subdivision. However, the disadvantage of this approach is the increase in the number of patches similar to Figure 1. In the future we would like to extend our method to triangular patches, and more generally triangle-quad surfaces [Stam and Loop 2003] using Bézier triangles.

Finally, Catmull-Clark surfaces are smooth everywhere while, in practice, many surfaces contain sharp edges or corners. We can handle these creases by marking edges in the mesh and treating them as boundary edges. However, DeRose et al. [1998] introduced rules to create semi-sharp creases in Catmull-Clark surfaces. We believe that we may be able to incorporate semi-sharp creases into our method by modifying the patch coefficients as well.

APPENDIX A. MESHES WITH BOUNDARIES

Orginally, Catmull-Clark surfaces were assumed to be closed surfaces; however, not all meshes are closed. Nasri [1987] extended Catmull-Clark subdivision to surfaces with boundaries. Along the boundary, Nasri chose the subdivision rules to reproduce cubic B-splines. To generalize our geometry patches to boundaries, we

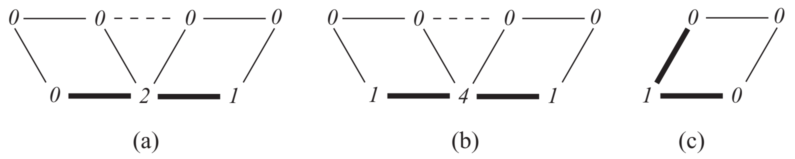

Fig. 20. Rules for Bézier control points along the boundary to create a cubic B-spline. Bolded edges indicate boundaries. From left to right: mask for an edge control point, corner control point and a corner control point contained by only one quad.

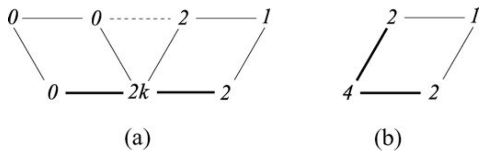

Fig. 21. Rules for creating the interior Bézier control point adjacent to a boundary vertex contained by more than one quad (a) and only one quad (b).

follow Nasri [1987] and require that the boundary curves form cubic B-splines.

A.1 Geometry Patches

For a boundary edge, the two edge points are found along the edge at ratios 1 : 2 and 2 : 1 from the endpoints. For a boundary vertex incident on two or more faces, we find the corner point as the midpoint of the two adjacent edge points. For a boundary vertex contained by only one face, we set the corner point to the boundary vertex. These rules are summarized in Figure 20 with the implied normalization that the masks sum to 1.

Besides modifying the Bézier control points along the boundary, we also modify the interior control point adjacent to a boundary vertex. For boundary vertices contained by more than one quad, we use the mask in Figure 21(a) where k is equal to the number of quads containing the boundary vertex. When a boundary vertex is contained by only one quad, the interior control point is given by Figure 21(b). Similar to Section 2, edge points on interior edges are placed at the midpoint of the adjacent interior points. In the interest of simplicity, we ignore topologically anomalous configurations, such as bow-ties and pin-wheels; though in principle, such configuration do not cause problems.

We make a distinction from the valence n used in Sections 2 and 3 and the number of quads k containing a boundary vertex because we treat the boundary as being half of a closed mesh. Therefore, $n = 2k$. In fact, if we apply this identity to Figure 21, we obtain the interior point mask for closed meshes shown in Figure 4.

A.2 Tangent Patches

Our tangent patch construction is identical to Section 3 except that we modify the way the limit tangents are computed at boundary vertices. For boundary vertices contained by more than one quad, we use the tangent masks derived for Catmull-Clark surfaces by Biermann et al. [2000] to compute the tangent vectors. These tangent vectors provide direction, but lack length information and we use the same normalization process from Section 3 to choose an appropriate length. The result is two tangent masks that create two vectors r_0 and r_1 that span the tangent plane at that vertex. Referring to Figure 22,

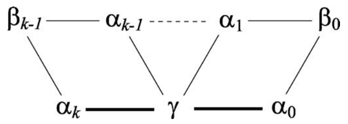

Fig. 22. Labeling for vertices around a boundary vertex contained by k quads.

the mask for r_0 is

$$\begin{aligned}\alpha_0 &= \tfrac{1}{2}\\ \alpha_k &= -\tfrac{1}{2}\\ \alpha_{i\neq 0,k} = \gamma = \beta_i &= 0,\end{aligned}$$

and for r_1

$$\begin{aligned}\gamma &= \tfrac{-4s}{3k+c}\\ \alpha_0 = \alpha_k &= -\tfrac{(1+2c)\sqrt{1+c}}{(3k+c)\sqrt{1-c}}\\ \alpha_{i\neq 0,k} &= \tfrac{4s_i}{3k+c}\\ \beta_i &= \tfrac{s_i+s_{i+1}}{3k+c}\end{aligned}$$

where $c = \cos\left(\frac{\pi}{k}\right)$, $s = \sin\left(\frac{\pi}{k}\right)$ and $s_i = \sin\left(\frac{\pi i}{k}\right)$.

The tangent vector along the j^{th} edge is then given by

$$\cos\left(\frac{\pi j}{k}\right) r_0 + \sin\left(\frac{\pi j}{k}\right) r_1.$$

The rest of the tangent patch construction is the same except that we use the substitution $n = 2k$ in Section 3.2 for the edges of the tangent patches.

For boundary vertices contained by only one quad ($k = 1$), we again change the tangent masks for r_0

$$\begin{aligned}\gamma &= -1\\ \alpha_0 &= 1\\ \alpha_{i\neq 0} = \beta_i &= 0\end{aligned}$$

and for r_1

$$\begin{aligned}\gamma &= -1\\ \alpha_1 &= 1\\ \alpha_{i\neq 1} = \beta_i &= 0.\end{aligned}$$

The rest of the tangent patch construction is identical in Section 3.2 except that we substitute $n = 4$ for the vertex contained by only one quad.

ACKNOWLEDGMENTS

We would like to thank Peter-Pike Sloan, David Blythe, and Henry Morton for many helpful discussions and insights, as well as Bay Raitt for the models of the “monster frog” and “big guy.”

REFERENCES

ATI/AMD. 2007. ATI Radeon HD 2900 Series—gpu specifications. http://ati.amd.com/products/Radeonhd2900/specs.html.

BIERMANN, H., LEVIN, A., AND ZORIN, D. 2000. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of SIGGRAPH Computer Graphics*. 113–120.

BLYTHE, D. 2006. The direct3d 10 system. *ACM Trans. Graph. 25*, 3, 724–734.

BOLZ, J. AND SCHRÖDER, P. 2002. Rapid evaluation of catmull-clark subdivision surfaces. In *Proceeding of the International Conference on 3D Web Technology*. 11–17.

BOYD, C. 2007. The furture of directx. http://download.microsoft.com/download/e/5/5/e5594812-cdaa-4e25-9cc0-c02096093ceb/theGame DeveloperConference.

CATMULL, E. AND CLARK, J. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Comput. Aid. Des. 10*, 6, 350–355.

CHIYOKURA, H. AND KIMURA, F. 1983. Design of solids with free-form surfaces. *Comput. Graph. 17*, 3, 289–298.

DEROSE, T., KASS, M., AND TRUONG, T. 1998. Sudivision surfaces in character animation. In *Proceedings of SIGGRAPH Computer Graphics*. 85–94.

BUNNELL, M. 2005. *Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping: CPU Gems 2.* Addison-Wesley, Chapter 7, 109–122.

HALSTEAD, M., KASS, M., AND DEROSE, T. 1993. Efficient, fair interpolation using catmull-clark surfaces. *Comput. Graph. 27*, Annual Conference Series, 35–44.

LEE, M. 2006. Next-generation graphics programming on xbox 360. http://download.microsoft.com/download/d/3/0/d30d58cd-87a2-41d5-bb53-baf560aa2373/Next_Generation_Graphics_Programming_on_Xbox_360.ppt.

NASRI, A. H. 1987. Polyhedral subdivision methods for free-form surfaces. *ACM Trans. Graph. 6*, 1, 29–73.

PETERS, J. 2000. Patching catmull-clark meshes. In *Proceedings of SIGGRAPH Computer Graphics*. 255–258.

REIF, U. 1995. A unified approach to subdivision algorithms near extraordinary vertices. *Comput. Aid. Geomet. Des. 12*, 153–174.

SHIUE, L.-J., JONES, I., AND PETERS, J. 2005. A realtime gpu subdivision kernel. *ACM Trans. Graph. 24*, 3, 1010–1015.

STAM, J. 1998. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Comput. Graph. 32*, Annual Conference Series, 395–404.

STAM, J. AND LOOP, C. 2003. Quad/triangle subdivision. *Comput. Graph. Forum 22*, 1, 1–7.

VLACHOS, A., PETERS, J., BOYD, C., AND MITCHELL, J. L. 2001. Curved pn triangles. In *Proceedings of the Symposium on Interactive 3D Graphics*. 159–166.

Received June 2007; accepted November 2007