Ptex system for advanced texture mapping

Andrija Bernik, Mario Kotlar

University North, University center Varaždin, 104. brigade 3., 42000 Varaždin, Croatia

*E-mail: abernik@unin.hr

Abstract

Ptex is a texture mapping method. It's main advantage is the elimination of UV mapping process by storing separate texture for each face. Ptex deals with the filtering across face boundaries, which makes all the little separate textures look seamless, as if they were one big texture. Ptex file format can efficiently store hundreds of thousands of images into a single file. Ptex API is released as open source, written in C++, which enables implementation of Ptex in various programs. This paper analyses advantages and drawbacks of this system and examines if Ptex is today usable outside of big production studios. Analysis leads us to a conclusion that Ptex can already be used by individuals outside of big production studios.

Key words: Mudbox, Ptex, rendering, texture mapping, VRay

1. Introduction

Brent Burley and Dylan Lacewell conducted a variety of atlase and volumetric mapping experiments in the Walt Disney Animation Studio in an attempt to get UV mapping methods that would be more generalized and more effective but failed to get satisfactory results in filtering the texture by render engine (which was in this case was Pixar's RenderMan, which is also the most useful renderer in the industry). Prior to the split surface modeling, NURBS patches were standard for modeling and mapping the texture, trivial for embedded 2D parameterization within the NURBS system itself and their logical modeling. The idea of Burley and Lacewell was to map the texture divided by the surface model to the level of simplicity that NURBS patches offered so far. Their experiences Burley and Lacewell presented at the symposium (Eurographics Symposium on Rendering) in the work of Ptex: Texture design by product for production, and it will, with further clarification, be transmitted largely in the continuation of the theoretical part of this paper. [1]

It was necessary to create a new texture mapping method that had to satisfy four conditions:

- **Film quality**. The method must be able to provide anisotropic filtering, must have no visible spatial or temporal alignment, it must support smooth filtering for displacement maps, and must not have visible seams. - **Generality**. The method must work on any geometric model, regardless of topology or complexity.
- **Efficiency**. The method must be rational in storing graphics data without spending excess storage space. It needs to have an efficient I / O (Input / Output), that is, the communication between code, renderer and texture drawing application must be fast enough to enable realtime drawing where every entry of new information by the artist must be processed and shown on the screen fast enough for continuous work. The method must be efficient and must not consume too much work memory, nor be too intensive for the CPU (Central Processing Unit).

- **Automation**. The method must allow the models to be quickly and efficiently mapped without any special preparation. [1]

The new method developed by Burley and Lacewell has succeeded in achieving all the stated goals using per-face parametrization inherent subdivision geometry. This idea is not new in itself, but it has never been done to achieve the film quality of filtering per-face textures. A key advantage in using parametrization that is inherent in subdivision geometry is that it completely frees us from the painstaking process of manual UV mapping. UV mapping is a responsibility of the Visual Development Department within the traditional film studio. This same department also produces textures and shaders that define the look of the model. For time efficiency, this department works simultaneously with most other departments, which means that many departments have access only to unparametric models, or models where the development department has not yet defined UV maps. Even in the case when UV mapping is made immediately after, or during the development of the model, the Visual Department may re-model the model at any time, if the current UV layout does not provide an adequate resolution where it is needed, which would ruin any existing textures for that model, including any textures that some other departments may have made. For these reasons, most departments do not have access to mapped texture models. With Ptex, this is no longer the case.

During the development, many problems were encountered, the main ones being filtering across the polygon boundaries, filtering with very large filter widths and anisotropic filtering. All of these problems were finally solved successfully.

The problems that the Ptex method solved, and for which solutions did not already exist in other methods, are the following: (1) data structures and methods for effectively storing and accessing data on adjacent polygons within the texture - maps; (2) a method for seamless, anisotropic filtering through a polygon of different resolutions that uses data on adjacent polygons to enable proper filtering; and (3) a method for blending textures over a polygon that has the ability to perform filtering when individual polygons are smaller than pixels in the final image. [1]

Previous methods of texture mapping (pelting, atlas, projection, volumetric and per-face) will be evaluated according to the aforementioned list of conditions (film quality, generality, efficiency and automation). The results of the evaluation are shown in Table 1.

Table 1. Comparison of texture mapping methods [1]

	Film quality	Generality	Efficiency	Automation
Pelting	✔		✔	
Atlas method	maybe	maybe	✔	
Projection	✔			✔
Volumetric textures		✔		✔
Previous per-face methods		maybe	✔	maybe
Ptex	✔	✔	✔	✔

When all advantages and disadvantages are considered, only two of all these methods, pelting and projection, support production quality filtering, but neither of these two methods is general, and the projection is particularly inefficient. Before Ptex, most models in the Walt Disney Animation studio were disassembled into smaller parts, which could then be mapped using standard methods, which unfortunately always came under the price of increased complexity and increased system resource consumption.

2. Main part

2.1 Per-face parametrization

Ptex stores separate textures for each polygon of subidivision-control geometry, each of which can independently change the resolution, and each surface can be mapped using a single Ptex file that contains all per-face textures, regardless of the level of complexity of the geometry. The Ptex file also contains information about adjacent polygons within the texture used by the renderer to filter the texture across the polygon. Seamless filtering is extremely important to avoid lighting errors when using displacement folders.

The input value of the Ptex algorithm is an arbitrary object whose geometry consists of quads (four-sided polygons). Such an object is, by itself, complete and uniquely parametrized using coordinates $faceid, u, v$ here $faceid$ s the position of each polygon within the geometry description. u nd v re parameters inherent to subd parameterization, and UV orientation is defined from the order of the vertex, which is assumed to be in the usual order $0,0,1,0,1,1,0,1$ Althou *faceid* it is not continuous over the surface, once the orientation is taken into account, the parametrization is still C^2 ()ontinuous across all polygons on the Catmull-Clark limit surface. For example, in the geometry of the figure 1, $\partial P / \partial u$ n polygon 12 is aligned with $\partial P / \partial v$ n polygon 8.

Each quad polygon has 4 edges, so it has 4 *edgeid* hich are numbered from 0 to 3 and follow the order of the vertex. The adjacency data required for filtering is stored in array variables.

Array consists of 4 faces that belong to the polygons with which the reference polygon borders (the quad polygon can always border with a maximum of 4 polygons) and the edges for each of these four polygons (which help to orient the polygon).

Faceid value -1 signifies that the edge of the polygon is on the edge of the object, that is, that there is no polygon that connects to the reference polygon on that edge. For example, the polygon 7 shown in Figure 1 would have the following data on adjacent polygons:

$\{\text{adjfaces } -1,8,17,-1 :;\text{djedges}$
$x,3,0,x, x = don't\,care :\}$

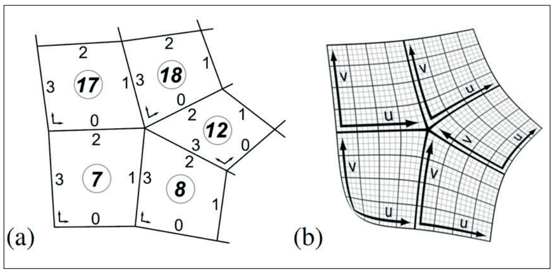

Figure 1. a) A part of the subd control mesh that displays the unique face-id and edge-id values. b) The appropriate limit-surface on which the continuity of the iso-line is seen over the entire surface. [1]

The values of the adjfaces poprilično are quite clearly shown in Figure 1, and the *adjedges* values indicate which edge of the polygon 8 this reference polygon borders, and in this case it is the edge 3. In defining the boundary with polygon 17, it is written that it borders with its edge 0. The X variable is used where the polygon, or its edge, does not borders with another polygon.

Thus, from the data $x,3,0,x$ the orientation of the adjacent polygon relative to the reference polygon (which is 90°), can be calculated using the formula $r = \left(e_i - e_j + 2\right)$ od 4, and the vertex valence can be determine using a simple traversal:

- mod 4 denotes the remainder when dividing by 4.
- valence is the number of lines that connect in one point, or vertex. [2]
- traversal is a recursive program algorithm that examines each node in a tree data structure (in this case, a geometry consisting of points and lines defining polygons). It has various types of traversal algorithms that are adapted for testing different tree types of data structures, because the tree data structure has significant variability.

32 bits are assigned to each faceid, and 2 bita are assigned to each edge. Thus, the amount of memory on the polygon occupies data on adjacent polygons of only 17 bytes per polygon. $32 + 2 = 34$ (ts for each side of the polygon, and given that each polygon is quad, ie 4 sides, $34 \times 4 = 136$ 36 bits, which is 17 bytes per polygon). These 17 bytes are further compressed within the Ptex file with other data in the header. [3]

2.1.1 Tris and ngon in quad geometry

Subsequently, support for the tris and ngones was added to Ptex. For a geometry that is generally quad, but also has some non-quad polygons, the Ptex file stores one texture per sub-pixel, as shown in Figure 2. The data on adjacent polygons are generally the same as in the geometry that consists only of the quad polygon. An addition is that now every subface has a faceid and references its four adjacent polygons and edges. The only change in the data on adjacent polygons is an additional flag in the FaceInfo data structure that indicates whether the polygon is actually a subface.

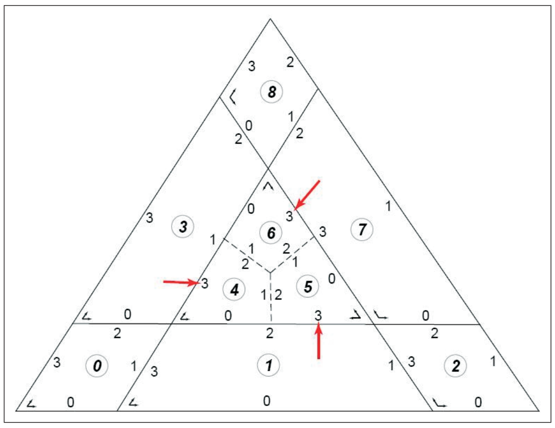

Figure 2. An example of a tris geometry. It is visible that the tris is effectively divided into three subfaces (boundary between the subface is indicated by a broken line). [4]

There is one important detail that is specific to quad / non-quad edges. When the quad polygon is located next to the non-quad polygon, it will have 2 sub-neighbors on that edge, and can only be referenced to one of them within the data on adjacent polygons. In this situation, the first sub-surface encountered by a traverse counterclockwise (for example, the edgeid order) will be referred to as an adjacent polygon.

In the example shown in Figure 2, these cases are marked with red arrows: polygon 1 would reference to polygon 5, edge 3 as its neighbor (and not polygon 4, edge 0). Polygon 7 would refer to polygon 6, edge 3, and polygon 3 would refer to polygon 4, and edge 3. A filter that calls data on neighboring polygons will expect this convention. This approach is also a useful traversal (using the data on neighboring polygons) because it does not have to worry about whether the polygon is actually a subface or not. Polygons with N edges (ngons) are mapped from N subfolders and followed by the same rules. [4]

2.1.2 Tris texels

Another option is subsequently added to Ptex, which is the ability to use the tris texel. Tris texels are used in cases where the user does not want to work with the quad geometry and the Catmull-Clark subdivision algorithm, but with the tris geometry and the corresponding loop subdivision algorithm.

For easy storage and access to data, Texel are divided into odd and even (Figure 3). The unwanted texels then turn around to form a quad formation, which can then be stored as a standard Ptex quad texture, with the indication that it is three-dimensional. Although the data on adjacent polygons is identical as for quads, the fourth values are ignored in adjfaces and adjedges, which shows additional differences between tris and quad textures. Also, three textures do not support subfaces, because their concept is unnecessary in geometry which consisting only of tris.

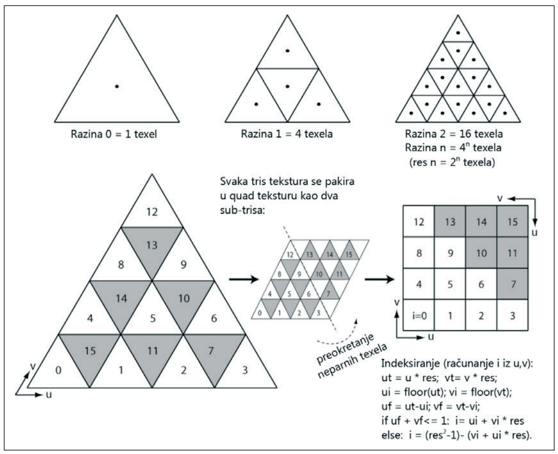
Razina 0 = 1 texel
Razina 1 = 4 texela
Razina 2 = 16 texela
Razina n = 4^n texela
(res n = 2^n texela)
Svaka tris tekstura se pakira u quad teksturu kao dva sub-trisa:
preokretanje neparnih texela
Indeksiranje (računanje i iz u,v):
ut = u * res; vt= v * res;
ui = floor(ut); vi = floor(vt);
uf = ut-ui; vf = vt-vi;
if uf + vf<= 1: i= ui + vi * res
else: i = (res^2-1)- (vi + ui * res).
i=0

Figure 3. Tris texels, their packaging and indexing. [5]

For GL display (eg OpenGL, Direct3D,...), the tris polygon can be rendered directly from the lower half of the texture (Figure 4). A small distance from the edge should be used to keep the polygon secure within the texture. Alternatively, the shader can make point-sample complete textures using the indexing method shown in Figure 3.

For color projections (eg rasterizing the three-dimensional texture in the quad texture - Figure 5), two projection screens are needed to cover the entire quad. A 1/6-point offset is also added, so that the points at which the program takes color patterns match the quad-core environments. [5]

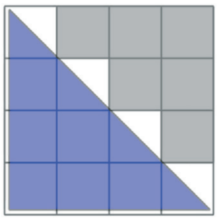

Figure 4. Method for rendering tris textures for GL. [5]

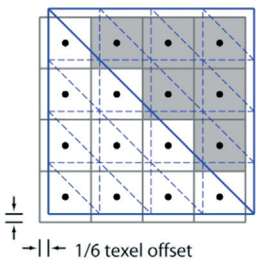

Figure 5. Rasterizing tris into quads. [5]

2.2 Size and storage of texture

Selecting the size of the texture in the textiles, depending on the surface, the process is automated, which is calculated as the derivative of the surface of each individual polygon.

These values are automatically calculated as soon as the 3D model is loaded into the 3D texturing application. Then the user can adjust the global texture resolution depending on the level of detail that will be required on the texture of that model. The global resolution setting is increased or diminished twice each time. In addition, users also have complete freedom to locally increase or decrease the resolution in the texels for each individual polygon, which is an option that is generally not available in other texture mapping systems, which can be very useful. For example, a user can increase the resolution of only those polygons that are known to be close to their camera. [1]

2.3 Mipmaps and ripmaps

Ptex stores box-filtered reduced versions of each texture, the so-called mipmap, within the Ptex file. Instead of using the usual mipmap store system in a pyramidal structure, Ptex stores them in layers of resolution. The first layer contains texture in full resolution, each of which contains a reduced version of the texture. Each texture within the layer is still stored in its own memory block that can be accessed directly. The layers of resolution serve mainly to optimize the time it takes to access the mipmap data inside the Ptex file. [1]

Mipmaps are achieved by the symetric reduction method, which ensures that the original texture is symmetrically reduced in axis, that is, its height is reduced as much as the width, so that the aspect ratio remains the same. Mipmaps are needed to increase the rendering speed and reduce the aliasing of artefacts in the final rendering. Mipmaps also speed up the rendering process because they enable the render engine to use the version of the texture of less resolution when it is not necessary to use maximum resolution, because the object is far away from the camera and the original texture details would not be seen on the final rendering, even maximum resolution texture was used. In many cases, filtering should not be the same on both axes, or should be anisotropic, which is the opposite of isotropic filtering (filtering that treats both axes equally). [1]

Mipmaps that support anisotropic filtering are called ripmaps. Some texture mapping systems work on the ripmap principle, but this is not a way that is often used in practice, as it takes up a lot of extra memory. Ripmap (asymmetric reduction) increases the texture size by 300% while mipmaps increase the texture size by only 33⅓ % symetric reduction). [6] Ptex uses asymmetric reduction to achieve anisotropic filtering, as without them the filter would have to read the Ptex file more intensely accessing one-axis data, which is not efficient and would significantly slow down the entire process. To avoid 300% increased texture size and the time required to enter and print data, Ptex generates asymmetric reduction dynamically using the closest matching resolution of the stored mippape, an option that has not been seen before in other texture mapping systems. [1]

2.4 Filtering

Ptex supports anisotropic filtering over surfaces of different resolutions while at the same time supporting the reduction and increase of these resolutions.

The filtering method for Ptex was originally designed for the Pixar Photorealistic Renderman (PRMan), but it can be customized for any other rendering that offers a production quality level.

2.4.1 The usual case

In REYES (Renders Everything You Ever Saw [7]) renderer such as PRMan, mapped textures are placed in locations called shading points, as shown in Figure 6. Renderer cuts geometry into a rectangular shading network and sends a spacing to the shader via the built-in variables du nd dv Since the Catmull-Clark subdivision surface cuts one by one a polygon, along the parametric lines u and v, the per-face textures are aligned with shading nets and the filter region is simply a rectangle of size $du \times dv$

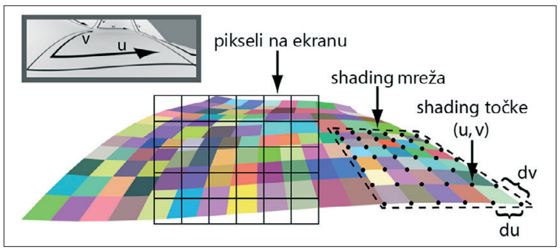

Figure 6. Quad subdivision polygon

Figure 6 shows the quad subdivision polygon, which is divided into micro-gallons that are approximately the size of one pixel on the screen. In order not to confuse, it should be noted that the colored squares in this figure do not represent the Texel, but the micro-polygons.

The size of the micro-polygons, $du \times dv$ s constant within each particular shading network. [1] This allows anisotropic filtering to be achieved with a separable filter. Any filter kernel could be used for the Ptex method, but separable filters can be calculated faster, so they are more time-efficient than radial or elliptical filters.

In order to achieve the production quality level of filtering, Ptex uses *bicubic filter kernel*:

$$k(x)=\frac{1}{6}\begin{cases}(3+6S)|x|^3+(-6-9S)|x|^2+(4+2S) & \text{if}\,|x|<1\\(-1-2S)|x|^3+(6+9S)|x|^2+(-12-12S)|x|+(8+4S) & \text{if}\,1\leq|x|<2\\0 & \textit{otherwise}\end{cases} \quad [1]$$

The parameter S is shown to the user as a sharpness control. For displacement maps, it is best to use S = 0, which results in C^2 ubic B-curve. For color maps, the B-curve results in a blurry texture display, so a sharper value is needed. The value S = 1 is the recommended color shift value because it results in a significantly sharper Catmull-Rom interpolated curve, but any value from 0 to 1 can be used if necessary.

The value $S = 2/3$ for example, results in a popular Mitchell filter. Kernel to wich $S \neq 0$ re C^1 which is sufficient for most maps.

For a given filter region of $du \times dv$ we choose a texture resolution $R_u \times R_v$ hich has texels just slightly less than the size of the filters:

$$R_u = 2^{\log_2 \frac{1}{du}}; R_v = 2^{\log_2 \frac{1}{dv}} \quad [2]$$

Then the kernel weights are:

$$k_{ij}(u,v) = k\left(\frac{u-\left(i-\frac{1}{2}\right)/R_u}{du}\right)k\left(\frac{v-\left(j-\frac{1}{2}\right)/R_v}{dv}\right) \quad [3]$$

and a filter convolution with texel data d_{ij} s:

$$f(u,v) = \frac{\sum i \sum jk_{ij}(u,v)d_{ij}}{\sum i \sum jk_{ij}(u,v)} \quad [4]$$

The width of the filter will vary between 1.0 and 2.0, and the bicubic kernel (which supports a width of up to 4) will require between 4 and 8 samplings in each direction. When the kernel is provided over the edge of the polygon, divide the kernel and perform a convolution (we calculate the level of overlap), depending on the division, with each overlapping polygon, as shown in Figure 7.a. To avoid confusion, this is not about overlapping the polygon, but about overlapping the kernel and the edge of the polygon. If the kernel overlaps with the edge of the polygon, we use the influence of the kernel of the corresponding edge texels (Figure 7.b). This is often called border clamping. If the adjacent polygon has an insufficiently high resolution, or the kernel overlaps with an extraordinary vertex, e.g. with the vertex of the valence $N \neq 4$ then special methods for each of these two individual cases are used.

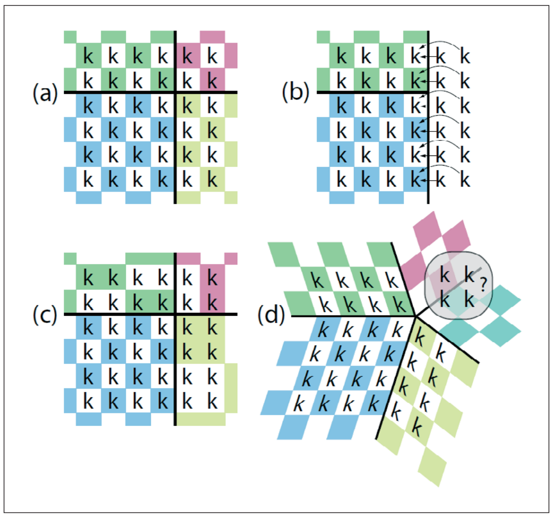

Figure 7. 6×6 ernel which overlaps with the angle of the polygon. [1]

a) In the normal case, the kernel is applied by the appropriate part to each polygon with which it overlaps, in order to get a seamless result.
b) In the case when the kernel crosses the edge of the polygon, the corresponding edge texels are used for influence.
c) In the case where the boundary polygon does not have the required resolution level, the closest matching texel is used for the influence of each kernel.
d) In addition to the extraordinary vertex, it's less clear how to apply a rectangular kernel.

2.4.2 A case of insufficiently high resolution

When the adjacent polygon has an oversized texture resolution for the current filter size, the texels can not be directly aligned with the kernel effects. In this case, we apply the influence of each kernel to the nearest corresponding weight, as shown in Figure 7.c. The method of border clamping could be used (as is the case with NURBS patches), but when used with displacement mapping, it results in wrapping the displaced normals to a non-slip surface, as shown in Figure 8.a, and can cause significant lighting errors. Filtering with edge textures results in much more consistent normals, as shown in Figure 8.b, and typically results in a minor discontinuity in the surface. Discontinuity, by itself, is not a problem, as PRMan automatically fills the gap with micro-polygons.

The problem, when the resolution fails, only appears to increase, which is rarely produced because artists always try to create textures with more than enough resolution. Also, this problem leads to large visible disturbances in large and mild displacement. For a moderate amount of magnification (zoom), the mesh spacing will be close to the spacing of the Texel, and the methods described here will be quite sufficient to hide any seams, and discontinuity will not be greater than what causes a transition between the prefilter levels of resolution. If the high magnification level is important, any interference can be completely resolved using the same resolution level across the entire model.

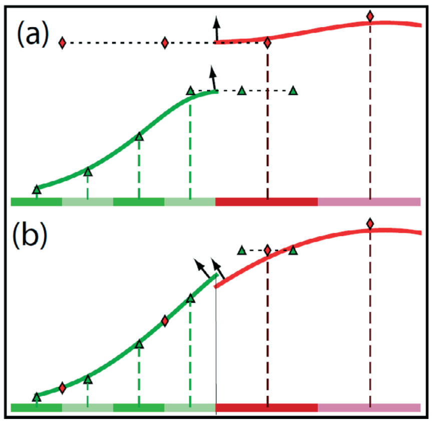

Figure 8. Two adjacent dislocated polygons with different resolution textures, shown in the 1d cross section. [1]

2.4.3 A case of extraordinary vertex

The kernel split method on up to 4 parts works well next to the regular vertex. It is less clear how to expand the rectangular kernel via EV (Extraordinary Vertex) (Figure 7.d).

Under minification or light magnification, the EV area is subpixel (less than pixels) and the ideal smooth reconstruction is not important. We can simply ignore the polygons in the corner (indicated by the questionnaire - Figure 7.d) and renormalize the kernel. This creates discontinuities along the edges next to the EV, but the disorders that occur there are negligible. Under significant magnification of large, mild displacement, seam artefacts can be visible. In that case, Catmull-Clark reconstruction can still be used, but so far in practice it was not necessary yet.

2.4.4 Large filter widths

When the rendered polygon is less than one pixel, the PRMan will only sample the angles of each polygon, and the width of the filter, du nd dv will increase over 1.0 (eg larger than the polygon). Furthermore, when many polygons are smaller than pixels on the screen, the renderer can undersamplify the surface (because only 1 micro-polygon per pixel sample is used) and this can lead to an aliasing. To manipulate filter widths greater than 1.0, a special prefiltering method is used where 1×1 exel per-face textures consistently mix through box-filtering with their neighbors. Conduction to the box filter n times generates a B-curve of order n and it has an effective filter width, proportional to $sqrt(n)$ Box filter of 3 units implemented through a convolution n times has an effective filter width of approx. $\frac{3}{2}sqrt(n)$. he required amount of calculation is insignificant, because they are used only for 1×1 er-face textures, and the filtered value is sufficient to compute once and then put in a memory cache for reuse.

2.5 Additional features

The Ptex file can be modified to support the texting of multiple separate objects within a single Ptex file. It is necessary somewhere inside the file to write indices for each object and to customize the code to read the textures, so that it can take that into account. This is possible because Ptex is open source (written in

C ++) [8], but advanced programming knowledge is required for such an adjustment. This is easier to do by connecting two objects, which are only then textured, but it is generally better to give each object its own Ptex texture, which is the most commonly automated process within the texture drawing program.

Ptex is not designed to support filtering on the GPU. Such a display is of course possible, but it will result in seam artifacts, which users can ignore if it's easier for them to work. In other words, the filtered quality level is not required for the model to be efficiently textured. It should be emphasized that GPU rendering inside games is not the same as GPU rendering for production.

There is currently no information on whether Ptex will be implemented in new experimental GPU renderers such as, for example, FurryBall and Octane renderer, but are speculated that it should be feasible [2]. It can be assumed that this will happen when GPU rendering becomes a new industry standard.

2.6 Ptex in practice: Disney experience

"Bolt" is Disney's first CGI animated full length film in which Ptex was used as the primary method for mapping textures. Bolt has over 100,000 subdivision surfaces that are textured using this method, using an average of 7 layers of surface texture and 2.3 mega-Texels per Ptex file. The largest Ptex file was larger than 3.3 giga-Texels, a level of resolution that can not be achieved by traditional texture mapping methods. [1]

In the Disney animation studio, the transition to Ptex has improved pipelines and the efficiency of artists, and there has also been a significant improvement in the input and output performance within the study. The reason for this is a significantly reduced number of individual texture files that are easier to exchange between artists and departments.

Also, during the rendering of the "Bolt" movie, the use of Ptex reduced the server load for input / output textures. The processor load during rendering during the rendering of the "Bolt" movie turned out to be the same sometimes, and sometimes faster than the previous texting methods. [1]

Below is a test for comparing the efficiency and performance of Ptex (Table 2), compared to the method they used before, called Per-patch texture.

Table 2. A comparative test between Per-patch texture and Ptex

1024x1024 pixels	**Per-patch teksture**	**Ptex**
CPU secs	171	141
# I/O calls	18,581,058	9,209
20x20 pixels	**Per-patch teksture**	**Ptex**
CPU secs	27	2
# I/O calss	4,218,733	84

The test clearly showed that Ptex is better for CPU performance, but also that it is far superior to the number of Input / Output calls that the renderer needs to perform in order to render when accessing data on where the part of the texture is mapped. Per-patch textures had a lot of problems because each patch had to use separate files with a separate texture, which proved to be problematic in the model of greater complexity. For example, in Disney's "Meet the Robinsons", it resulted in over 100,000 textures per rendering and a total of 6 million textures for the entire movie, of which the average texture size was 30 KB, and the median texture size 2.5 KB. [1]

2.7 Ptex Supported Applications and Applications

Before starting the next chapter with the presentation of the results of the practical work, there are some drawing applications that support Ptex, some of which were used in the test:

1. **Paint3D**, Disney's in-house tool, for the first and longest time, is the only program to create Ptex textures with full support for all Ptex methods. Unfortunately, it is not possible to get to it because Disney does not sell it.
2. **Mari**, probably the best texture drawing program that recently received support for Ptex. It should be noted that it requires exceptionally strong computer configuration to run, but also supports high resolution and complexity of textures. [9]
3. **Mudbox**, program from Autodesk with good Ptex support. One of its disadvantages is that it can not import Ptex files in which Trises are textured. It is interesting

that it can introduce geometry with tris and ngons, texture it and it is successful to export. However, this deficiency is not so important to him at the moment, because the Ptex texture and tris model can be saved in Mudbox's native .mud format.[10]

4. **3D Coat**, the program that was the first commercial program to get support for Ptex. Unfortunately, it does not support Tritex and Subfaces, so it works well only with a complete quad geometry. It is also possible to introduce a geometry containing tris, but this results in the automatic Catmull-Clark subdivision of geometry, which in the vast majority of cases is not acceptable for the user. It also has problems with compatibility with other programs and renderers, so before using it, it is necessary to examine the current level of compatibility with the rendering engine that it later intends to use. 3D Coat also emphasizes the unique possibility of importing Ptex texture as a UV folder, which is then compatible with each program. The problem with such an approach is that it is no longer a Ptex method because it does not provide the rendering quality of the rendering.
5. **Stripes**, a free program with a completely basic set of tools for drawing Ptex textures. It does not support the tablet, neither Tritex nor Subfaces. [11]

Rendering applications that support Ptex:

1. **PRMan**, the leading renderer in the industry. Renderer for which Ptex was originally created. [12]
2. **VRay**, a popular raytrace renderer that Ptex has received in its version 2.0.
3. **Mantra** renderer, renderer of the Houdini program. Like PRMan, it belongs to the REYES renderer category (based on micropolygon rendering technology).
4. **Cinema 4D** (a plugin that is not free). It supports all Ptex elements and is well integrated with Cinema 4D.
5. **Stripes** (free plugin for Mentalray for Maya and 3ds Max). It is located in the alpha stage of development, it is very unreliable and has poor documentation (Maya plugin does not have documentation at all)[11]

3. Conclusion

This paper presents an advanced texture mapping method that is optimized for subdivision models but can also be conveniently used for non-subdivision models in accordance with the comments of the creators of Burley and Lacewell. It has been shown that Ptex efficiently stores and accesss data on neighboring polygons and uses these data for seamless filtering over the edges of the polygon. It is explained that the Ptex method meets all 4 conditions that are important for industrial production and according to which other methods have been evaluated:

- Ptex provides movie quality by supporting seamless anisotropic filtering of texture maps, as well as mild displacement maps.
- Ptex is general. It works regardless of the level of complexity of the geometry and the size of the resolution, allowing levels of detail that were previously not feasible. Ptex successfully manages to save details better than the previous method by aligning its texture-grid with the shading grid.
- Ptex is efficient because it provides independent resolution control individually for each polygon, it requires only one file per object and significantly reduces memory and input / output requirements.
- Ptex is automated. It is not necessary to make UV folders or anything similar, which allows access to the text mapping maps for each department within a production study.

Results are presented that confirm that Ptex is the most effective method for mapping textures for film production.

4. Reference

1. Burley, B., Lacewell, D., 2008. Ptex: Per-Face Texture Mapping for Production Rendering. InComputer Graphics Forum 2008 Jun 1 (Vol. 27, No. 4, pp. 1155-1164). Blackwell Publishing Ltd.
2. Alliez, P., Gotsman, C., 2005. Recent advances in compression of 3D meshes. InAdvances in multiresolution for geometric modelling (pp. 3-26). Springer, Berlin, Heidelberg.
3. Jacobson, G. 1989. Space-efficient static trees and graphs. InFoundations of Computer Science, 1989., 30th Annual Symposium, (pp. 549-554). IEEE.
4. http://Ptex.us/adjdata.html, srpanj 2017.
5. http://Ptex.us/tritex.html, srpanj 2017.
6. Yang R., Pollefeys, M. 2003. Multi-resolution real-time stereo on commodity graphics hardware.

InComputer Vision and Pattern Recognition, 2003. IEEE Computer Society Conference (Vol. 1, pp. I-I).

7. Zhou, K., Hou, Q., Ren, Z., Gong, M., Sun, X., Guo, B., 2009. RenderAnts: interactive Reyes rendering on GPUs. InACM Transactions on Graphics (TOG) 2009 Dec 17 (Vol. 28, No. 5, p. 155). ACM.
8. http://Ptex.us/documentation.html, siječanj 2018.
9. http://www.thefoundry.co.uk/products/mari/, srpanj 2017.
10. http://usa.autodesk.com/adsk/servlet/pc/index? id= 13565063&siteID=123112, srpanj 2017.
11. http://www.mankua.com/Stripes/Stripes.php, srpanj 2017.
12. Apodaca, AA., Mantle, MW. 1990. Renderman: Pursuing the future of graphics. IEEE Computer Graphics and Applications 10., (4):44-9.