Eurographics Symposium on Rendering - Experimental Ideas & Implementations (2017)
P. Sander and M. Zwicker (Editors)

# Ambient Dice

Michał Iwanicki[†1] and Peter-Pike Sloan[‡1]

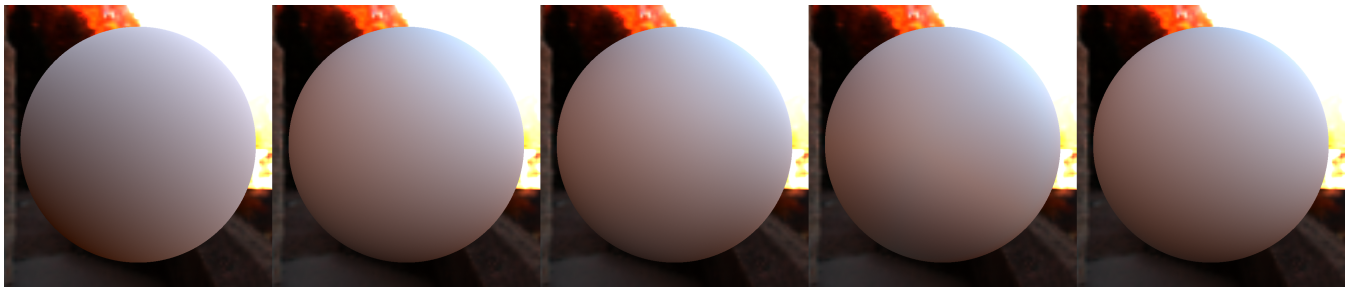


**Figure 1:** *Diffuse irradiance reconstructed using (left to right): 2nd order Spherical Harmonics, 3rd order Spherical Harmonics Ambient Dice (our method) using the RBF variant, Ambient Dice (our method) using the YCoCg variant, importance sampling (reference)*

**Abstract**

*We present a family of basis functions designed to accurately and efficiently represent illumination signals on the unit sphere. The bases are built of locally supported functions, needing three to six basis functions in a given direction. This minimizes the number of memory transactions and bandwidth requirements needed for reconstruction.*

*There are three variations of our basis. All are based on storing coefficients at the 12 vertices of an icosahedron. The first one stores the values directly, together with their directional derivatives and hybrid Bézier patches are used for interpolation. This allows one to achieve quality comparable to 3rd-5th order spherical harmonics while still requiring 27 coefficients for the reconstruction. The second variation encodes the signal in YCoCg space and uses a reduced quality, linear reconstruction for the chromaticity components - requiring only 15 coefficients while marginally reducing the quality. The third option exploits the partition of unity formed by* $\cos^2$ *and* $\cos^4$ *restricted to a hemisphere oriented along the directions of the icosahedron vertices. It uses 18 coefficients for the reconstruction, but trades the additional bandwidth requirements for simpler calculations. The quality of that version is still comparable to 3rd order spherical harmonics (SH).*

*We name the basis Ambient Dice as a reference to both: the Ambient Cube basis - as ours is an extension of some of its properties - and the 20-sided dice commonly used in pen-and-paper role-playing games, which is an icosahedron.*

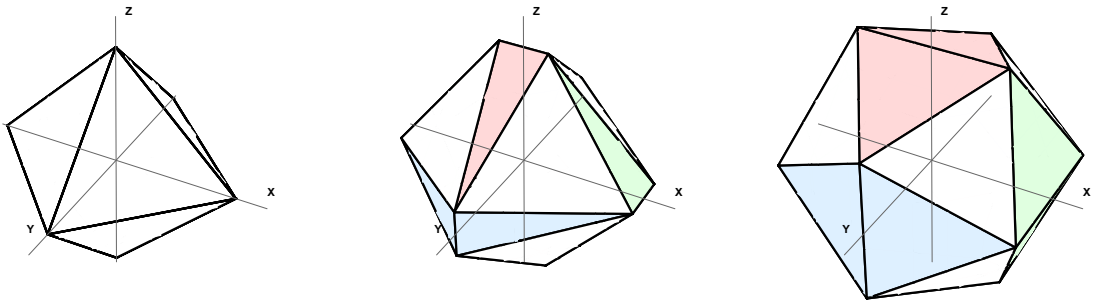
## 1. Introduction

Modern games typically require some way of representing indirect illumination. For diffuse lighting the most common solution is to store irradiance in a volumetric data structure, either regular [GSHG98, McT04] or irregular [Cup12]. The data must be represented in some directional form for the lighting to respond to normal variation. The two most popular solutions are the Ambient Cube (AC) [McT04] which stores the irradiance projected onto six $\cos^2$ hemispherical lobes oriented along the cardinal axes, or low order spherical harmonics (SH), where the irradiance is stored as a set of coefficients for mutually orthogonal basis functions defined over a unit sphere [RH01].

Both solutions come with limitations. The AC basis is formed of wide lobes covering whole hemispheres with little overlap, which makes it poorly reproduce irradiance signals. It is, however, simple and efficient at runtime - for any direction only three of the six basis functions are non-zero and need to be brought from memory.

Spherical harmonics offer much greater precision. The SH basis functions cover the whole sphere, evaluating in any direction all of the coefficients are used. For 3rd order SH it's 9 coefficients for each of the three color channels, 27 total, which is expensive and

† e-mail: michal.iwanicki@activision.com
‡ e-mail: ppsloan@activision.com




**(a)** *Base octahedron* **(b)** *Intermediate stage* **(c)** *Resulting icosahedron*

**Figure 2:** *Construction of an icosahedron from an octahedron*

games often revert to 2nd order SH (4 coefficients for each channel), which has lower quality. Additionally, SH can exhibit artifacts typical for the truncated-spectrum-type solutions - discontinuities in the input signal cause ringing and extra processing of the coefficients [Slo08] needs to be performed to eliminate it.

Recently, some developers began using a sets of 9-12 Spherical Gaussian lobes (SGs) [TS06] as a representation of the radiance signal, that's later convolved with the cosine lobe to form irradiance [NP15] and while those bases show fewer ringing artifacts, the SGs also have global support over the unit sphere, so all the coefficients are needed during evaluation.

Our aim was to design an alternative to those bases. On modern GPUs, bringing excessive amounts of data from memory to compute units can pose performance problems. We wanted to combine the local support of the functions that make up the basis - so that the evaluation would only need a handful of coefficients, limiting both bandwidth and register needs - with the quality of reconstruction typical for more complex bases. We assumed that overall memory consumed by the representation is less of a concern, and seek a better balance between bandwidth and computation.

We call the proposed basis Ambient Dice (AD) - as a reference to both Ambient Cube and the 20-sided dice commonly used in pen-and-paper role-playing games. Of the variations described in the paper, we think two are promising: Spherical Radial Basis Functions (SRBFs) with 12 hemisphere lobes, and the other uses the YCoCg color space and Bézier patches, requiring 18 and 15 coefficients respectively to reconstruct irradiance for a given normal.

## 2. Ambient Dice

### 2.1. Description

The AD basis stores values at the vertices of a regular icosahedron, inscribed into a unit sphere. An icosahedron is composed of 12 vertices and 20 equilateral triangles. All the vertices lie on the surface of a sphere and projections of the triangles onto the sphere form 20 identical spherical triangles. Evaluation of the function in any direction requires data from the three vertices at the corners of the spherical triangle that contains the direction. Subdivision of a sphere is often used in meteorological modeling, where the unit sphere represents the Earth. The projection from an icosahedron to the surface of a sphere has low distortion and many operations can be performed directly on the flat triangles making up the icosahedron, rather than on spherical triangles, without sacrificing quality.

### 2.2. Indexing an icosahedron

To use icosahedron as a basic building block, an efficient way of indexing it is needed. One option is to determine the triangle that a given direction points to one could iterate over icosaheron triangles, or use some form of spatial hierarchy to make the search more efficient. This results in complicated code, with a lot of branching, which doesn't execute efficiently on modern GPUs. To simplify the indexing we rely on a construction of an icosahedron from a regular octahedron described in [Ban96].

An octahedron is placed in the origin of the coordinate system, with the vertices placed on the major axes (see figure 2a). The vertices are next split, to create 12 additional triangles, while simultaneously shrinking and rotating the initial octahedron faces (figure 2b). As the vertices are split further, we reach a point where all the triangles are equilateral and the 3d shape is an icosahedron (figure 2c). We will refer to the triangles formed from the initial octahedron faces as *base triangles* and the triangles created during the vertex splitting process as *side triangles*.

All icosahedron vertices are in one of the following three forms:

- group A: $(\pm 1.0, \pm \Phi, 0.0)$
- group B: $(0.0, \pm 1.0, \pm \Phi)$
- group C: $(\pm \Phi, 0.0, \pm 1.0)$

where $\Phi$ is the reciprocal of the golden ratio $\phi$. The vertices formed this way are not on the surface of a unit sphere, but all have the same length $\sqrt{1+\Phi^2}$.

Given three distinct groups of vertices, with 4 members each we can index them using the sign bits of their coordinates and the group they belong to. Vertices from group A are indexed 0-3, vertices

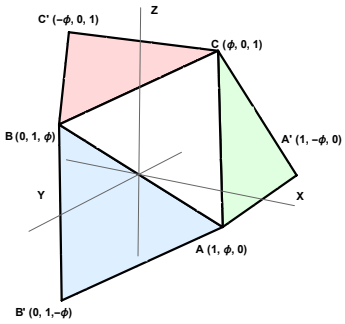



**Figure 3:** *Icosahedron triangles intersecting a single octant of the coordinate system. The triangle that contains a given direction vector can be determined by choosing one vertex from each one of the pairs $(A,C')$, $(B,A')$, $(C,B')$ based on the relation between direction and the plane passing through the opposite edge and $(0,0,0)$*

from groups B and C, 4-7 and 8-11 respectively. Two sign bits of the two non-zero coordinates form an index relative to the start of the group, with the first sign being the low order bit. For instance, a vertex with coordinates $(-\Phi,0,1)$ has the index 9 (group C, sign bits 1 and 0). For base triangles each vertex comes from a different group, for side triangles, two vertices come from the same group (with the sign on one of the axis flipped).

To get the vertices making up the spherical triangle that intersects the given direction, we extract the sign bits of the direction coordinates and apply them to the three patterns of icosahedron vertex groups, in the same way we specified indexing. This way we get the coordinates of the 3 vertices (and their indices in the icosahedron) making up the base triangle that is entirely contained within the octant of the coordinate system pointed to by the direction. Given the octant, the direction can either be pointing to the base triangle or one of the three side triangles. Each of the side triangles shares two vertices with the base triangle and the remaining vertex can be obtained by changing the group of the third vertex of the base triangle according to the following rules: $A \rightarrow C$, $B \rightarrow A$, $C \rightarrow B$ and flipping the sign of the coordinate with value $\Phi$. To check which triangle should be used one needs to simply check what side of the planes spanned by the pairs of base triangle vertices the direction is. This check can be performed in the $(+,+,+)$ octant, to benefit from computing dot products with fixed vectors. For this, the absolute value of direction coordinates need to be computed, but on recent GPUs this is a free operation.

Indexing triangles is also straightforward. We index the base triangles with the three sign bits of the octant they are placed in (for instance base triangle from $(+x,-y,-z)$ octant has an index 6). The remaining 12 side triangles are classified based on the coordinate system plane they intersect. Side triangles intersecting plane $YZ$ are marked red on the figures, triangles intersecting $XZ$ plane are green and ones intersecting $XY$ plane are blue - we will refer to those groups of triangles by the color with which they are marked. Triangles in the *red* group use indices 8-11, triangles from the *green* and *blue* groups use indices 12-15 and 16-19 respectively. There are 4 side triangles intersecting each plane, each one belonging to a different quadrant of that plane, the sign bits of the coordinates in that plane are used to index within the group. Triangles from group *red* use signs of the $(y,z)$ coordinates of the quadrant for indexing and groups *green* and *blue* use $(x,z)$ and $(x,y)$ respectively. Flipping across edge $(B,C)$ (choosing between vertex $C'$ instead of $A$) picks the triangle from *red* group, flipping across edge $(A,C)$ picks triangle from *green* group and across edge $(A,B)$ one from *blue* group.

The pseudocode in listing 1 illustrates the indexing process.

```
kT          = 0.618034f;
kT2         = kT * kT;

octantSign  = { dir.x < 0.0f ? -1.0f : 1.0f,
                dir.y < 0.0f ? -1.0f : 1.0f,
                dir.z < 0.0f ? -1.0f : 1.0f };

octantBit   = { dir.x < 0.0f ? 1 : 0,
                dir.y < 0.0f ? 1 : 0,
                dir.z < 0.0f ? 1 : 0 };

octantBitFlipped = 1 - octantBit;

// vertex coordinates
vertA = { 1.0f, kT, 0.0f } * octantSign;
vertB = { 0.0f, 1.0f, kT } * octantSign;
vertC = { kT, 0.0f, 1.0f } * octantSign;

vertAflipped = { -kT, 0.0f, 1.0f } * octantSign;
vertBflipped = { 1.0f, -kT, 0.0f } * octantSign;
vertCflipped = { 0.0f, 1.0f, -kT } * octantSign;

// vertex indices
indexA = octantBit.y * 2 + octantBit.x + 0;
indexB = octantBit.z * 2 + octantBit.y + 4;
indexC = octantBit.z * 2 + octantBit.x + 8;

indexAflipped = octantBit.z * 2 + octantBitFlipped.x + 8;
indexBflipped = octantBitFlipped.y * 2 + octantBit.x + 0;
indexCflipped = octantBitFlipped.z * 2 + octantBit.y + 4;

// triangle indices
t = octantBit.x + octantBit.y * 2 + octantBit.z * 4;
tRed    =  8 + octantBit.y + octantBit.z * 2;
tGreen  = 12 + octantBit.x + octantBit.z * 2;
tBlue   = 16 + octantBit.x + octantBit.y * 2;

// selection
vertAselect = dot(abs(dir), { 1.0f, kT2, -kT }) > 0.0f;
vertBselect = dot(abs(dir), { -kT, 1.0f, kT2 }) > 0.0f;
vertCselect = dot(abs(dir), { kT2, -kT, 1.0f }) > 0.0f;

v0 = vertAselect ? vertA : vertAflipped;
v1 = vertBselect ? vertB : vertBflipped;
v2 = vertCselect ? vertC : vertCflipped;

i0 = vertAselect ? indexA : indexAflipped;
i1 = vertBselect ? indexB : indexBflipped;
i2 = vertCselect ? indexC : indexCflipped;

t = vertAselect ? t : tRed;
t = vertBselect ? t : tGreen;
t = vertCselect ? t : tBlue
```

**Listing 1:** *Pseudocode illustrating indexing the icosahedron*

### 2.3. Evaluating function over the sphere

Given the indexing scheme, we specify how values stored at the vertices are used to compute the function value over the sphere.

This is in fact a scattered data interpolation problem, that has

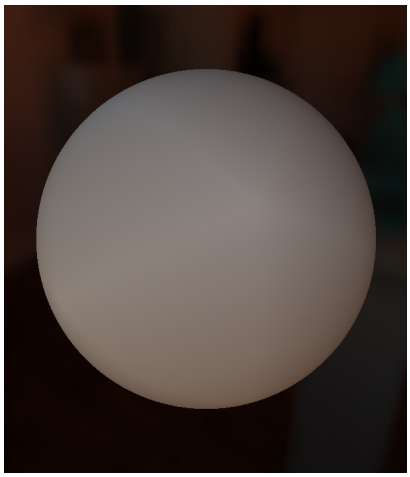


**Figure 4:** *Mach bands visibile when linear interpolation is used between icosahedron vertices*

been extensively studied for applications in different fields, most notably in geophysics [FS].

While the most efficient solution would be to use linear interpolation across the spherical triangles to compute the values at any point on the unit sphere, the quality of such solution is insufficient for our purposes. For instance, when used for encoding diffuse irradiance signal, the rapid change of the derivative across the edges of the triangulation creates Mach bands visible on smooth surfaces (see figure 4)

We propose the following alternatives as solutions, with varying performance characteristics:

- Hybrid cubic Bézier patch for individual color channels. In this version we store color (an RGB triple) and its derivatives in two perpendicular directions, tangent to sphere surface (two RGB triples) for each vertex of the icosahedron (27 values needed for reconstruction of the RGB signal).
- Hybrid cubic Bézier patch for luminosity component combined with linear interpolation for chromaticity components. In this version we store color (a YCoCg triple) and the derivatives of the luminosity component (two scalar values) for each vertex of the icosahedron (15 values needed for the reconstruction of the RGB signal).
- SRBFs (mixture of $\cos^2$ and $\cos^4$ lobes with hemispherical support) oriented along the directions defined by icosahedron vertices. In this version we store a color (an RGB triple) for each vertex of the icosahedron (18 values needed for the reconstruction of the RGB signal).

### 2.3.1. Hybrid cubic Bézier patch for individual color channels

We chose to use spherical hybrid cubic Bézier patch [ANS96b] to perform interpolation between the vertices. The hybrid spherical cubic patch is a convex combination of three spherical cubic Bézier patches with different center coefficients ($c_{111}$ coefficient - see figure 5 for configuration of control points of a cubic Bézier patch).

Each of the three base patches chooses the center coefficient to

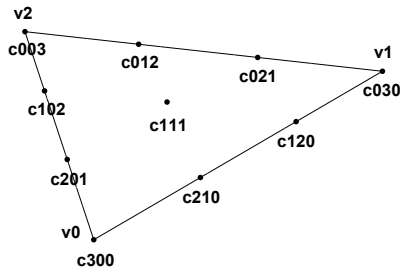



**Figure 5:** *Configuration of control points of a cubic Bézier patch*

ensure C1 continuity of the reconstruction across one of the triangle edges. The patches are blended in such a way that, near each edge, only the component that provided the continuity across that edge is blended in. There are other ways to compute the center coefficient [LS96], we use the method described for the planar case [GS91, LS96]. It forces the cross derivative of the reconstructed signal to change in linear fashion along the edges. The results obtained this way are not as smooth as if the cross derivative changed quadratically, but this approach does not need to store any additional data (and the difference in smoothness was not noticeable in our tests).

Since all three of the base patches differ only by one coefficient, the evaluation code first determines the value of that one coefficients and then proceeds with a single evaluation of a Bézier patch, instead of doing three evaluations and combining the results.

We evaluate the Bézier patch using spherical barycentric coordinates [ANS96a]. All spherical triangles defined by the icosahedron are identical, so computations of spherical barycentric coordinates of a given direction simplify to computing three dot products.

Given the barycentrics coordinates of a lookup point, we construct the coefficients of the Bézier patch based on the values stored on the icosahedron vertices.

The coefficients for the Bézier patch are constructed as follows:

- $c_{300}$, $c_{030}$ and $c_{003}$ coefficients are simply values of the function at the vertices, $f(v_i)$.
- $c_{210}$, $c_{120}$, $c_{201}$, $c_{102}$, $c_{021}$ and $c_{012}$ can be computed using signal values and its directional derivatives at the vertices. Computing the derivatives of the Bézier patch w.r.t. barycentric coordinates and using the chain rule to compute the directional derivative along the triangle edges, the edge coefficients are of the form : $c_{xyz} = \frac{1}{\alpha}\left(-\beta f(v_i) + \frac{1}{3}F_{ij}(v_i)\right)$, where $F_{ij}(v_i)$ is a directional derivative of the function at vertex $v_i$ along the edge $v_i \rightarrow v_j$ (which is computed by projecting the derivative stored on the vertices onto the edge $ij$) and $(\beta, \alpha, 0)$ is a unit vector tangent to the sphere, along the edge $v_i \rightarrow v_j$, expressed in spherical barycentric coordinates of the triangle used for interpolation. Since all the triangles of the icosahedron are identical, $\alpha$ and $\beta$ are constant and equal to $\frac{1}{2}\sqrt{\frac{1}{2}(5+\sqrt{5})}$ and $-\frac{1}{2}\sqrt{\frac{1}{10}(5+\sqrt{5})}$ respectively.

- To ensure C1 continuity across the edges of the triangles, the cross derivative of the signal in the middle of the edge needs to be equal to the average of the cross derivatives of the signal on the ends of the edge. This condition makes the cross derivative change linearly along the edge as opposed to quadratically. Since the cross derivatives on the vertices are given, this guarantees that triangles on both sides of shared edge will have the same value of the cross derivative along the whole edge, giving C1 continuity.
  Computing the cross derivatives as described, one can see that to ensure the above constrains along the edge opposite vertex 0, the center coefficient needs to be equal to:

$$c^0_{111} = a_0 c_{030} + a_1 c_{021} + a_1 c_{012} + a_0 c_{003} + a_2 c_{120} + a_2 c_{102}$$

  where:

$$a_0 = \frac{\sqrt{5}-5}{40}$$
$$a_1 = \frac{11\sqrt{5}-15}{40}$$
$$a_2 = \frac{\sqrt{5}}{10}$$

  and for remaining two edges:

$$c^1_{111} = a_0 c_{003} + a_1 c_{102} + a_1 c_{201} + a_0 c_{300} + a_2 c_{012} + a_2 c_{210}$$
$$c^2_{111} = a_0 c_{300} + a_1 c_{210} + a_1 c_{120} + a_0 c_{030} + a_2 c_{201} + a_2 c_{021}$$

  The final coefficient is a convex combination of the above:

$$c_{111} = w_0 c^0_{111} + w_1 c^1_{111} + w_2 c^2_{111}$$

  where:

$$w_0 = \frac{b_1 b_2}{b_1 b_2 + b_0 b_2 + b_0 b_1}$$
$$w_1 = \frac{b_0 b_2}{b_1 b_2 + b_0 b_2 + b_0 b_1}$$
$$w_2 = \frac{b_0 b_1}{b_1 b_2 + b_0 b_2 + b_0 b_1}$$

  where $b_0$, $b_1$ and $b_2$ are the spherical barycentric coordinates of the evaluation point. The above formula has singularities when any two of the barycentric coordinates are equal to zero (at the vertices). In those cases a value of one should be used instead as the weight for the component associated with the vertex the value is evaluated at and zero for the two remaining weights.

  To construct the coefficients we need:

- Function values at the vertices, $f(v_i)$.
- Directional derivatives of the function at the vertices, in two orthogonal directions, tangent to the surface of the unit sphere. Computing the derivatives along the edges, needed for the construction of the Bézier patch coefficients is simply projecting the gradient onto an axis along the edge, tangent to the surface of the unit sphere.

Cubic Berenstein polynomials in spherical barycentric coordinates do not have the property of *constant reproduction* which means that constant functions cannot be accurately represented. Since constant signals are not encountered often when dealing with precomputed lighting, this was not a problem in our tests (it also means that the change of the cross derivatives on the shared triangle edges is not actually purely linear, only close to being linear - but since it is the same on both sides of the edge, it does not pose any issues). If constant reproduction is required, regular barycentric interpolation on the planar triangles of the icosahedron can be performed. This poses the problem of transfoming the gradient information from a space tangent to the unit sphere to the surface of the triangles. Our experiments showed that the most visually pleasing way of doing this is simply using tangent space gradients as if they were defined in the plane of the triangle used for interpolation. Due to the angle deficit at every vertex, we can no longer ensure that the cross derivatives on the neighboring triangles is identical across the edge, but the loss of quality created is acceptable in many cases. Another alternative would be to compute the edge coefficients using spherical barycentric weights and use them for linear interpolation - but doing that not only loses the constant reproduction property but also lacks the quality, even though the cross derivatives match.

For evaluating full RGB color in any direction, one needs 27 coefficients - which is the same as for the 3rd order spherical harmonics. For the quality comparison between the two, see the Results section.

#### 2.3.2. Hybrid cubic Bézier patch for luminance with linear interpolation for chrominance

For some applications, the cost of the above solution might be too high. To reduce both bandwidth and ALU cost we propose the following alternative: the input RGB signal is first converted to YCoCg space. As the human visual system is more sensitive to changes in brightness than it is to changes in color, we store the Y component (luminance) together with its directional derivatives and reconstruct using the above scheme, while the Co and Cg components (chrominance) are stored without derivatives and interpolated linearly. This can create slight discoloration artifacts, but due to the local character of the interpolation, they have very limited angular extents and are barely noticable. This way one needs to only store 5 coefficients for each icosahedron vertex. It is worth noting that similar trick does not work when working with spherical harmonics - storing Y and Co/Cg as different order SH tends to create fairly visible artifacts covering large portions of spherical domain.

#### 2.3.3. SRBF oriented along the directions formed by icosahedron vertices

For certain applications, even the simplified scheme might be too costly in terms of ALU. We propose yet another variation of the basis that trades some extra bandwidth for inexpensive reconstruction. One can notice that a set of $\cos^2$ lobes clamped to a hemisphere, scaled by 0.5, oriented along the directions formed by icosahedron vertices form a partition of unity over the unit sphere. This is a surprising property, as in many domains RBFs cannot form a partition of unity.

Even more interestingly, the clamped $\cos^4$ lobes scaled by $\frac{5}{6}$ have the same property. As a result, any linear combination of the $\cos^2$ and $\cos^4$ lobes will form partition of unity over the unit sphere. After a number of experiments, we determined that combining the lobes with weights of 0.7 for the $\cos^2$ lobe and 0.3 for the $\cos^4$ lobe gives the best reconstruction of the irradiance over a wide variety of input signals, both visually as well as in terms of the mean square error.

Reconstruction finds the lobes that are nonzero for a given direction. Each of the $A$, $B$, $C$ groups consists of a pair of antipodal points, the sign of the 6 dot products determines the relative index, and evaluating the basis function requires the dot products squared and squared again.

## 3. Projection

Given an illumination signal $l(s)$ on the sphere, approximation with a function space $f(s) = \sum_i c_i f_i(s)$ can be posed as a least squares problem, minimizing the error function:

$$\int_\Omega (f(s) - l(s))^2 ds \tag{1}$$

The coefficients $c$ that minimize the above expression are computed by differentiating and solving for zero. The partial with respect to $c_k$ is:

$$2\int_\Omega (f(s) - l(s))\, f_k(s) ds \tag{2}$$

Exploiting the linearity of integration, this equals:

$$2\left(\sum_i c_i \int_\Omega f_i(s) f_k(s) ds - \int_\Omega l(s) f_k(s) ds\right) \tag{3}$$

Each partial corresponds to a row of the linear equation $\mathbf{G}c = b$. The matrix $\mathbf{G}c$ is referred to as the Gramm matrix, where each entry $\mathbf{G}_{ij}$ is simply the inner product of a pairs of basis functions $\int f_i(s) f_j(s)$ and the right hand side are the raw moments of the lighting environment $\int l(s) f_k(s)$. The inverse of the Gramm matrix $\mathbf{G}^{-1}$ has the coefficients for the duals $\tilde{f}_i$ of the basis functions. These are functions that have the property that $\int f_i(s)\tilde{f}_j(s) ds = \delta_{ij}$ and are of the form: $\tilde{f}_i = \sum_j \mathbf{G}^{-1}_{ij} f_j(s)$. If the function space is orthogonal, the Gramm matrix is an identity and the duals are just the primal functions.

For our various basis functions, we generated the Gramm matrices by evaluating with coefficient vectors that were zero for all degrees of freedom but one, and integrated the products over the sphere using numerical integration. Least squares projection is done by computing the moments from the lighting signal and multiplying by the Gramm matrix. If the lighting exists in another basis $g(s) = \sum_j g_j g_j(s)$, you can precompute operators that project it into a specific basis, in that case you have the error function:

$$\int_\Omega (f(s) - g(s))^2 ds \tag{4}$$

Differentiating this with respect to $c_k$ you get:

$$2\int_\Omega (f(s) - g(s))\, f_k(s) ds, \tag{5}$$

which simplifies to:

$$2\left(\sum_i c_i \int_\Omega f_i(s) f_k(s) ds - \sum_j g_j \int_\Omega g_j(s) f_k(s) ds\right) \tag{6}$$

This results in a linear system $\mathbf{G}c = \mathbf{B}g$, where $\mathbf{B}$ is a matrix that generates the raw moments of a signal in $g$ for the basis $f$ and $\mathbf{B}_{ij} = \int f_i(s) g_j(s)$. Moving from spherical harmonics, ambient cubes, spherical gaussians with fixed parameters or any other linear basis is easy to do.

### 3.1. Other operators

There are some other linear operators that can be constructed. For example if you have raw radiance moments, and want to generate irradiance, this can be achieved by projecting the radiance signal into spherical harmonics, doing the convolution and projecting back. Mathematically the operator is:

$$\mathbf{G}^{-1}\mathbf{B}_{sh}\mathbf{C}_{diff}\mathbf{B}^T_{sh}, \tag{7}$$

where $\mathbf{C}$ is a diagonal matrix that convolves with the normalized cosine kernel. In practice we use a high order SH expansion to generate this operator.

You can also window in any basis by factoring the above matrix:

$$\left(\mathbf{G}^{-1}\mathbf{B}_{sh}\right)\mathbf{C}_{diff}\mathbf{B}^T_{sh} \tag{8}$$

The term in the parentheses can be precomputed, allowing the convolution matrix to be efficiently changed. For very fine functions this is somewhat impractical, since a very high order SH expansion would have to be used, but for deringing least squares projection of irradiance signals this turns out to be useful.

#### 3.1.1. Relationship with Spherical Harmonics

One question that can be asked is how well can a given spherical basis reproduce certain classes of functions? If $\mathbf{B}_{sh}$ is the matrix that generates raw moments from spherical harmonics, you can project SH into the $f$ basis and then back. Analyzing the structure of these operators tells you how well the given basis reproduces the various bands of spherical harmonics. The operator defined below projects lighting coefficients in the $f$ basis into the orthogonal spherical harmonic basis[†], the closer the block sub-matrix for a given order is

[†] If the rhs basis is not orthonormal, you would project into it so you would have $\mathbf{G}^{-1}_{bf}\mathbf{B}^T_{bf}\mathbf{G}^{-1}\mathbf{B}_{bf}$.

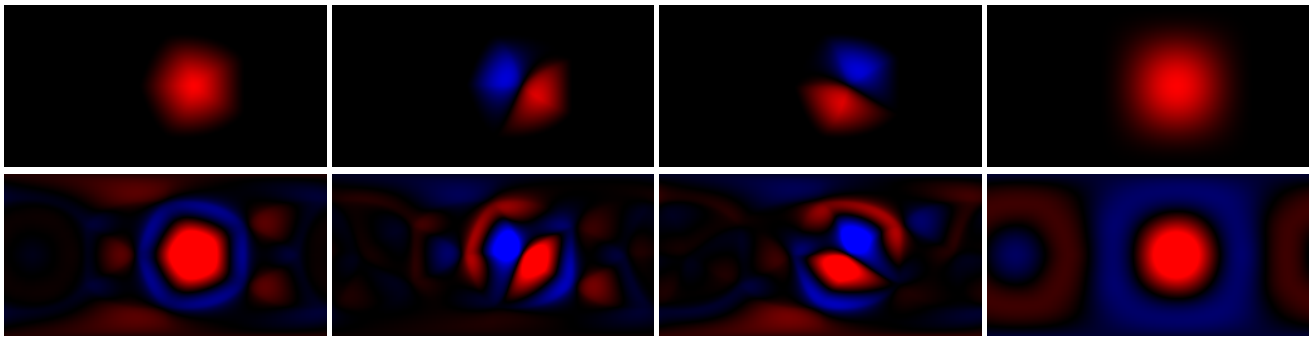


**Figure 6:** *Primal (top row) and Dual (bottom row) basis function images. Bezier vertex value, u derivative, v derivative,* $\cos^2 + \cos^4$

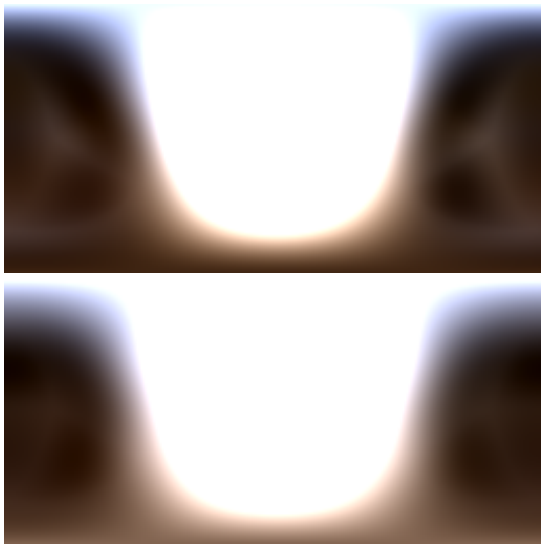


**Figure 7:** *AD using spherical Bézier patch before and after windowing.*

to an identity, the better the corresponding basis functions can be approximated.

$$\mathbf{B}_{sh}^T \mathbf{G}^{-1} \mathbf{B}_{sh} \tag{9}$$

If there are some SH that are in the null-space of the basis, you would have a zero row. Below is a table of the Frobenius norm of $\mathbf{I} - \mathbf{B}_{sh}^T \mathbf{G}^{-1} \mathbf{B}_{sh}$ for various basis functions:

| Basis | SH2 | SH3 | SH4 | SH5 |
|---|---|---|---|---|
| Ambient Cube | 0.10825 | 1.73543 | 3.16228 | 4.3589 |
| AD Linear | 3.543e-3 | 0.18454 | 2.01751 | 4.13203 |
| AD $\cos^2$ $\cos^4$ | 3.971e-4 | 1.427e-2 | 2.0001 | 4.12316 |
| AD $\cos^2$ | 2.229e-3 | 3.153e-3 | 2.0001 | 4.12316 |
| AD Flat | 2.806e-4 | 3.103e-3 | 2.804e-2 | 0.19754 |
| AD Spherical | 3.857e-4 | 1.539e-3 | 2.777e-2 | 0.17101 |

The ambient cube basis poorly reproduces the quadratic spherical harmonics (SH3), where 3 of the 5 quadratic basis functions are in the null-space of the basis. The Bézier patch variant of the ambient dice basis is fairly accurate through the cubics, and the error in the quartics is low, and shoots up at the quintic polynomials. The SRBF version reproduces constants, and if it was just $\cos^2$ also reproduces the quadratic band. In practice, the blending of the basis functions improves the approximation of the linear band, which lowers overall reconstruction error on the lighting environments we have tested.

## 4. Results

When comparing images *any* numerical metric can be misleading and cannot be fully relied on. To provide some quantitative values for different spherical bases, we decided to simply use root mean square error, averaged over the three color channels. To give a better intuition for *how* the error changes over the domain we also provide visualizations of the error itself.

Figure 9 shows different irradiance signals projected onto different bases. The first column shows an example with a large gradient in the signal. All representations exhibit some form of ringing (more global in case of SH and more localized for AD), this can be reduced as shown in Figure 7. The other columns show lighting that is more typical of indoor lighting.

Table 8 compares the root mean square error for the same examples. Ambient Dice basis, even in the cheapest variants - with SRBF reconstruction or when only Y component is evaluated with full precision, is numerically comparable to 3rd order SH (and the second variant oftentimes to even 5th order SH), despite requiring fewer coefficients for evaluation. It is worth repeating that numerical comparison of the error values is not really the best way of evaluating the results, as for instance the flat version of Ambient Dice shows lower error, but in certain situations the tessellation artifacts generated by unmatched cross derivatives between neighboring triangles can become noticeable.

Figure 11 shows the projection of a very peaked function into the basis, and that function convolved with a cosine. The parametrization used has only vertical distortion, and the shape should be invariant when projected for two different directions. Linear interpolation on the Ambient Dice shows a fairly pronounced "wobble" of the projection, and mach bands are evident in the diffuse convolved

| Basis | RMSE *Ennis* | RMSE *Wells* | RMSE *Hallstatt* | Coeffs total | Coeffs eval |
|---|---|---|---|---|---|
| 2nd order SH | 0.556209 | 0.0550163 | 0.0383354 | 12 | 12 |
| 3rd order SH | 0.0950065 | 0.00753032 | 0.00757991 | 27 | 27 |
| 5th order SH | 0.0310738 | 0.00426008 | 0.00564306 | 75 | 75 |
| AD $\cos^2$ $\cos^4$ | 0.0914495 | 0.00641993 | 0.00606067 | 36 | 18 |
| AD YCoCg spherical | 0.0408273 | 0.00657454 | 0.0080691 | 60 | 15 |
| AD RGB flat | 0.0374315 | 0.00343113 | 0.00287001 | 108 | 27 |
| AD RGB spherical | 0.0392434 | 0.00505698 | 0.00480793 | 108 | 27 |

**Figure 8:** *Comparison of the root mean squared error of projections of various lighting environment, total number of coefficients to encode and number of coefficients required for point evaluation of the RGB signal for different bases*

**Figure 12:** *Scene from a commercial game engine used for perfromance tests. Indirect lighting of the objects marked in green was computed using the described methods to provide the timing measurements.*

result when not using spherical barycentric coordinates. There is ringing in the irradiance function, but the magnitude of the ring is at most 3% of the peak, so they are not visible in these images.

We ran the performance tests in a commercial game engine. The test was running on Playstation 4 console, at 1920x1080 resolution. Certain classes of objects are using local irradiance volumes (stored in textures) as a source of indirect illumination. For those objects, we tested encoding the lighting using the describes bases (the full RGB Bézier patch was not included in the test, as after the initial quality tests we decided that for such a constrained environment we are willing to accept the reduced quality and utilize the memory and performance savings elsewhere). We timed the rendering of the main pass only, so the results do not include any of the shadowmap, depth only rendering or image post-processing times. They do however include times to render objects that use other techniques to compute the indirect lighting (most notably lightmaps).

| Variant used | Time |
|---|---|
| Constant ambient | 5.08 ms |
| SH2 | 5.17 ms |
| SH3 | 5.31 ms |
| AD $\cos^2$ $\cos^4$ | 5.26 ms |
| AD YCoCg Spherical | 5.36 ms |

## 5. Limitations

The bases were designed for efficient evaluation in a given direction such that only a small subset of all the coefficients that describe the spherical signal are needed. In some applications, operations other than a simple lookup might be needed - such as computing a convolution with an arbitrary kernel. Those operations will require access to more coefficients, possibly even all of them. In such cases, other bases might be more performant or bandwidth efficient.

The basis is not rotationally invariant, but is reasonable at steering a tight light source.

## 6. Conclusions and Future work

We have investigated a family of basis functions that have attractive trade offs between performance and storage and are competitive in quality to spherical harmonics. There are several basis functions that did not pan out that are worth mentioning: Using linear or Bezier interpolation over octahedrons and generalized barycentric coordinates [LD89] over the pentagons on a dodecahedron as well as using various other locally supported SRBFs.

We also think that it should be possible to find a similarly simple indexing scheme that would allow to index a subdivided icosahedron. Such a scheme could be used to construct a more detailed version of the signal, similar to a higher MIP level in a texture. This way a whole chain of lighting environments could be stored and accessed, possibly storing signals convolved with a different sized kernels. This is however an area of a future research.

## 7. Acknowledgements

We wanted to thank Derek Nowrouzezahrai, Josiah Manson and Bart Wroński for feedback. We would like to thank Yuriy O'Donnell and David Neubelt for both discussions and releasing the open source Probulator project [ON16] - we intend to provide the implementation of the methods described in this paper within that framework in the near future. We also wanted to thank Infinity Ward for allowing us to use their engine in our tests and Jennifer Velazquez for helping with logistical issues.

## References

[ANS96a] ALFELD P., NEAMTU M., SCHUMAKER L. L.: Bernstein-bézier polynomials on spheres and sphere-like surfaces. *Computer Aided Geometric Design 13*, 4 (1996), 333–349. 4

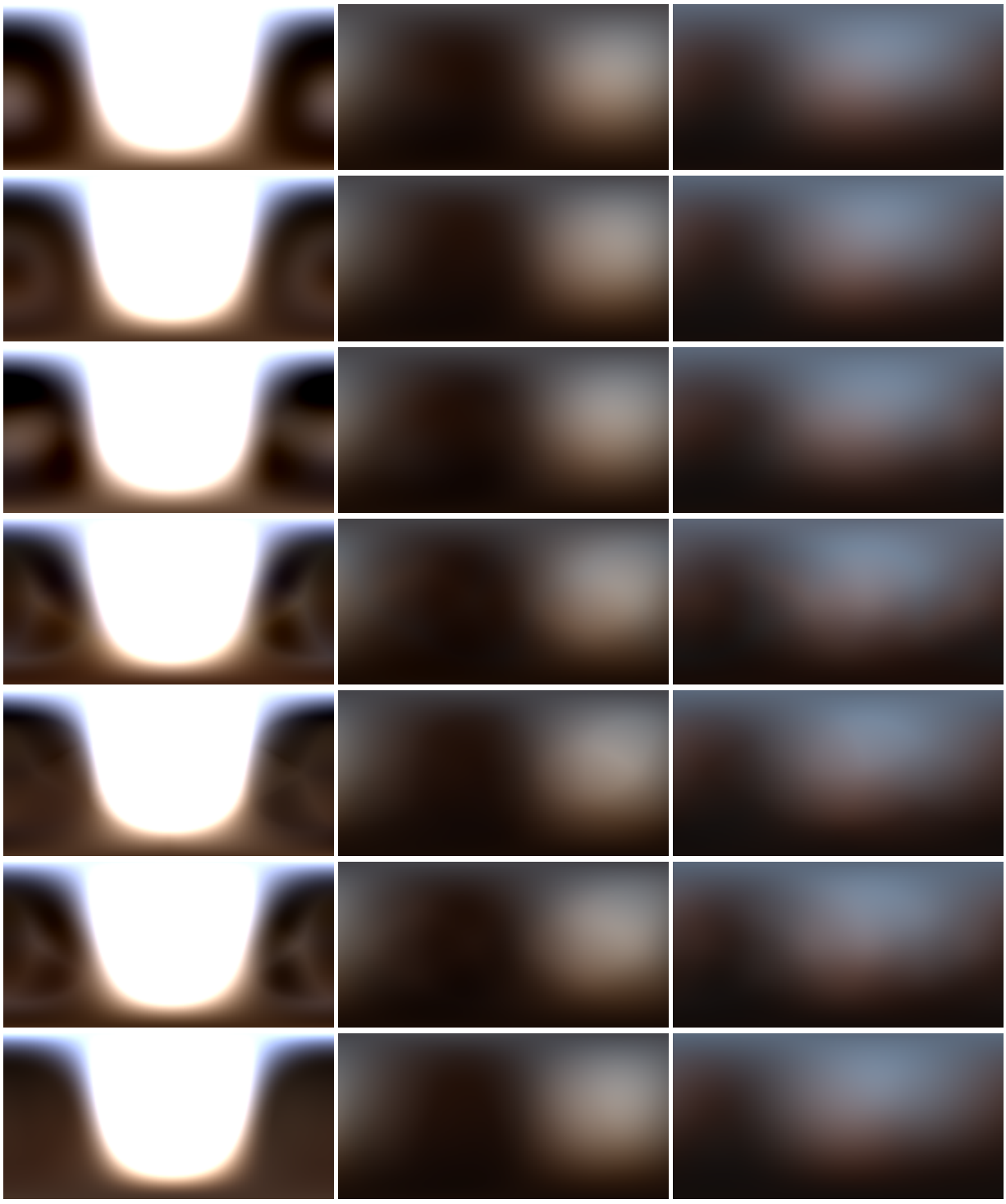


**Figure 9:** *Irradiance environment maps generated for various lighting environments projected to different bases, top to bottom: 3rd order SH, 5th order SH, Ambient Dice with mixture of* $\cos^2$ *and* $\cos^4$ *lobes, Ambient Dice with cubic Y and linear Co/Cg, interpolated with spherical Bezier patches, Ambient Dice with cubic RGB interpolated with flat Bezier patches, Ambient Dice with cubic RGB interpolated on spherical Bezier patches, importance sampled reference. No de-rining or gradient suppression was performed, all examples are just least square projection of the signal into the basis.*

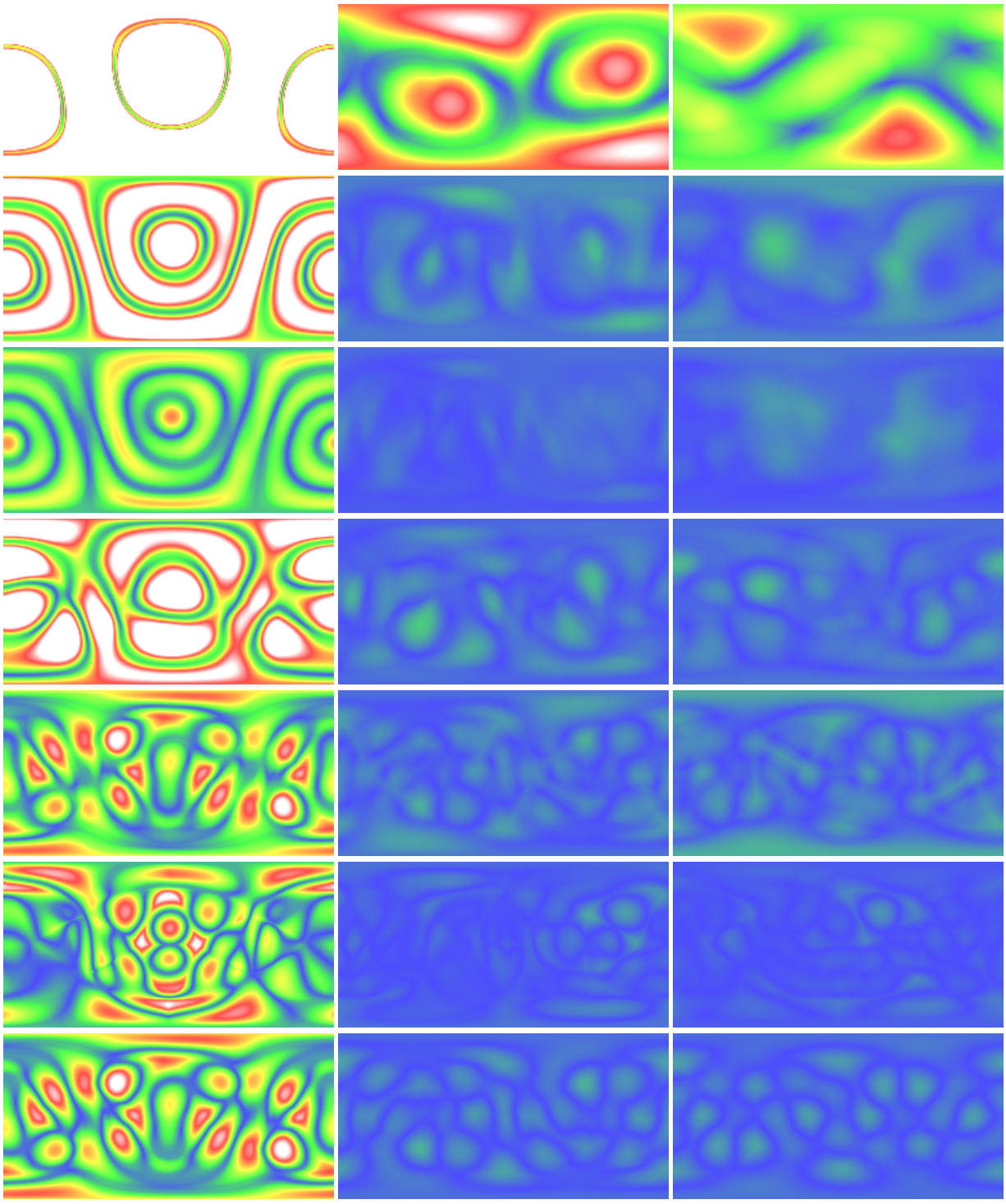


**Figure 10:** *Visualization of the error of the reconstruction of the irradinace environment maps using different bases. top to bottom: 2nd order SH, 3rd order SH, 5th order SH, Ambient Dice with mixture of* $\cos^2$ *and* $\cos^4$ *lobes, Ambient Dice with cubic Y and linear Co/Cg, interpolated with spherical Bezier patches, Ambient Dice with cubic RGB interpolated with flat Bezier patches, Ambient Dice with cubic RGB interpolated on spherical Bezier patches*

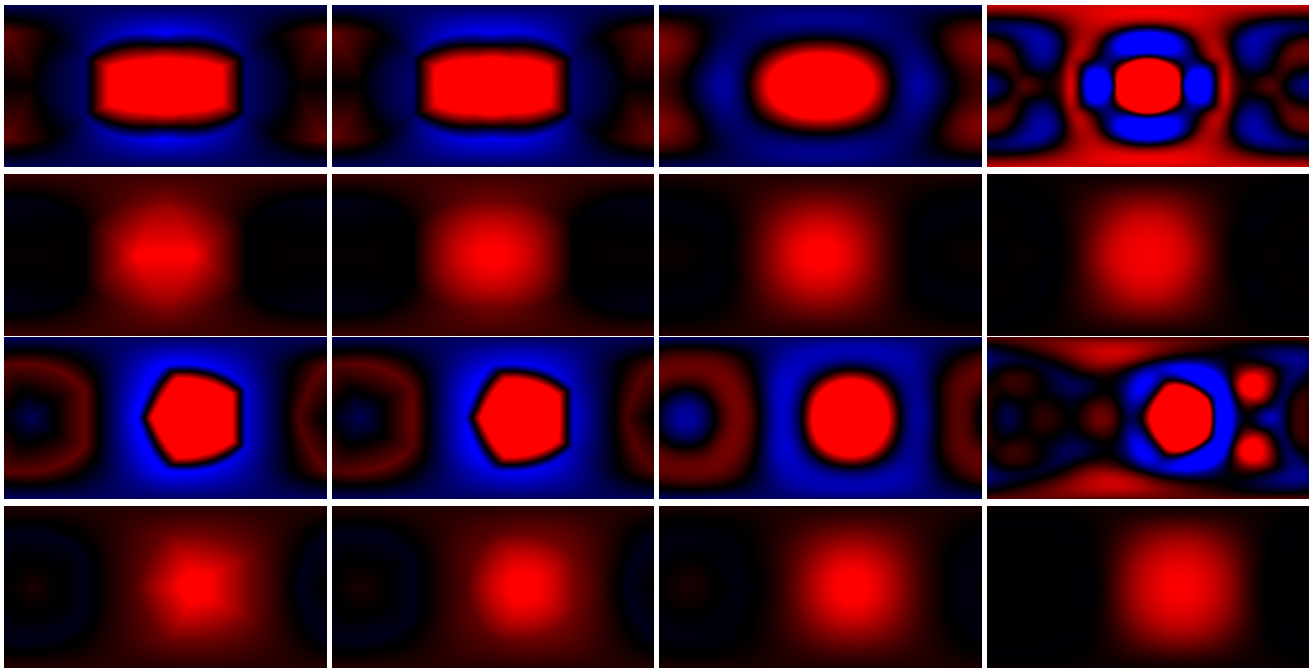


**Figure 11:** *These images show the 11th order spherical harmonic evaluation of a delta function projected into the basis at two different locations, red is positive and blue is negative. The columns are the basis functions: Linear, Spherical Linear,* $\cos^2 \cos^4$ *and Spherical Cubic. The rows are radiance for light position 0, irradiance for light position 0, radiance for light position 1, irradiance for light position 1.*


[ANS96b] ALFELD P., NEAMTU M., SCHUMAKER L. L.: Fitting scattered data on sphere-like surfaces using spherical splines. *Journal of Computational and Applied Mathematics 73*, 1-2 (1996), 5–43. 4

[Ban96] BANCHOFF T.: *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions*. Scientific American Library series. Scientific American Library, 1996. 2

[Cup12] CUPISZ R.: Lightprobe interpolation using tetrahedral tessellations. In *Game Developers Conference* (2012). 1

[FS] FASSHAUER G. E., SCHUMAKER L. L.: Scattered data fitting on the sphere. In *Proceedings of the International Conference on Mathematical Methods for Curves and Surfaces, 1997*. 4

[GS91] GOODMAN T., SAID H.: A c1 triangular interpolant suitable for scattered data interpolation. *International Journal for Numerical Methods in Biomedical Engineering 7*, 6 (1991), 479–485. 4

[GSHG98] GREGER G., SHIRLEY P., HUBBARD P. M., GREENBERG D. P.: The irradiance volume. *IEEE Comput. Graph. Appl. 18*, 2 (Mar. 1998), 32–43. 1

[LD89] LOOP C. T., DEROSE T. D.: A multisided generalization of bÉzier surfaces. *ACM Trans. Graph. 8*, 3 (July 1989), 204–234. 8

[LS96] LIU X., SCHUMAKER L. L.: Hybrid bézier patches on sphere-like surfaces. *Journal of computational and applied Mathematics 73*, 1-2 (1996), 157–172. 4

[McT04] MCTAGGART G.: Half-life 2 source shading. In *Game Developers Conference* (2004). 1

[NP15] NEUBELT D., PETTINEO M.: Advanced lighting r&d at ready at dawn studios. In *SIGGRAPH 2015 Course: Physically Based Shading in Theory and Practice* (2015). 2

[ON16] O'DONNELL Y., NEUBELT D.: Probulator. https://github.com/kayru/Probulator, 2016. 8

[RH01] RAMAMOORTHI R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *SIGGRAPH 2001 Conference Proceedings, August 12–17, 2001, Los Angeles, CA* (2001), ACM Press, pp. 497–500. 1

[Slo08] SLOAN P.-P.: Stupid spherical harmonics (sh) tricks. In *Game Developers Conference* (2008). 2

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph. 25*, 3 (July 2006), 967–976. 2