Visualising higher-dimensional space-time and space-scale objects as projections to \(\mathbb{R}^3\)

Ken Arroyo Ohori Corresp., [1] , **Hugo Ledoux** [1] , **Jantien Stoter** [1]

[1] 3D Geoinformation, Delft University of Technology, Delft, Netherlands

Corresponding Author: Ken Arroyo Ohori
Email address: g.a.k.arroyoohori@tudelft.nl

Objects of more than three dimensions can be used to model geographic phenomena that occur in space, time and scale. For instance, a single 4D object can be used to represent the changes in a 3D object's shape across time or all its optimal representations at various levels of detail. In this paper, we look at how such higher-dimensional space-time and space-scale objects can be visualised as projections from \(\mathbb{R}^4\) to \(\mathbb{R}^3\). We present three projections that we believe are particularly intuitive for this purpose: (i) a simple `long axis' projection that puts 3D objects side by side; (ii) the well-known orthographic and perspective projections; and (iii) a projection to a 3-sphere (\(S^3\)) followed by a stereographic projection to \(\mathbb{R}^3\), which results in an inwards-outwards fourth axis. Our focus is in using these projections from \(\mathbb{R}^4\) to \(\mathbb{R}^3\), but they are formulated from \(\mathbb{R}^n\) to \(\mathbb{R}^{n-1}\) so as to be easily extensible and to incorporate other non-spatial characteristics. We present a prototype interactive visualiser that applies these projections from 4D to 3D in real-time using the programmable pipeline and compute shaders of the Metal graphics API.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

Visualising higher-dimensional space-time and space-scale objects as projections to $\mathbb{R}^3$

Ken Arroyo Ohori[1], **Hugo Ledoux**[2], **and Jantien Stoter**[3]

[1–3]**3D Geoinformation, Delft University of Technology, Delft, Netherlands**

Corresponding author:
First Author[1]

Email address: g.a.k.arroyoohori@tudelft.nl

ABSTRACT

Objects of more than three dimensions can be used to model geographic phenomena that occur in space, time and scale. For instance, a single 4D object can be used to represent the changes in a 3D object's shape across time or all its optimal representations at various levels of detail. In this paper, we look at how such higher-dimensional space-time and space-scale objects can be visualised as projections from $\mathbb{R}^4$ to $\mathbb{R}^3$. We present three projections that we believe are particularly intuitive for this purpose: (i) a simple 'long axis' projection that puts 3D objects side by side; (ii) the well-known orthographic and perspective projections; and (iii) a projection to a 3-sphere (S^3) followed by a stereographic projection to $\mathbb{R}^3$, which results in an inwards-outwards fourth axis. Our focus is in using these projections from $\mathbb{R}^4$ to $\mathbb{R}^3$, but they are formulated from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$ so as to be easily extensible and to incorporate other non-spatial characteristics. We present a prototype interactive visualiser that applies these projections from 4D to 3D in real-time using the programmable pipeline and compute shaders of the Metal graphics API.

BACKGROUND

Projecting the 3D nature of the world down to two dimensions is one of the most common problems at the juncture of geographic information and computer graphics, whether as the map projections in both paper and digital maps (Snyder, 1987; Grafarend and You, 2014) or as part of an interactive visualisation of a 3D city model on a computer screen (Foley and Nielson, 1992; Shreiner et al., 2013). However, geographic information is not inherently limited to objects of three dimensions. Non-spatial characteristics such as time (Hägerstrand, 1970; Güting et al., 2000; Hornsby and Egenhofer, 2002; Kraak, 2003) and scale (Meijers, 2011a) are often conceived and modelled as additional dimensions, and objects of three or more dimensions can be used to model objects in 2D or 3D space that also have changing geometries along these non-spatial characteristics (van Oosterom and Stoter, 2010; Arroyo Ohori, 2016). For example, a single 4D object can be used to represent the changes in a 3D object's shape across time or all the best representations of a 3D object at various levels of detail (van Oosterom and Meijers, 2014; Arroyo Ohori et al., 2015a,c).

Objects of more than three dimensions can be however unintuitive (Noll, 1967; Frank, 2014), and visualising them is a challenge. While some operations on a higher-dimensional object can be achieved by running automated methods (e.g. certain validation tests or area/volume computations) or by visualising only a chosen 2D or 3D subset (e.g. some of its bounding faces or a cross-section), sometimes there is no substitute for being able to view a complete nD object—much like viewing floor or façade plans is often no substitute for interactively viewing the complete 3D model of a building. By viewing a complete model, one can see at once the 3D objects embedded in the model at every point in time or scale as well as the equivalences and topological relationships between their constituting elements. More directly, it also makes it possible to get an intuitive understanding of the complexity of a given 4D model.

For instance, in Fig. 1 we show an example of a 4D model representing a house at two different levels of detail and all the equivalences its composing elements. It forms a valid manifold 4-cell (Arroyo Ohori

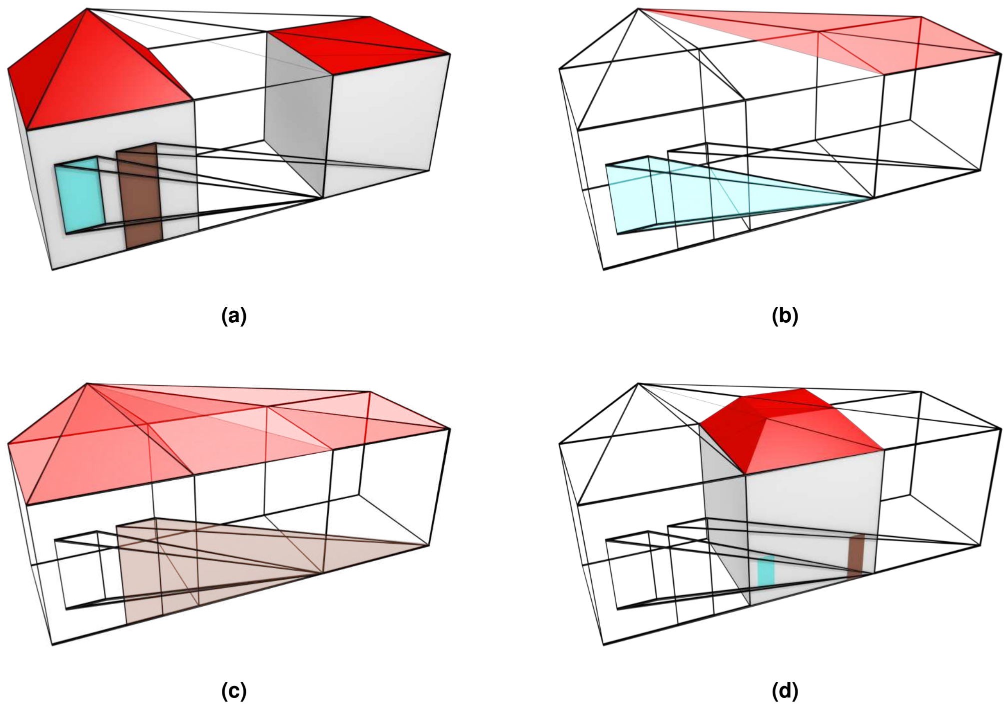


Figure 1. A 4D model of a house at two levels of detail and all the equivalences its composing elements is a polychoron bounded by: (a) volumes representing the house at the two levels of detail, (b) a pyramidal volume representing the window at the higher LOD collapsing to a vertex at the lower LOD, (c) a pyramidal volume representing the door at the higher LOD collapsing to a vertex at the lower LOD, and a roof volume bounded by (a) the roof faces of the two LODs, (b) the ridges at the lower LOD collapsing to the tip at the higher LOD and (c) the hips at the higher LOD collapsing to the vertex below them at the lower LOD. (d) A 3D cross-section of the model obtained at the middle point along the LOD axis.

et al., 2014), allowing it to be represented using data structures such as a 4D generalised or combinatorial map.

This paper thus looks at a key aspect that allows higher-dimensional objects to be visualised interactively, namely how to project higher-dimensional objects down to fewer dimensions. While there is previous research on the visualisation of higher-dimensional objects, we aim to do so in a manner that is reasonably intuitive, implementable and fast. We therefore discuss some relevant practical concerns, such as how to also display edges and vertices and how to use compute shaders to achieve good framerates in practice.

In order to do this, we first briefly review the most well-known transformations (translation, rotation and scale) and the cross-product in nD, which we use as fundamental operations in order to project objects and to move around the viewer in an nD scene. Afterwards, we show how to apply three different projections from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$ and argue why we believe they are intuitive enough for real-world use. These can be used to project objects from $\mathbb{R}^4$ to $\mathbb{R}^3$, and if necessary, they can be used iteratively in order to bring objects of any dimension down to 3D or 2D. We thus present: (i) a simple 'long axis' projection that stretches objects along one custom axis while preserving all other coordinates, resulting in 3D objects that are presented side by side; (ii) the orthographic and perspective projections, which are analogous to those used from 3D to 2D; and (iii) an inwards/outwards projection to an $(n-1)$-sphere followed by an stereographic projection to $\mathbb{R}^{n-1}$, which results in a new inwards-outwards axis.

We present a prototype that applies these projections from 4D to 3D and then applies a standard perspective projection down to 2D. We also show that with the help of low-level graphics APIs, all the required operations can be applied at interactive framerates for the 4D to 3D case. We finish with a discussion of the advantages and disadvantages of this approach.

Higher-dimensional modelling of space, time and scale

There are a great number of models of geographic information, but most consider space, time and scale separately. For instance, space can be modelled using primitive instancing (Foley et al., 1995; Kada, 2007), constructive solid geometry (Requicha and Voelcker, 1977) or various boundary representation approaches (Muller and Preparata, 1978; Guibas and Stolfi, 1985; Lienhardt, 1994), among others. Time can be modelled on the basis of snapshots (Armstrong, 1988; Hamre et al., 1997), space-time composites (Peucker and Chrisman, 1975; Chrisman, 1983), events (Worboys, 1992; Peuquet, 1994; Peuquet and Duan, 1995), or a combination of all of these (Abiteboul and Hull, 1987; Worboys et al., 1990; Worboys, 1994; Wachowicz and Healy, 1994). Scale is usually modelled based on independent datasets at each scale (Buttenfield and DeLotto, 1989; Friis-Christensen and Jensen, 2003; Meijers, 2011b), although approaches to combine them into single datasets (Gröger et al., 2012) or to create progressive and continuous representations also exist (Ballard, 1981; Jones and Abraham, 1986; Günther, 1988; van Oosterom, 1990; Filho et al., 1995; Rigaux and Scholl, 1995; Plümer and Gröger, 1997; van Oosterom, 2005).

As an alternative to the all these methods, it is possible to represent any number of parametrisable characteristics (e.g. two or three spatial dimensions, time and scale) as additional dimensions in a geometric sense, modelling them as orthogonal axes such that real-world 0D–3D entities are modelled as higher-dimensional objects embedded in higher-dimensional space. These objects can be consequently stored using higher-dimensional data structures and representation schemes Čomić and de Floriani (2012); Arroyo Ohori et al. (2015b). Possible approaches include incidence graphs Rossignac and O'Connor (1989); Masuda (1993); Sohanpanah (1989); Hansen and Christensen (1993), Nef polyhedra Bieri and Nef (1988), and ordered topological models Brisson (1993); Lienhardt (1994). This is consistent with the basic tenets of n-dimensional geometry (Descartes, 1637; Riemann, 1868) and topology (Poincaré, 1895), which means that it is possible to apply a wide variety of computational geometry and topology methods to these objects.

In a practical sense, 4D topological relationships between 4D objects provide insights that 3D topological relationships cannot (Arroyo Ohori et al., 2013). Also, McKenzie et al. (2001) contends that weather and groundwater phenomena cannot be adequately studied in less than four dimensions, and van Oosterom and Stoter (2010) argue that the integration of space, time and scale into a 5D model for GIS can be used to ease data maintenance and improve consistency, as algorithms could detect if the 5D representation of an object is self-consistent and does not conflict with other objects.

Basic transformations and the cross-product in nD

The basic transformations (translation, scale and rotation) have a straightforward definition in n dimensions, which can be used to move and zoom around a scene composed of nD objects. In addition, the n-dimensional cross-product can be used to obtain a new vector that is orthogonal to a set of other $n-1$ vectors in $\mathbb{R}^n$. We use these operations as a base for nD visualisation and are thus described briefly below.

The **translation** of a set of points in $\mathbb{R}^n$ can be easily expressed as a sum with a vector $t = [t_0, \dots, t_n]$, or alternatively as a multiplication with a matrix using homogeneous coordinates, which is defined as:

$$T = \begin{bmatrix} 1 & 0 & \cdots & 0 & t_0 \\ 0 & 1 & \cdots & 0 & t_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & t_n \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Scaling is similarly simple. Given a vector $s = [s_0, s_1, \dots, s_n]$ that defines a scale factor per axis (which in the simplest case can be the same for all axes), it is possible to define a matrix to scale an object as:

$$S = \begin{bmatrix} s_0 & 0 & \cdots & 0 \\ 0 & s_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_n \end{bmatrix}$$

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

Rotation is somewhat more complex. Rotations in 3D are often conceptualised intuitively as rotations *around* the x, y and z axes. However, this view of the matter is only valid in 3D. In higher dimensions, it is necessary to consider instead rotations *parallel to a given plane* (Hollasch, 1991), such that a point that is continuously rotated (without changing the rotation direction) will form a circle that is parallel to that plane. This view is valid in 2D (where there is only one such plane), in 3D (where a plane is orthogonal to the usually defined axis of rotation) and in any higher dimension. Incidentally, this shows that the degree of rotational freedom in nD is given by the number of possible combinations of two axes (which define a plane) on that dimension (Hanson, 1994), i.e. $\binom{n}{2}$.

Thus, in a 4D coordinate system defined by the axes x, y, z and w, it is possible to define six 4D rotation matrices, which correspond to the six rotational degrees of freedom in 4D (Hanson, 1994). These respectively rotate points in $\mathbb{R}^4$ parallel to the xy, xz, xw, yz, yw and zw planes:

$$R_{xy} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_{xz} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{xw} = \begin{bmatrix} \cos\theta & 0 & 0 & -\sin\theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\theta & 0 & 0 & \cos\theta \end{bmatrix} \qquad R_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{yw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & -\sin\theta \\ 0 & 0 & 1 & 0 \\ 0 & \sin\theta & 0 & \cos\theta \end{bmatrix} \qquad R_{zw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & \sin\theta & \cos\theta \end{bmatrix}$$

The n-dimensional **cross-product** is easy to understand by first considering the lower-dimensional cases. In 2D, it is possible to obtain a normal vector to a 1D line as defined by two (different) points p^0 and p^1, or equivalently a normal vector to a vector from p^0 to p^1. In 3D, it is possible to obtain a normal vector to a 2D plane as defined by three (non-collinear) points p^0, p^1 and p^2, or equivalently a normal vector to a pair of vectors from p^0 to p^1 and from p^0 to p^2. Similarly, in nD it is possible to obtain a normal vector to a $(n-1)$D subspace—probably easier to picture as an $(n-1)$-simplex—as defined by n linearly independent points $p^0, p^1, \ldots, p^{n-1}$, or equivalently a normal vector to a set of $n-1$ vectors from p^0 to every other point (i.e. $p^1, p^2, \ldots, p^{n-1}$) (Massey, 1983; Elduque, 2004).

Hanson (1994) follows the latter explanation using a set of $n-1$ vectors all starting from the first point to give an intuitive definition of the n-dimensional cross-product. Assuming that a point p^i in $\mathbb{R}^n$ is defined by a tuple of coordinates denoted as $(p^i_0, p^i_1, \ldots, p^i_{n-1})$ and a unit vector along the i-th dimension is denoted as $\hat{x}_i$, the n-dimensional cross-product $\vec{N}$ of a set of points $p^0, p^1, \ldots, p^{n-1}$ can be expressed compactly as *the cofactors of the last column* in the following determinant:

$$\vec{N} = \begin{vmatrix} (p^1_0 - p^0_0) & (p^2_0 - p^0_0) & \cdots & (p^{n-1}_0) & \hat{x}_0 \\ (p^1_1 - p^0_1) & (p^2_1 - p^0_1) & \cdots & (p^{n-1}_1) & \hat{x}_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (p^1_{n-1} - p^0_{n-1}) & (p^2_{n-1} - p^0_{n-1}) & \cdots & (p^{n-1}_{n-1}) & \hat{x}_{n-1} \end{vmatrix}$$

The components of the normal vector $\vec{N}$ are thus given by the minors of the unit vectors $\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_{n-1}$. This vector $\vec{N}$—like all other vectors—can be normalised into a unit vector by dividing it by its norm $\|\vec{N}\|$.

Previous work on the visualisation of higher-dimensional objects

There is a reasonably extensive body of work on the visualisation of 4D and nD objects, although it is still more often used for its creative possibilities (e.g. making nice-looking graphics) than for practical applications. In literature, visual metaphors of 4D space were already described in the 1880s in Flatland:

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

A Romance of Many Dimensions (Abbott, 1884) and A New Era of Thought (Hinton, 1888). Other books that treat the topic intuitively include Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions (Banchoff, 1996) and The Visual Guide To Extra Dimensions: Visualizing The Fourth Dimension, Higher-Dimensional Polytopes, And Curved Hypersurfaces (McMullen, 2008).

In a more concrete computer graphics context, already in the 1960s, Noll (1967) described a computer implementations of the 4D to 3D perspective projection and its application in art (Noll, 1968).

Beshers and Feiner (1988) describe a system that displays animating (i.e. continuously transformed) 4D objects that are rendered in real-time and use colour intensity to provide a visual cue for the 4D depth. It is extended to n dimensions by Feiner and Beshers (1990).

Banks (1992) describes a system that manipulates surfaces in 4D space. It describes interaction techniques and methods to deal with intersections, transparency and the silhouettes of every surface.

Hanson and Cross (1993) describes a high-speed method to render surfaces in 4D space with shading using a 4D light and occlusion, while Hanson (1994) describes much of the mathematics that are necessary for nD visualisation. A more practical implementation is described in Hanson et al. (1999).

Chu et al. (2009) describe a system to visualise 2-manifolds and 3-manifolds embedded in 4D space and illuminated by 4D light sources. Notably, it uses a custom rendering pipeline that projects tetrahedra in 4D to volumetric images in 3D—analogous to how triangles in 3D that are usually projected to 2D images.

A different possible approach lies in using meaningful 3D cross-sections of a 4D dataset. For instance, Kageyama (2016) describes how to visualise 4D objects as a set of hyperplane slices. Bhaniramka et al. (2000) describe how to compute isosurfaces in dimensions higher than three using an algorithm similar to marching cubes. D'Zmura et al. (2000) describe a system that displays 3D cross-sections of a 4D virtual world one at a time.

Similar to the methods described above, Hollasch (1991) gives a simple formulation to describe the 4D to 3D projections, which is itself based on the 3D to 2D orthographic and perspective projection methods described by Foley and Nielson (1992). This is the method that we extend to define n-dimensional versions of these projections and is thus explained in greater detail below. The mathematical notation is however changed slightly so as to have a cleaner extension to higher dimensions.

In order to apply the required transformations, Hollasch (1991) first defines a point $from \in \mathbb{R}^4$ where the viewer (or camera) is located, a point $to \in \mathbb{R}^4$ that the viewer directly points towards, and a set of two vectors $\overrightarrow{up}$ and $\overrightarrow{over}$. Based on these variables, he defines a set of four unit vectors $\hat{a}$, $\hat{b}$, $\hat{c}$ and $\hat{d}$ that define the axes of a 4D coordinate system centred at the $from$ point. These are ensured to be orthogonal by using the 4D cross-product to compute them, such that:

$$
\begin{aligned}
\hat{d} &= \frac{to - from}{\|to - from\|} \\
\hat{a} &= \frac{up \times over \times \hat{d}}{\|up \times over \times \hat{d}\|} \\
\hat{b} &= \frac{over \times \hat{d} \times \hat{a}}{\|over \times \hat{d} \times \hat{a}\|} \\
\hat{c} &= \hat{d} \times \hat{a} \times \hat{b}
\end{aligned}
$$

Note two aspects in the equations above: (i) that the input vectors $\overrightarrow{up}$ and $\overrightarrow{over}$ are left unchanged (i.e. $\hat{b} = \overrightarrow{up}$ and $\hat{c} = \overrightarrow{over}$) if they are already orthogonal to each other and orthogonal to the vector from $from$ to to (i.e. $to - from$), and (ii) that the last vector $\hat{c}$ does not need to be normalised since the cross-product already returns a unit vector. These new unit vectors can then be used to define a transformation matrix to transform the 4D coordinates into a new set of points E (as in *eye* coordinates) with a coordinate system with the viewer at its centre and oriented according to the unit vectors. The points are given by:

$$
E = \begin{bmatrix} P - from \end{bmatrix} \begin{bmatrix} \hat{a} & \hat{b} & \hat{c} & \hat{d} \end{bmatrix}
$$

For an **orthographic projection** given $E = \begin{bmatrix} e_0 & e_1 e_2 & e_3 \end{bmatrix}$, the first three columns e_0, e_1 and e_2 can be used as-is, while the fourth column e_3 defines the orthogonal distance to the viewer (i.e. the *depth*).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

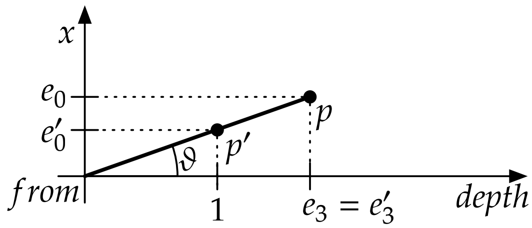


Figure 2. The geometry of a 4D perspective projection along the x axis for a point p. By analysing the depth along the depth axis given by e_3, it is possible to see that the coordinates of the point along the x axis, given by e_0, are scaled inwards in order to obtain e'_0 based on the viewing angle ϑ. Note that $\hat{x}_{n-1}$ is an arbitrary viewing hyperplane and another value can be used just as well.

Finally, in order to obtain a **perspective projection**, he scales the points inwards in direct proportion to their depth. Starting from E, he computes $E' = [e'_0\ e'_1 e'_2\ e'_3]$ as:

$$
\begin{aligned}
e'_0 &= \frac{e_0}{e_3 \tan \vartheta/2} \\
e'_1 &= \frac{e_1}{e_3 \tan \vartheta/2} \\
e'_2 &= \frac{e_2}{e_3 \tan \vartheta/2} \\
e'_3 &= e_3
\end{aligned}
$$

Where ϑ is the viewing angle between x and the line between the $from$ point and every point as shown in Fig. 2. A similar computation is done for y and z. In E', the first three columns (i.e. e'_0, e'_1 and e'_2) similarly give the 3D coordinates for a perspective projection of the 4D points while the fourth column is also the depth of the point.

METHODOLOGY

We present here three different projections from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$ which can be applied iteratively to bring objects of any dimension down to 3D for display. We three projections that are reasonably intuitive in 4D to 3D: a 'long axis' projection that puts 3D objects side by side, the orthographic and perspective projections that work in the same way as their 3D to 2D analogues, and a projection to an $(n-1)$-sphere followed by a stereographic projection to $\mathbb{R}^{n-1}$.

'Long axis' projection

First we aim to replicate the idea behind the example previously shown in Fig. 1—a series of 3D objects that are shown next to each other, seemingly projected separately with the correspondences across scale or time shown as long edges (as in Fig. 1) or faces connecting the 3D objects. Edges would join correspondences between vertices across the models, while faces would join correspondences between elements of dimension up to one (e.g. a pair of edges, or an edge and a vertex). Since every 3D object is apparently projected separately using a perspective projection to 2D, it is thus shown in the same intuitive way in which a single 3D object is projected down to 2D. The result of this projection is shown in Fig. 3a.

Although to the best of our knowledge this projection does not have a well-known name, it is widely used in explanations of 4D and nD geometry—especially when drawn by hand or when the intent is to focus on the connectivity between different elements. For instance, it is usually used in the typical explanation for how to construct a tesseract, i.e. a 4-cube or the 4D analogue of a 2D square or 3D cube, which is based on drawing two cubes and connecting the corresponding vertices between the two (Fig. 4). Among other examples in the scientific literature, this kind of projection can be seen in Figure 2 in Yau and Srihari (1983), Figure 3.4 in Hollasch (1991), Figure 3 in Blanchoff and Cervone (1992), Figures 1–4

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

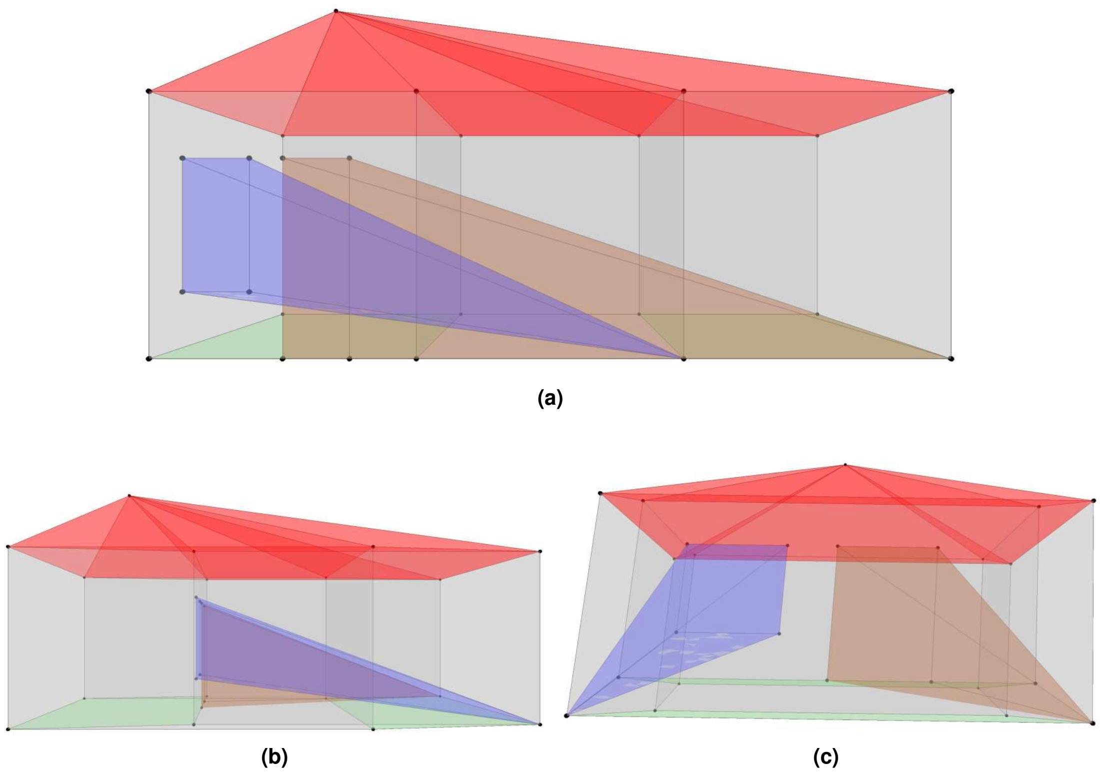


Figure 3. A model of a 4D house similar to the example shown previously in Fig. 1, here including also a window and a door that are collapsed to a vertex in the 3D object at the lower level of detail. (a) shows the two 3D objects positioned as in Fig. 1, (b) rotates these models 90° so that the front of the house is on the right, and (c) orients the two 3D objects front to back. Many more interesting views are possible, but these show the correspondences particularly clearly. Unlike the other model, this one was generated with 4D coordinates and projected using our prototype that applies the projection described in this section.

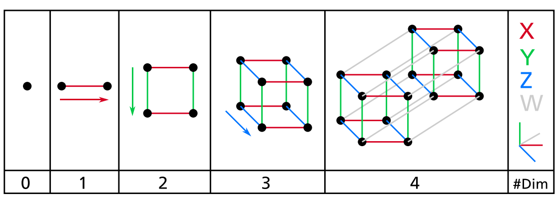


Figure 4. The typical explanation for how to draw the vertices and edges in an i-cube. Starting from a single vertex representing a point (i.e. a 0-cube), an $(i+1)$-cube can be created by drawing two i-cubes and connecting the corresponding vertices of the two. Image credit: Wikimedia Commons.

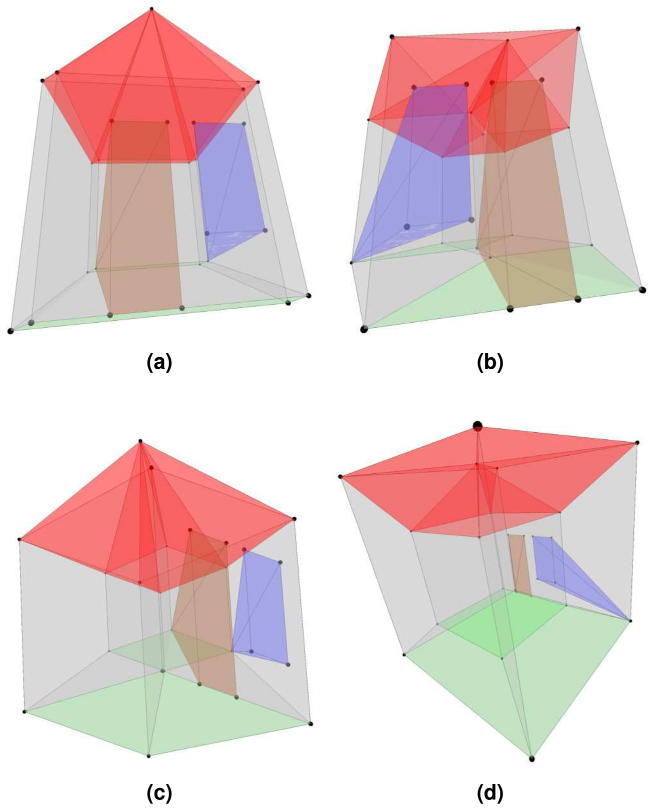


Figure 5. The 4D house model projected down to 3D using an orthographic projection. The different views are obtained by applying different rotations in 4D. The less and more detailed 3D models can be found by looking at where the door and window are collapsed.

in Arenas and Pérez-Aguila (2006), Figure 6 in Grasset-Simon et al. (2006), Figure 1 in Paul (2012) and Figure 16 in van Oosterom and Meijers (2014).

Conceptually, describing this projection from n to $n-1$ dimensions, which we hereafter refer to as a 'long axis' projection, is very simple. Considering a set of points P in $\mathbb{R}^n$, the projected set of points P' in $\mathbb{R}^{n-1}$ is given by taking the coordinates of P for the first $n-1$ axes and adding to them the last coordinate of P which is spread over all coordinates according to weights specified in a customisable vector $\hat{x}_n$. For instance, Fig. 3 uses $\hat{x}_n = [2\ 0\ 0]$, resulting in 3D models that are 2 units displaced for every unit in which they are apart along the n-th axis. In matrix form, this kind of projection can then be applied as $P' = P[I\ \hat{x}_n]$.

Orthographic and perspective projections

Another reasonably intuitive pair of projections are the orthographic and perspective projections from nD to $(n-1)$D. These treat all axes similarly and thus make it more difficult to see the different $(n-1)$-dimensional models along the n-th axis, but they result in models that are much less deformed. Also, as shown in the 4D example in Fig. 5, it is easy to rotate models in such a way that the corresponding features are easily seen.

Based on the description of 4D-to-3D orthographic and perspective projection described from Hollasch (1991), we here extend the method in order to describe the n-dimensional to $(n-1)$-dimensional case, changing some aspects to give a clearer geometric meaning for each vector.

Similarly, we start with a point $from \in \mathbb{R}^n$ where the viewer is located, a point $to \in \mathbb{R}^n$ that the viewer directly points towards (which can be easily set to the centre or centroid of the dataset), and a set of $n-2$ initial vectors $\overrightarrow{v}_1, \ldots, \overrightarrow{v}_{n-2}$ in $\mathbb{R}^n$ that are not all necessarily orthogonal but nevertheless are linearly independent from each other and from the vector $to - from$. In this setup, the $\overrightarrow{v}_i$ vectors serve as a base to define the *orientation* of the system, much like the traditional $\overrightarrow{up}$ vector that is used in 3D to 2D

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

projections and the $\overrightarrow{over}$ vector described previously. From the above mentioned variables and using the nD cross-product, it is possible to define a new set of orthogonal unit vectors $\hat{x}_0, \ldots, \hat{x}_{n-1}$ that define the axes $x_0, \ldots, x_{n-1}$ of a coordinate system in $\mathbb{R}^n$ as:

$$
\begin{aligned}
\hat{x}_{n-1} &= \frac{to - from}{\|to - from\|} \\
\hat{x}_0 &= \frac{\overrightarrow{v}_1 \times \cdots \times \overrightarrow{v}_{n-2} \times \hat{x}_{n-1}}{\|\overrightarrow{v}_1 \times \cdots \times \overrightarrow{v}_{n-2} \times \hat{x}_{n-1}\|} \\
\hat{x}_i &= \frac{\overrightarrow{v}_{i+1} \times \cdots \times \overrightarrow{v}_{n-2} \times \hat{x}_{n-1} \times \hat{x}_0 \times \cdots \times \hat{x}_{i-1}}{\|\overrightarrow{v}_{i+1} \times \cdots \times \overrightarrow{v}_{n-2} \times \hat{x}_{n-1} \times \hat{x}_0 \times \cdots \times \hat{x}_{i-1}\|} \\
\hat{x}_{n-2} &= \hat{x}_{n-1} \times \hat{x}_0 \times \cdots \times \hat{x}_{n-2}
\end{aligned}
$$

The vector $\hat{x}_{n-1}$ is the first that needs to be computed and is oriented along the line from the viewer ($from$) to the point that it is oriented towards (to). Afterwards, the vectors are computed in order from $\hat{x}_0$ to $\hat{x}_{n-2}$ as normalised n-dimensional cross products of $n-1$ vectors. These contain a mixture of the input vectors $\overrightarrow{v}_1, \ldots, \overrightarrow{v}_{n-2}$ and the computed unit vectors $\hat{x}_0, \ldots, \hat{x}_{n-1}$, starting from $n-2$ input vectors and one unit vector for $\hat{x}_0$, and removing one input vector and adding the previously computed unit vector for the next $\hat{x}_i$ vector. Note that if $\overrightarrow{v}_1, \ldots, \overrightarrow{v}_{n-2}$ and $\hat{x}_{n-1}$ are all orthogonal to each other, $\forall 0 < i < n-1$, $\hat{x}_i$ is simply a normalised $\overrightarrow{v}_i$.

Like in the previous case, the vectors $\hat{x}_0, \ldots, \hat{x}_{n-1}$ can then be used to transform an $m \times n$ matrix of m nD points in world coordinates P into an $m \times n$ matrix of m nD points in eye coordinates E by applying the following transformation:

$$
E = \begin{bmatrix} P - from \end{bmatrix} \begin{bmatrix} \hat{x}_0 & \cdots & \hat{x}_{n-1} \end{bmatrix}
$$

As before, if E has rows of the form $\begin{bmatrix} e_0 & \cdots & e_{n-1} \end{bmatrix}$ representing points, $e_0, \ldots, e_{n-2}$ are directly usable as the coordinates in $\mathbb{R}^{n-1}$ of the projected point in an n-dimensional to $(n-1)$-dimensional **orthographic projection**, while e_{n-1} represents the depth, i.e. the distance between the point and the projection $(n-1)$-dimensional subspace, which can be used for visual cues[1]. The coordinates along $e_0, \ldots, e_{n-2}$ could be made to fit within a certain bounding box by computing their extent along each axis, then scaling appropriately using the extent that is largest in proportion to the extent of the bounding box's corresponding axis.

For an n-dimensional to $(n-1)$-dimensional **perspective projection**, it is only necessary to compute the distance between a point and the viewer along every axis by taking into account the viewing angle ϑ between $\hat{x}_{n-1}$ and the line between the to point and every point. Intuitively, this means that if an object is n times farther than another identical object, it is depicted n times smaller, or $\frac{1}{n}$ of its size. This situation is shown in Fig. 6 and results in new $e'_0, \ldots, e'_{n-2}$ coordinates that are shifted inwards. The coordinates are computed as:

$$
e'_i = \frac{e_i}{e_{n-1} \tan \vartheta / 2}, \quad \text{for } 0 \leq i \leq n-2
$$

The $(n-1)$-dimensional coordinates generated by this process can then be recursively projected down to progressively lower dimensions using this method. The objects represented by these coordinates can also be discretised into images of any dimension. For instance, Hanson (1994) describes how to perform many of the operations that would be required, such as dimension-independent clipping tests and ray-tracing methods.

Stereographic projection

A final projection possibility is to apply a stereographic projection from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$, which for us was partly inspired by Jenn 3D[2] (Fig. 7). This program visualises polyhedra and polychora embedded in

[1] Visual cues can still be useful in higher dimensions. See http://eusebeia.dyndns.org/4d/vis/08-hsr.
[2] http://www.math.cmu.edu/~fho/jenn/

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

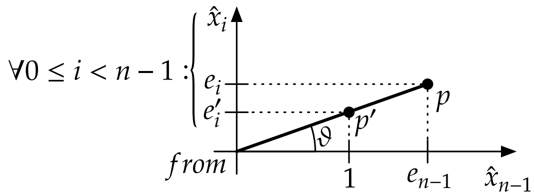
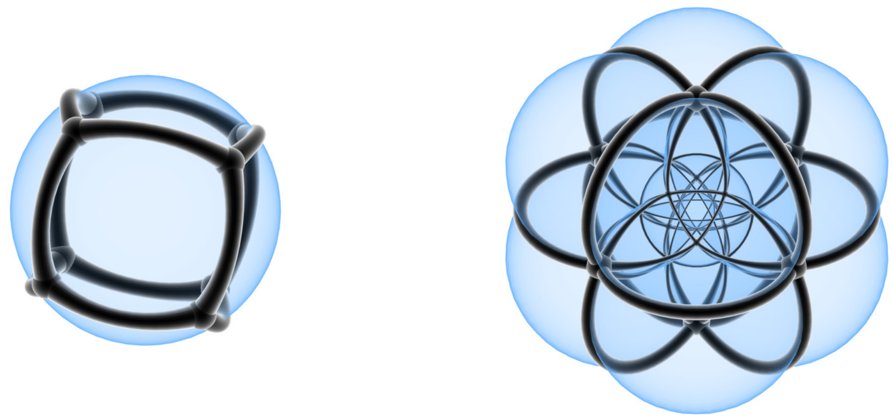


Figure 6. The geometry of an nD perspective projection for a point p. By analysing each axis $\hat{x}_i$ ($\forall 0 \leq i < n-1$) independently together with the final axis $\hat{x}_{n-1}$, it is possible to see that the coordinates of the point along that axis, given by e_i, are scaled inwards based on the viewing angle ϑ.



(a) (b)

Figure 7. A polyhedron and a polychoron in Jenn 3D: (a) a cube and (b) a 24-cell.

$\mathbb{R}^4$ by first projecting them inwards/outwards to the volume of a 3-sphere[3] and then projecting them stereographically to $\mathbb{R}^3$, resulting in curved edges, faces and volumes.

In a dimension-independent form, this type of projection can be easily done by considering the angles $\vartheta_0, \ldots, \vartheta_{n-2}$ in an *n-dimensional spherical coordinate system*. Steeb (2011, §12.2) formulates such a system as:

$$
\begin{aligned}
r &= \sqrt{x_0^2 + \cdots + x_{n-1}^2} \\
\vartheta_i &= \cos^{-1}\left(\frac{x_i}{\sqrt{r^2 - \sum_{j=0}^{i-1} x_j^2}}\right), \quad \text{for } 0 \leq i < n-2 \\
\vartheta_{n-2} &= \tan^{-1}\left(\frac{x_{n-1}}{x_{n-2}}\right)
\end{aligned}
$$

It is worth to note that the radius r of such a coordinate system is a measure of the depth with respect to the projection $(n-1)$-sphere S^{n-1} and can be used similarly to the previous projection examples. The points can then be converted back into points on the surface of an $(n-1)$-sphere of radius 1 by making $r = 1$ and applying the inverse transformation. Steeb (2011, §12.2) formulates it as:

[3] Intuitively, an unbounded volume that wraps around itself, much like a 2-sphere can be seen as an unbounded surface that wraps around itself.

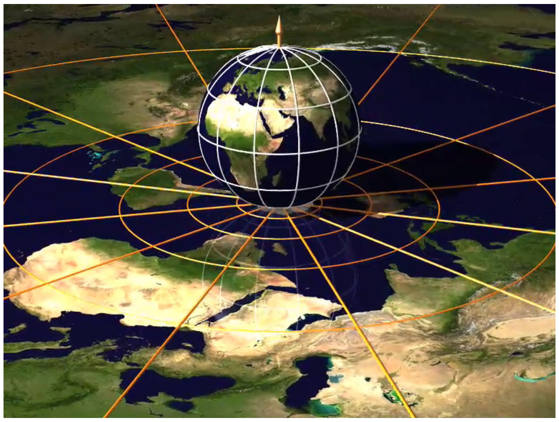


Figure 8. The most common use of the stereographic projection is to map the surface of the Earth to the plane. Here, every point p on the sphere is projected to the intersection of the plane with a line passing through the North pole and p. Image credit: screen capture from Leys et al. (2008).

$$x_i = r\cos\vartheta_i \prod_{j=0}^{i-1} \sin\vartheta_j, \quad \text{for } 0 \leq i < n-2$$

$$x_{n-1} = r\prod_{j=0}^{n-2} \sin\vartheta_j$$

The next step, a stereographic projection (Fig. 8), is also easy to apply in higher dimensions, mapping an $(n+1)$-dimensional point $x = (x_0, \ldots, x_n)$ on an n-sphere S^n to an n-dimensional point $x' = (x_0, \ldots, x_{n-1})$ in the n-dimensional Euclidean space $\mathbb{R}^n$. Chisholm (2000) formulates this projection as:

$$x'_i = \frac{x_i}{x_n - 1}, \quad \text{for } 0 \leq i < n$$

The stereographic projection from nD to $(n-1)$D is particularly intuitive because it results in the n-th axis being converted into an inwards-outwards axis. As shown in Fig. 9, when it is applied to scale, this results in models that decrease or increase in detail as one moves inwards or outwards. The case with time is similar: as one moves inwards/outwards, it is easy to see the state of a model at a time before/after.

RESULTS

We have implemented a small prototype for an interactive viewer of arbitrary 4D objects that performs the three projections previously described. It was used to generate Figures 3, 5 and 9, which were obtained by moving around the scene, zooming in/out and capturing screenshots using the software.

The prototype was implemented using part of the codebase of azul[4] and is written in a combination of Swift 3 and C++11 using Metal (a low-level and low-overhead graphics API) under macOS 10.12[5]. Its source code is available under the GPLv3 licence at `https://github.com/kenohori/azul4d`.

For this prototype, we only consider the vertices, edges and faces of the 4D objects, as the higher-dimensional 3D and 4D primitives—whose 0D, 1D and 2D boundaries are however shown—would readily obscure each other in any sort of 2D or 3D visualisation (Banks, 1992). Every face of an object is thus stored as a sequence of vertices with coordinates in $\mathbb{R}^4$ and is appended with an RGBA colour attribute with possible transparency. The alpha value of each face is used see all faces at once, as they would otherwise overlap with each other on the screen.

[4] `https://github.com/tudelft3d/azul`
[5] `https://developer.apple.com/metal/`

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

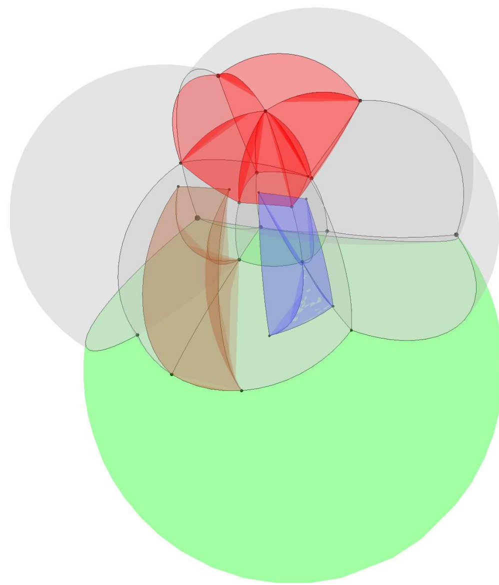


Figure 9. The 4D house model projected first inwards/outwards to the closest point on the 3-sphere S^3 and then stereographically to $\mathbb{R}^3$. The round surfaces are obtained by first refining every face in the 4D model.

The 4D models were manually constructed based on defining their vertices with 4D coordinates and their faces as successions of vertices. In addition to the 4D house previously shown, we built a simpler tesseract for testing (Fig. 10). As built, the tesseract consists of 16 vertices and 24 vertices, while the 4D house consists of 24 vertices and 43 faces. However, we used the face refining process described below to test our prototype with models with up to a few thousand faces. Once created, the models were still displayed and manipulated smoothly.

To start, we preprocess a 4D model by triangulating and possibly refining each face, which makes it possible to display concave faces and to properly see the curved shapes that are caused by the stereographic projection previously described. For this, we first compute the plane passing through the first three points of each face[6] and project each point from $\mathbb{R}^4$ to a new coordinate system in $\mathbb{R}^2$ on the plane. We then triangulate and refine separately each face in $\mathbb{R}^2$ with the help of a few packages of the Computational Geometry Algorithms Library (CGAL)[7], and then we reproject the results back to the previously computed plane in $\mathbb{R}^4$.

We then use a Metal Shading Language compute shader—a technique to perform general-purpose computing on graphics processing units (GPGPU)—in order to apply the desired projection from $\mathbb{R}^4$ to $\mathbb{R}^3$. The three different projections presented previously are each implemented as a compute shader. This is necessary because we want to extract the projected $\mathbb{R}^3$ vertex coordinates of every face and use them to generate separate representations of their bounding edges and vertices[8]. Using their projected coordinates in $\mathbb{R}^3$, the edges and vertices surrounding each face are thus displayed respectively as possibly refined line segments and as icosahedral approximations of spheres (icospheres).

Finally, we use a standard perspective projection in a Metal vertex shader to display the projected model with all its faces, edges and vertices. We use a couple of tricks in order to keep the process fast and as parallel as possible: separate threads for each CPU process (the generation of the vertex and edge geometries and the modification of the projection matrices according to user interaction) and GPU process (4D-to-3D projection and 3D-to-2D projection for display), and blending with order-independent transparency without depth checks. For complex models, this results in a small lag where the vertices and

[6]This is sufficient for our purposes, but other applications would need to find three linearly-independent points or to use a more computationally expensive method that finds the best fitting plane for the face.

[7]`http://www.cgal.org`

[8]An alternative would be to embed these in 4D from the beginning, but it would result in distorted shapes depending on their position and orientation due to the extra degrees of rotational freedom in $\mathbb{R}^4$.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

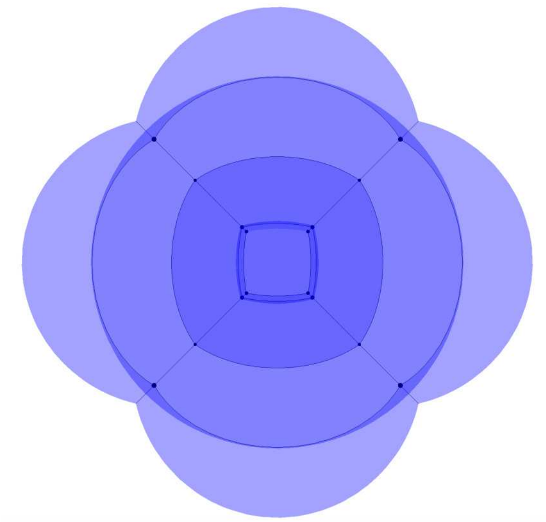


Figure 10. A tesseract visualised using the $\mathbb{R}^4$ to S^3 to $\mathbb{R}^3$ projection.

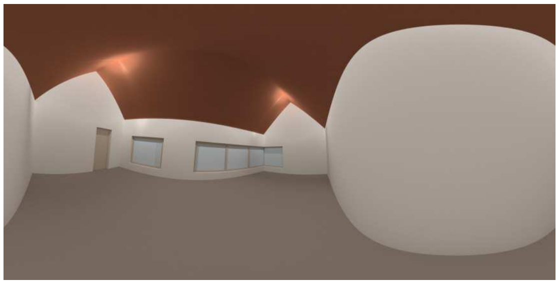


Figure 11. The Equirectangular projection directly maps angles to coordinates. A 360° view is here mapped into a rectangular 180°×360° image. Rendered from a viewpoint inside the IfcOpenHouse dataset[9].

edges move slightly after the faces.

DISCUSSION AND CONCLUSIONS

Visualising complete 4D and *n*D objects projected to 3D and displayed in 2D is often unintuitive, but it enables analysing higher-dimensional objects in a thorough manner that cross-sections do not. The three projections we have shown here are nevertheless reasonably intuitive due to their similarity to common projections from 3D to 2D, the relatively small distortions in the models and the existence of a clear fourth axis. They also have a dimension-independent formulation.

There are however many other types of interesting projections that can be defined in any dimension, such as the equirectangular projection shown in Fig. 11 where evenly spaced angles along a *rotation plane* can be directly converted into evenly spaced coordinates—in this case covering 180° vertically and 360° horizontally. Extending such a projection to *n*D would result in an *n*-orthotope, such as a (filled) rectangle in 2D or a cuboid (i.e. a box) in 3D.

By applying the projections shown in this paper to 4D objects depicting 3D objects that change in time or scale, it is possible to see at once all correspondences between different elements of the 3D objects and the topological relationships between them.

Compared to other 4D visualisation techniques, we opt for a rather minimal approach without lighting and shading. In our application, we believe that this is optimal due to better performance and because it makes for simpler-looking and more intuitive output. In this manner, progressively darker shades of a colour are a good visual cue for the number of faces of the same colour that are visually overlapping at

[9] `http://blog.ifcopenshell.org/2012/11/say-hi-to-ifcopenhouse.html`

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

any given point. Since we apply the projection from 4D to 3D in the GPU, it is cumbersome to extract the surfaces again in order to compute the 3D normals required for lighting in 3D, while lighting in 4D results in unintuitive visual cues.

REFERENCES

Abbott, E. A. (1884). *Flatland: A Romance of Many Dimensions*. Seely & Co.

Abiteboul, S. and Hull, R. (1987). Update propagation in the IFO database model. In Ghosh, S. P., Kambayashi, Y., and Tanaka, K., editors, *Foundations of Data Organization*, pages 319–331. Springer US.

Arenas, Y. and Pérez-Aguila, R. (2006). Visualizing 3d projections of higher dimensional polytopes: An approach linking art and computers. In *Memorias del Cuarto Congreso Nacional de Ciencias de la Computacion*.

Armstrong, M. P. (1988). Temporality in spatial databases. In *GIS/LIS '88 : proceedings : accessing the world : third annual International Conference, Exhibits, and Workshops*, pages 880–889. American Society for Photogrammetry and Remote Sensing.

Arroyo Ohori, K. (2016). *Higher-dimensional modelling of geographic information*. PhD thesis, Delft University of Technology.

Arroyo Ohori, K., Boguslawski, P., and Ledoux, H. (2013). Representing the dual of objects in a four-dimensional GIS. In Abdul Rahman, A., Boguslawski, P., Gold, C., and Said, M., editors, *Developments in Multidimensional Spatial Data Models*, Lecture Notes in Geoinformation and Cartography, pages 17–31. Springer Berlin Heidelberg, Johor Bahru, Malaysia.

Arroyo Ohori, K., Damiand, G., and Ledoux, H. (2014). Constructing an n-dimensional cell complex from a soup of (n-1)-dimensional faces. In Gupta, P. and Zaroliagis, C., editors, *Applied Algorithms. First International Conference, ICAA 2014, Kolkata, India, January 13-15, 2014. Proceedings*, volume 8321 of *Lecture Notes in Computer Science*, pages 37–48. Springer International Publishing Switzerland, Kolkata, India.

Arroyo Ohori, K., Ledoux, H., Biljecki, F., and Stoter, J. (2015a). Modelling a 3D city model and its levels of detail as a true 4D model. *ISPRS International Journal of Geo-Information*, 4(3):1055–1075.

Arroyo Ohori, K., Ledoux, H., and Stoter, J. (2015b). An evaluation and classification of nD topological data structures for the representation of objects in a higher-dimensional GIS. *International Journal of Geographical Information Science*, 29(5):825–849.

Arroyo Ohori, K., Ledoux, H., and Stoter, J. (2015c). Storing a 3D city model, its levels of detail and the correspondences between objects as a 4D combinatorial map. In Rahman, A. A., Isikdag, U., and Castro, F. A., editors, *Joint International Geoinformation Conference 2015, 28–30 October 2015, Kuala Lumpur, Malaysia*, volume II–2/W2 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 1–8, Kuala Lumpur, Malaysia. ISPRS.

Ballard, D. H. (1981). Strip trees: A hierarchical representation for curves. *Communications of the ACM*, 24(5):310–321.

Banchoff, T. F. (1996). *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions*. Scientific American Library Series.

Banks, D. (1992). Interactive manipulation and display of surfaces in four dimensions. In *I3D '92 Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 197–207. ACM.

Beshers, C. M. and Feiner, S. K. (1988). Real-time 4D animation on a 3D graphics workstation. In *Graphics Interface '88*, pages 1–7. CHCCS/SCDHM.

Bhaniramka, P., Wenger, R., and Crawfis, R. (2000). Isosurfacing in higher dimensions. In *VIS '00 Proceedings of the conference on Visualization '00*. IEEE.

Bieri, H. and Nef, W. (1988). Elementary set operations with d-dimensional polyhedra. In Noltemeier, H., editor, *Computational Geometry and its Applications*, volume 333 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin Heidelberg.

Blanchoff, T. and Cervone, D. P. (1992). Illustrating beyond the third dimension. *Leonardo*, 25(3–4):273–280.

Brisson, E. (1993). Representing geometric structures in d dimensions: topology and order. *Discrete & Computational Geometry*, 9:387–426.

Buttenfield, B. P. and DeLotto, J. S. (1989). Multiple representations: Scientific report for the specialist meeting. Technical Report 89–3, National Center for Geographic Information and Analysis.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

Chisholm, M. (2000). The sphere in three dimensions and higher: Generalizations and special cases. Available at `https://theory.org/geotopo/3-sphere/3-sphere.ps`.

Chrisman, N. R. (1983). The role of quality information in the long-term functioning of a geographic information system. *Cartographica*.

Chu, A., Fu, C.-W., Hanson, A. J., and Heng, P.-A. (2009). GL4D: A GPU-based architecture for interactive 4D visualization. In *IEEE Transactions on Visualization and Computer Graphics*, volume 15, pages 1587–1594. IEEE.

Čomić, L. and de Floriani, L. (2012). *Modeling and Manipulating Cell Complexes in Two, Three and Higher Dimensions*, volume 2 of *Lecture Notes in Computational Vision and Biomechanics*, chapter 4, pages 109–144. Springer.

Descartes, R. (1637). *Discours de la méthode*. Jan Maire, Leyde.

D'Zmura, M., Colantoni, P., and Seyranian, G. (2000). Virtual environments with four or more spatial dimensions. *Presence*, 9(6):616==631.

Elduque, A. (2004). Vector cross products. Talk presented at the Seminario Rubio de Francia of the Universidad de Zaragoza on April 1, 2004.

Feiner, S. and Beshers, C. (1990). Visualizing n-dimensional virtual worlds with n-vision. In *Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 37–38. ACM.

Filho, W. C., de Figueiredo, L. H., Gattass, M., and Carvalho, P. C. (1995). A topological data structure for hierarchical planar subdivisions. Technical Report CS-95-53, Department of Computer Science, University of Waterloo.

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1995). *Computer Graphics: Principles and Practice in C*. Addison-Wesley Professional.

Foley, T. A. and Nielson, G. M. (1992). Practical techniques for producing 3D graphical images. In Black, J., editor, *The System Engineer's Handbook: A guide to building VMEbus and VXIbus systems*, chapter 19, pages 223–237. Academic Press.

Frank, A. U. (2014). Four-dimensional representation in human cognition and difficulties with demonstrations: A commentary on wang. *Spatial Cognition & Computation*, 14:114–120.

Friis-Christensen, A. and Jensen, C. S. (2003). Object-relational management of multiply represented geographic entities. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, pages 150–159. IEEE Computer Society.

Grafarend, E. W. and You, R.-J. (2014). *Map Projections: Cartographic Information Systems*. Springer-VerlagBerlinHeidelber.

Grasset-Simon, C., Damiand, G., and Lienhardt, P. (2006). nd generalized map pyramids: definition, representations and basic operations. *Pattern Recognition*, 39(4):527–538.

Gröger, G., Kolbe, T. H., Nagel, C., and Häfele, K.-H. (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard. Version 2.0.0*. Open Geospatial Consortium.

Guibas, L. J. and Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123.

Günther, O. (1988). The arc tree: An approximation scheme to represent arbitrary curved shapes. In *Efficient structures for geometric data management*, volume 337 of *Lecture Notes in Computer Science*, chapter 6, pages 85–121. Springer Berlin Heidelberg.

Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1–42.

Hägerstrand, T. (1970). What about people in regional science? *Papers of the Regional Science Association*, 24(1):6–21.

Hamre, T., Mughal, K. A., and Jacob, A. (1997). A 4D marine data model: Design and application in ice monitoring. *Marine Geodesy*, 20(2–3):121–136.

Hansen, H. Ø. and Christensen, N. J. (1993). A model for n-dimensional boundary topology. In *Proceedings of the 2nd ACM Symposium on Solid Modelling and Applications*. ACM.

Hanson, A. J. (1994). Geometry for n-dimensional graphics. In Heckbert, P. S., editor, *Graphics Gems IV*, chapter II.6, pages 149–170. Academic Press Professional.

Hanson, A. J. and Cross, R. A. (1993). Interactive visualization methods for four dimensions. In *VIS '93 Proceedings of the 4th conference on Visualization '93*, pages 196–203. ACM.

Hanson, A. J., Ishkov, K. I., and Ma, J. H. (1999). Meshview: Visualizing the fourth dimension. Technical

report, Indiana University.
Hinton, C. H. (1888). *A New Era of Thought*. Swan Sonnenschein & Co. Ltd.
Hollasch, S. R. (1991). Four-space visualization of 4D objects. Master's thesis, Arizona State University.
Hornsby, K. and Egenhofer, M. J. (2002). Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1–2).
Jones, C. and Abraham, I. (1986). Design considerations for a scale-independent cartographic database. In Marble, D., editor, *Proceedings of the 2nd International Symposium on Spatial Data Handling*, pages 384–398.
Kada, M. (2007). Scale-dependent simplification of 3D building models based on cell decomposition and primitive instancing. In *COSIT 2007*, volume 4736 of *Lecture Notes in Computer Science*, pages 222–237. Springer-Verlag Berlin Heidelberg.
Kageyama, A. (2016). A visualization method of four dimensional polytopes by oval display of parallel hyperplane slices. Available at `https://arxiv.org/pdf/1607.01102.pdf`.
Kraak, M.-J. (2003). The space-time cube revisited from a geovisualization perspective. In *Proceedings of the 21st International Cartographic Conference*, pages 1988–1996.
Leys, J., Ghys, E., and Alvarez, A. (2008). Dimensions: une promenade mathématique. Available at `http://www.dimensions-math.org/Dim_E.htm`.
Lienhardt, P. (1994). *n*-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275–324.
Massey, W. S. (1983). Cross products of vectors in higher dimensional Euclidean spaces. *The American Mathematical Monthly*, 90(10):697–701.
Masuda, H. (1993). Topological operators and Boolean operations for complex-based non-manifold geometric models. *Computer-Aided Design*, 25(2).
McKenzie, J. W., Williamson, I. P., and Hazelton, N. (2001). 4-D adaptive GIS: Justification and methodologies. Technical report, Department of Geomatics, The University of Melbourne.
McMullen, C. (2008). *The Visual Guide To Extra Dimensions: Visualizing The Fourth Dimension, Higher-Dimensional Polytopes, And Curved Hypersurfaces*. CreateSpace Independent Publishing Platform.
Meijers, M. (2011a). The space-scale cube: an integrated model for 2d polygonal areas and scale. In Fendel, E. M., Ledoux, H., Rumor, M., and Zlatanova, S., editors, *Proceedings of the 28th Urban Data Management Symposium*, volume XXXVIII-4/C21, pages 95–101, Delft, The Netherlands. ISPRS Archives.
Meijers, M. (2011b). *Variable-scale Geo-information*. PhD thesis, Delft University of Technology.
Muller, D. E. and Preparata, F. P. (1978). Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7(2):217–236.
Noll, A. M. (1967). A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM*, 10(8):469–473.
Noll, A. M. (1968). Computer animation and the fourth dimension. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part II*, pages 1279–1283. ACM.
Paul, N. (2012). Signed simplicial decomposition and overlay of n-d polytope complexes. Available at `http://arxiv.org/abs/1205.5691`.
Peucker, T. K. and Chrisman, N. R. (1975). Cartographic data strutures. *The American Cartographer*, 2(1):55–69.
Peuquet, D. J. (1994). It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems. *Annals of the Association of American Geographers*, 84(3):441–461.
Peuquet, D. J. and Duan, N. (1995). An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data. *International Journal of Geographical Information Science*, 9(1):7–24.
Plümer, L. and Gröger, G. (1997). Achieving integrity in geographic information systems—maps and nested maps. *GeoInformatica*, 1(4):345–367.
Poincaré, M. (1895). Analysis situs. *Journal de l'École polytechnique*, 2(1):1–123.
Requicha, A. A. G. and Voelcker, H. B. (1977). Constructive solid geometry. Technical Memorandum 25, College of Engineering & Applied Science, The University of Rochester.
Riemann, B. (1868). *Ueber die Hypothesen, welche der Geometrie zu Grunde liegen*. PhD thesis, Abhandlungen der Königlichen Gesellschaft der Wissenschaften zu Göttingen.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017

Rigaux, P. and Scholl, M. (1995). Multi-scale partitions: Application to spatial and statistical databases. In Egenhofer, M. J. and Herring, J. R., editors, *Advances in Spatial Databases*, volume 951 of *Lecture Notes in Computer Science*, pages 170–183. Springer Berlin Heidelberg.

Rossignac, J. and O'Connor, M. (1989). SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries. In Wosny, M., Turner, J., and Preiss, K., editors, *Proceedings of the IFIP Workshop on CAD/CAM*, pages 145–180.

Shreiner, D., Sellers, G., Kessenich, J., Licea-Kane, B., and Khronos ARB Working Group (2013). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. Addison-Wesley, 8th edition.

Snyder, J. P. (1987). *Map Projections—A Working Manual*. U.S. Geological Survey.

Sohanpanah, C. (1989). Extension of a boundary representation technique for the description of n dimensional polytopes. *Computers & Graphics*, 13(1):17–23.

Steeb, W.-H. (2011). *The Nonlinear Workbook*. World Scientific Publishing, 5th edition.

van Oosterom, P. (1990). *Reactive Data Structures for Geographic Information Systems*. PhD thesis, Leiden University.

van Oosterom, P. (2005). Variable-scale topological data structures suitable for progressive data transfer: The GAP-face tree and GAP-edge forest. *Cartography and Geographic Information Science*, 32(4):331–346.

van Oosterom, P. and Meijers, M. (2014). Vario-scale data structures supporting smooth zoom and progressive transfer of 2D and 3D data. *International Journal of Geographical Information Science*, 28:455–478.

van Oosterom, P. and Stoter, J. (2010). 5D data modelling: Full integration of 2D/3D space, time and scale dimensions. In Fabrikant, S. I., Reichenbacher, T., van Kreveld, M., and Schlieder, C., editors, *Geographic Information Science: 6th International Conference, GIScience 2010, Zurich, Switzerland, September 14-17, 2010. Proceedings*, pages 311–324. Springer Berlin Heidelberg.

Wachowicz, M. and Healy, R. G. (1994). Towards temporality in gis. In *Innovations in GIS*. Taylor & Francis.

Worboys, M. (1992). A model for spatio-temporal information. In *Proceedings of the 5th International Symposium on Spatial Data Handling*, pages 602–611.

Worboys, M. F. (1994). A unified model for spatial and temporal information. *The Computer Journal*, 37(1):26–34.

Worboys, M. F., Hearnshaw, H. M., and Maguire, D. J. (1990). Object-oriented data modelling for spatial databases. *International Journal of Geographical Information Systems*, 4(4):369–383.

Yau, M.-M. and Srihari, S. N. (1983). A hierarchical data structure for multidimensional digital images. *Communications of the ACM*, 26(7):504–515.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2844v1 | CC BY 4.0 Open Access | rec: 2 Mar 2017, publ: 2 Mar 2017