A METHOD OF PROCESSING INFORMATION THAT IS INDICATIVE OF A SHAPE

100

102

Projecting a plurality of vertices of geometrical shapes that are defined in three-dimensions onto a plane.

104

Determining reference information that is usable to associate the projected vertices with each other, the reference information being determined by applying a reversible function to the projected vertices

Figure 1

The present disclosure provides a method of processing information that is indicative of a shape in three-dimensional space. The processed information is usable in displaying a representation of the shape. The method comprises projecting a plurality of vertices of the shape onto a plane. The method also comprises determining reference information that is usable to associate the projected vertices with each other. The reference information is determined by applying a reversible function to the projected vertices. Information that is indicative of the projected vertices and the reference information can be used to represent the shape on a display.
A METHOD OF PROCESSING INFORMATION THAT IS INDICATIVE OF A SHAPE

Field of the Invention

The present invention relates to a method of processing information that is indicative of a shape. The invention has been developed primarily for use in a computing environment and will be described hereinafter with reference to this application. However, it will be appreciated that the invention is not limited to this particular field of use.

Background of the Invention

The transmission of two-dimensional rendered images between computing systems (and/or from a computing system to a display) remains a popular method for sending images across a computing network, say when a user is playing a networked game across the Internet. Two-dimensional images tend to maintain a low near constant bandwidth cost despite any geometric complexity, making them an attractive format for transmission in situations where there are inherent limitations in bandwidth.

However, in order to provide a more realistic and immersive experience for an end user, it is desirable to transmit and render three-dimensional images, which provide a "richer" end user experience.

In the past, methods and systems which attempted to send three-dimensional image data over a network concentrated on methods that sent all image data to the end user, so
that a user would be able to fully appreciate the three-dimensional image in real time. For example, a user navigating a 'scene' in a three-dimensional game would like to be able to navigate the scene in real time, with little or no noticeable delay or lag time. As such, the user will typically be required to download all visible geometry data to achieve the given experience.

However, delivering all three-dimensional image data requires high bandwidth capabilities or the ability to successfully compress the image data to reduce bandwidth requirements. Existing methods for three-dimensional compression tend to focus on compression methods that require offline processing. These techniques are not well suited to runtime, low bandwidth, low computation computing systems and networks. As a result, the use of image based compression techniques for sending three-dimensional image data remains the dominant form of delivery.

Summary of the Invention

In accordance with a first aspect of the present invention, there is provided a method of processing information that is indicative of a shape in three-dimensional space, the processed information being usable in displaying a representation of the shape, the method comprising the steps of:

projecting a plurality of vertices of the shape onto a plane; and

determining reference information that is usable to associate the projected vertices with each other, the reference information being determined by applying a
reversible function to the projected vertices;
wherein information that is indicative of the
projected vertices and the reference information can be
used to represent the shape on a display.

Embodiments of the present invention may provide the
advantage wherein three dimension image data having a
geometric complexity may be encoded and transmitted with
an improved proportional cost of bandwidth. This may
generate an effective bandwidth scalable solution for the
transmission of three dimension content.

The reference information may be indicative of a point in
two-dimensional space.

The point in two-dimensional space may be bound by its
associated projected vertices.

The reversible function may be one of a centroid method or
a bisector method, although it will be appreciated that
any appropriate reversible function can be used.

The method may comprise a step of applying a compression
algorithm to compress the information that is indicative
of the projected vertices and the reference information.

The shape may be one of a plurality of shapes that
comprise a representation of an object in three-
dimensional space, and the method may comprise the steps
of:
projecting a plurality of vertices of each shape onto
the plane; and
for each shape, determining reference information
that is usable to associate the projected vertices of the
shape with each other, the reference information being
determined by applying a reversible function to the
projected vertices of the shape;
wherein information that is indicative of the
projected vertices and the reference information can be
used to represent the shapes that comprise the object on a
display.

In accordance with a second aspect of the invention, there
is provided a method of processing three-dimensional image
data comprising the steps of:
processing the three-dimensional image data to
provide a data set for a corresponding two-dimensional
image; and
adding at least one set of pixel data to the two-
dimensional image data set, the at least one set of pixel
data comprising additional data that is representative of
depth information of the three-dimensional image data.

In an embodiment of the second aspect, the at least one
set of pixel data is a bit plane incorporated into the
two-dimensional image data.

In an embodiment of the second aspect, the bit plane is a
matrix, wherein each element in the matrix represents a
pixel of the two-dimensional image data.

In an embodiment of the second aspect, each element in the
matrix that represents a vertex of the three-dimensional
image data is represented by a first value, and each
element in the matrix that does not represent a vertex of
the three-dimensional image data is represented by a
second value.
In an embodiment of the second aspect, the first value is an integer value of one (1) and the second value is an integer value of zero (0).

In an embodiment of the second aspect, the method further comprises the step of determining vertex information that is indicative of a two dimensional position of a plurality of vertices of the three-dimensional data.

In an embodiment of the second aspect, the method comprises a step of determining a connectivity between each vertex utilising a connectivity algorithm and encoding the connectivity data in an additional set of pixel data.

In an embodiment of the second aspect, the at least one set of additional pixel data is an additional bit plane incorporated into the two-dimensional image data.

In an embodiment of the second aspect, the bit plane is a matrix, wherein each element in the matrix represents a pixel of the image information.

In an embodiment of the second aspect, the connectivity data is derived by traversing a triangulated irregular network to encode a single bit for each triangle, each single bit being representative of a connectivity point in the connectivity data.

In an embodiment of the second aspect, the connectivity data is derived using a generalised relationship:

$$[a_{up_r}, a_{up_p}] = f(b_1, b_2, ..., b_n)$$
wherein: $b_1, b_2, \ldots b_n$ represent vertices of an $N$-sided polygon, $f$ represents the connectivity algorithm applied to the vertices, and $a_{up}$ is an auxiliary point derived from the algorithm.

In an embodiment of the second aspect, the connectivity data is derived using a generalised relationship:

$$|a_{up}, a_{up}| = f(b_i, b_j, b_k)$$

wherein: $b_1, b_2, b_3$ represent vertices of a triangle, $f$ represents the connectivity algorithm applied to the vertices, and $a_{up}$ is an auxiliary point derived from the algorithm.

In an embodiment of the second aspect, the connectivity algorithm applied is one of a centroid algorithm and a bisector algorithm, although it will be appreciated that any appropriate reversible connectivity algorithm can be used.

In an embodiment of the second aspect, the method comprises a step of including information that is indicative of a depth of a plurality of vertices of the three-dimensional image data in the two-dimensional data set.

In an embodiment of the second aspect, the step of including the vertex depth information comprises including first vertex information associated with the three-dimensional data with respect to a first viewpoint, and including second vertex information associated with the three-dimensional image with respect to a second viewpoint.
In an embodiment of the second aspect, the first vertex information is encoded in a first set of pixel data in the two-dimensional image, and the second vertex information is encoded in a second set of pixel data in the two-dimensional image.

In an embodiment of the second aspect, the method comprises a step of utilising a correspondence parameter to provide a correspondence between the first viewpoint and the second viewpoint.

In an embodiment of the second aspect, the correspondence parameter is represented as an integer value.

In an embodiment of the second aspect, the correspondence parameter is represented as a single bit in a string of zero values.

In an embodiment of the second aspect, the second viewpoint is selected by performing a step of analysing the three-dimensional data and the vertex depth information.

In an embodiment of the second aspect, the method further comprises a step of compressing the sets of pixel data prior to transmission of the data.

In an embodiment of the second aspect, the method comprises a further step of encoding the vertex depth information in a one-dimensional array.

In an embodiment of the second aspect, a closest point algorithm is utilised to find a predicted and actual
position in the second viewpoint. The closest point algorithm may be utilised to find a predicted and actual position along a scanline in the second viewpoint.

In accordance with a third aspect of the present invention, there is provided a system for processing information that is indicative of a shape in three-dimensional space, the processed information being usable in displaying a representation of the shape, the system comprising a processor arranged to:

- project a plurality of vertices of the shape onto a plane; and
- determine reference information that is usable to associate the projected vertices with each other, the reference information being determined by applying a reversible function to the projected vertices;
- wherein information that is indicative of the projected vertices and the reference information can be used to represent the shape on a display.

The reference information may be indicative of a point in two-dimensional space.

The point in two-dimensional space may be bound by its associated projected vertices.

The reversible function may be one of a centroid method or a bisector method, although it will be appreciated that any appropriate reversible function can be used.

In one embodiment, the processor is arranged to apply a compression algorithm to compress the information that is indicative of the projected vertices and the reference
information.

The shape may be one of a plurality of shapes that comprise a representation of an object in three-dimensional space, and the processor may be arranged to:

project a plurality of vertices of each shape onto the plane; and

for each shape, determine reference information that is usable to associate the projected vertices of the shape with each other, the reference information being determined by applying a reversible function to the projected vertices of the shape;

wherein information that is indicative of the projected vertices and the reference information can be used to represent the shapes that comprise the object on a display.

In accordance with a forth aspect of the present invention, there is provided a system for processing three-dimensional image data, the system comprising a processor arranged to:

process the three-dimensional image data to provide a data set for a corresponding two-dimensional image; and

add at least one set of pixel data to the two-dimensional image data set, the at least one set of pixel data comprising additional data that is representative of depth information of the three-dimensional image data.

In an embodiment of the fourth aspect, the at least one set of pixel data is a bit plane incorporated into the two-dimensional image data.

In an embodiment of the fourth aspect, the bit plane is a matrix, wherein each element in the matrix represents a
pixel of the two-dimensional image data.

In an embodiment of the fourth aspect, each element in the matrix that represents a vertex of the three-dimensional image data is represented by a first value, and each element in the matrix that does not represent a vertex of the three-dimensional image data is represented by a second value.

In an embodiment of the fourth aspect, the first value is an integer value of one (1) and the second value is an integer value of zero (0).

In an embodiment of the fourth aspect, the processor is further arranged to determine vertex information that is indicative of a two dimensional position of a plurality of vertices of the three-dimensional data.

In an embodiment of the fourth aspect, the processor is arranged to determine a connectivity between each vertex utilising a connectivity algorithm and to encode the connectivity data in an additional set of pixel data.

In an embodiment of the fourth aspect, the at least one set of additional pixel data is an additional bit plane incorporated into the two-dimensional image data.

In an embodiment of the fourth aspect, the bit plane is a matrix, wherein each element in the matrix represents a pixel of the image information.

In an embodiment of the fourth aspect, the connectivity data is derived by traversing a triangulated irregular
network to encode a single bit for each triangle, each single bit being representative of a connectivity point in the connectivity data.

5 In an embodiment of the fourth aspect, the connectivity data is derived using a generalised relationship:

\[ [a_{up}, a_{up}] = f(b_1, b_2, \ldots b_N) \]

wherein: \( b_1, b_2, \ldots b_N \) represent vertices of an \( N \)-sided polygon, \( f \) represents the connectivity algorithm applied to the vertices, and \( a_{up} \) is an auxiliary point derived from the algorithm.

In an embodiment of the fourth aspect, the connectivity data is derived using a generalised relationship:

\[ [a_{up}, a_{up}] = f(b_1, b_2, b_3) \]

wherein: \( b_1, b_2, b_3 \) represent vertices of a triangle, \( f \) represents the connectivity algorithm applied to the vertices, and \( a_{up} \) is an auxiliary point derived by the algorithm.

Detailed Description of the Drawings

Notwithstanding any other embodiments that may fall within the scope of the present invention, an embodiment of the present invention will now be described, by way of example only, with reference to the accompanying figures, in which:

Figure 1 is flow chart illustrating a method embodiment of the present invention;

Figure 2 is an example of a projection of a three-dimensional image onto a two-dimensional surface;
Figure 3 is an example of three-dimensional vertices projected onto a two-dimensional pixel grid at varying resolutions;

Figure 4 is a graph illustrating a relationship between geometric error and resolutions;

Figure 5 is a diagram illustrating the relationship between the vertex plane and the connectivity plane;

Figure 6 is an example of the centroid method, which is utilised to use the average of the three triangle vertices to make a representative auxiliary point;

Figure 7 is a graph illustrating the relationship between the percentage of triangles (vertices) that can be encoded using the centroid function for a given range of resolutions;

Figure 8 is an example of the bisector method, which is utilised to make a representative auxiliary point;

Figure 9 illustrates two examples illustrating how the bisector validates and invalidates reconstruction candidates;

Figures 10 (a) to (c) provide examples of encoded images;

Figure 11 (a) and (b) are plots illustrating the data size as a function of resolution and a compression ratio as a function of the resolution, respectively;
Figures 12 (a), (b) and (c) are illustrations of a rendered images (Figure 12(a) is a default render, Figure 12(b) is the render when viewed from a notional "camera one" and Figure 12(c) is a render when viewed from a notional "camera two". The darkest part of the line represents the centre of the cuboid, whereas the grey part of the line represents the location of camera one).

Figure 13 is a diagram illustrating the largest change in the size of the second camera projection, which occurs when the second projection direction is perpendicular to the plane of the hypotenuse of the view volume;

Figure 14 (a) represents a first view which shows one point a₁, and the two extrapolated points necessary to create a 3D plane;

Figure 14 (b) illustrates a second view which shows the second point used to create a ray beginning from b₁ in the viewing direction;

Figure 15 illustrates that a ray plane intersection can be used to determine a depth of the vertex v, wherein the plane point p and normal n may then be computed using the first view, b₂-b₁.

Figure 16 illustrates a unitised cuboid of geometric data from which five test points are chosen on each of three depth planes (z = 0; z = 1/2; z = 1);

Figure 17 is an illustration of re-projecting a vertex with the whole depth range in using x₁ and x₂ separately when compared to \( dx = |x₂-x₁| \):
Figure 18 (a) and (b) illustrate three scenarios for sending depth information, namely "unchanged", "pixel coordinate in second view" and "difference between the pixel coordinate of view one appearing in view two";

Figure 19 illustrates an array which utilises uniform quantisation for a sparse bit plane;

Figure 20 illustrates a non-uniform quantisation for a sparse bit plane; and

Figure 21 is an example implementation of a system in accordance with an embodiment of the present invention.

**Detailed Description of Embodiments of the present invention**

Figure 1 is a flow chart illustrating a method 100 of processing information that is indicative of a shape in three-dimensional space in accordance with an embodiment of the present invention. The processed information is usable in displaying a representation of the shape. The shape may be the shape of a portion of an object. The surface of a three-dimensional object may be approximated by network of geometrical shapes that have vertices, such as polygonal shapes. The method comprises in this embodiment the initial step 102 of projecting a plurality of vertices of geometrical shapes that are defined in three-dimensions onto a plane. Reference information is determined for each group of vertices associated with a
respective shape (step 104). The reference information is usable to link the projected vertices associated with each shape with each other. The reference information is determined by applying a reversible function to the projected vertices. The information that is indicative of the projected vertices and the reference information can be used to represent the shape on a display.

The following will describe the decomposition of three-dimensional Image data in more detail. The compression of any type of image firstly includes the decomposition of the geometric figures that make up the image into 'primitives' (i.e. a mathematical representation of each geometric figure). Primitives are the "building blocks" which are used to represent three-dimensional objects. In fact, modern graphics hardware is optimised to render primitives. In the rasterisation rendering process, the three-dimensional primitives may undergo several transformations, before finally being projected onto a screen surface (in a two-dimensional coordinate space).

To minimise data transmission, only a view-dependent geometry is considered, i.e. geometry which makes a contribution to the final rendered image. By considering view-dependent geometry only all redundant geometry that is not going to be rendered by being outside the viewing volume is removed, thus reducing the total amount of data required for rendering of the final image on the screen surface.

The focus at this level of decomposition arises from the ease of separating computation. The vertex shader program in modern day graphics cards operates on every incoming
- 16 -

vertex independently of one another. Projection is a necessary operation applied to each vertex. Figure 2 illustrates the projection of triangular shapes 202 and 204 into a plane 206. It is usually required to maintain all nine parameters (three vertices of each triangular shape in three dimensions) together at a given time in order to encode the relationship of each triangle.

A camera matrix $C$ contains all the information needed for this projection operation to transform coordinates from the world coordinates $(x; y; z)$ to the homogeneous coordinates $(wx; wy; wz)$:

$$[wx, wy, wz] = [x, y, z]C$$

In this regard, the term "camera" as referred to herein includes a virtual camera position which is used to reference a point, vertex or vector with a relationship to the rendering, display or rasterization of a three dimension image. This projection reduces the required number of bits to represent any given vertex parameter (coordinates). For example, consider the three-dimensional vertex with three 32 bit floating point numbers given by $(x; y; z)$. The resolution of the volume can be defined as the minimum difference between any two floating point numbers such that:

$$x + \Delta x = \xi$$

Once a camera view volume has been established as a sub-volume smaller than the entire three-dimensional volume, all relevant geometry will change the coordinate set to be
referenced relative to this view volume. As such, the coordinates are still 32 bit floats, but the represented region is now much smaller and:

\[ wx + \delta wx = \varepsilon \]

remains the smallest unit of measure. Therefore, the number of bits per unit volume in the projective space is over-representative of the original three-dimensional space.

The projection serves a secondary purpose, namely to convert the coordinate set of three-dimensional point data such that \( wx;wy \) represent screen coordinates and \( wz \) is the distance from the given camera position (depth). This serves as the basis for representing the vertices. All vertices are transformed to projective space and their screen coordinates \( (wx;wy) \) are saved using a fixed resolution screen surface. As for each vertex, the corresponding pixel is activated.

It is assumed that the screen remains at a fixed resolution, which allows the coordinates on the screen to be quantised as \( (wx';wy') \). This has implications when reversing the projection process, as the transformation with these coordinates and \( wz \) will yield an approximation to the original data \( (x';y';z') \).

By working at the primitive level and using projection operations described, several advantages are gained. For example, no prior conditions are required as a precaution to the encoding or decoding process. In another words the process is not dependent on an initial state. Further,
other methodologies utilised to compress, transmit and render three-dimensional images may require the client and server to synchronise their hierarchical scene description tree so that updates can be minimised. Such a vertex encoding method does not require any prior shared information. Such state tracking is a common problem among sharing three-dimensional graphics, as it is often needed to determine the modes of rendering and current matrices for transformation. Spatial repetition in the three-dimensional data is also minimised through projection. The common coordinates of the vertex data defined for a view volume is discarded as projection makes coordinates relative to the view volume. (The camera matrix describes where the view volume begins in three-dimensional space). Alternative mesh compression techniques such as triangle stripification or mesh splitting maintain dependencies in the encoding and/or decoding process may also be considered.

As an example a “bunny” is used to perform a single rendering pass to obtain the visible geometry. Figure 3 shows the of bunny’s vertices three progressive resolutions 300, 302 and 304 as they are projected onto a two-dimensional grid (128x128, 200x200, 400x400). As the resolution increases, the number of vertices remains near constant, however, the purpose is to increase the accuracy of each vertex’s respective (x;y) coordinate. In practice, more zeroes are provided in the image which lends itself to better compression ratios.

In this example, the image upon which the vertices are projected onto will be referenced as the vertex plane. It is a monochrome image where each lit pixel represents the
projected vertex. The associated geometric error by way of this reversal process is directly related to the quantization of screen coordinates. Higher resolutions and lower depths will ensure accuracy loss is minimised. Plot 320 shown in Figure 4 demonstrates this relationship of geometric error with steadily increasing resolutions.

To represent the relationship between three vertices to form a triangle, the method in accordance with an embodiment of the present invention uses only a single bit that is to be stored in a second bitplane termed the connectivity plane. During the encoding process, the three points of the projected triangle are provided as input \((b_1; b_2; b_3)\) from the vertex plane. Based on this data, a function \(f(b_1; b_2; b_3)\) outputs a two-dimensional pixel location on the connectivity plane to represent the relation:

\[
[a_{uv_1},a_{uv_2}] = f(b_1,b_2,b_3)
\]

This is known as the auxiliary point and when applied to the inverse of \(f\) with the entire vertex plane the three vertices being represented can be determined:

\[
[b_1,b_2,b_3] = f^{-1}([a_{uv_1},a_{uv_2}]_{\text{vertex plane}})
\]

Figure 5 illustrates three vertices 332, 334 and 336 of a triangle in the vertex plane, and the corresponding auxiliary point 338 is generated based on those three points and a given function. If considering planar triangulations, choose a suitable function to utilise the locality of points to our advantage in the encoding.
process.

The encoding function $f$ should be reversible, should minimise collisions for a given resolution, and when the reverse of the function occurs, it should converge (search deadline). A function of this kind will exhibit properties that favour an expected class of triangles for encoding.

One such example function is the computation of the centroid of the triangle. The raw three-dimensional triangle data of three vertices $v_1; v_2; v_3$ each have three parameters $v_i = [x_i, y_i, z_i]$. A projection of each vertex produces a vertex in window coordinates $wv_i = [wx_i, wy_i, wz_i]$. This is quantised (via down sampling) to fit the chosen resolution of the bit plane

$$b_i = wx_i^* wy_i^*.$$  

The $z$ coordinate is discarded, as the connectivity in two-dimensional space has the same vertex references in the three-dimensional space. The centroid is computed for the auxiliary point:

$$aup = \frac{b_1 + b_2 + b_3}{3}, or$$

$$aup = \left[ \frac{1}{3} \sum_{i=1}^{3} wx_i^* \frac{1}{3} \sum_{i=1}^{3} wy_i^* \right]$$

The connectivity plane flips the bit in the corresponding pixel coordinate $aup$, which is illustrated by the diagram shown in Figure 6.

The decoding process typically still requires more information. The input data for the decoding process are data that relate to the vertex plane and the auxiliary
point. It is usually not known which vertices are candidates for the representative relation, thus there are some rules which are followed during the encoding process.

For the centroid method, not all vertices from the vertex plane are tested, only those within a certain distance of the auxiliary point. The encoding algorithm requires computing the centroid of all local triplets to ensure that the most correct triplet will be closest to the given auxiliary point and thus will be guessed first.

Comparing triplets becomes $O(n^3)$ in complexity, however several optimisations are available to restrict the total amount of computation required. The search distance may have a threshold based on projected triangle sizes, the search size may be limited by ordering candidate vertices by distance from the auxiliary point. Balancing maximum and minimum search sizes later is taken into account when considering if all triangles may be encoded using the single function. These rules ensure that no false recoveries occur. Where the projected geometry remains consistently planar, this technique has generous performance for compression efficiency.

The number of possible auxiliary points that can be generated by this method are exactly $\binom{n}{3}$ for $n$ vertices. Where a neighbourhood of vertices small and dense, the number of generated auxiliary points will saturate region and the result is ambiguous.

Plot 350 shown in Figure 7 illustrates the percentage of triangles that can be encoded versus the given resolution. At low resolutions, vertices will appear more equidistant
to the auxiliary point and thus will be falsely guessed as part of the relation. As the resolution increases, so does the sparseness of the vertices and hence their uniqueness with distance to the auxiliary point. The second resolution increase in the connectivity plane will allow for more precise centroid coordinates. There is a limit however, on how much higher the connectivity plane resolution can be with respect to vertex plane resolution for an effect to be seen.

Another function that takes advantage of planarity in meshes uses the bisector of the triangle. As schematically indicated by the illustration of Figure 8, three points may define a triangle. A first is \( b_j \) and the remaining two points \( b_i; b_l \) will generate a midpoint \( \text{mid} \) where the bisector can be drawn from \( b_j \). The auxiliary point (\( \text{aup} \)) created is in the example the halfway point of this bisector:

\[
\text{mid} = b_i + \frac{1}{2}(b_l - b_i) \\
\text{aup} = b_j + \frac{1}{2}(\text{mid} - b_j)
\]

The recovery process depends more on the choice of \( b_j \). Given \( \text{aup}, b_j \) is expected to be the closest point, from there an estimated bisector is constructed and the estimated midpoint \( \text{mid}^* \). The remaining process is a search for two vertices that are near equidistant from the approximated midpoint \( \text{mid}^* \). In accordance with an embodiment of the present invention the conditions for this function are that:

- the closest vertex to this point is any vertex
on the triangle to be encoded \((b_3)\); 
- the auxiliary point is aligned along the 
bisector of such a triangle \((aup)\); 
- the point that is extrapolated along the 
bisector is equidistant; 
- from the other two points on the triangle 
\(\text{approx. mid}\).

Increasing the performance begins with the first 
condition, closest vertex will become \(b_1\). This leads to 
the second performance consideration, the alignment of the 
bisector such that the correct midpoint will be recovered. 
The best choice of \(b_1\) for the latter case is going to 
choose the longest bisector. This is due to low 
resolutions and limited pixels used in approximation; the 
more that are available the efficiency improves. The 
trade off now becomes choose the closest distance, or the 
longest bisector - both cannot be maximised as the longest 
bisector also has greatest distance.

Generally, experimentation shows that choosing the closest 
point provides a better result. Due to the equations 
required to reach the auxiliary point the connectivity 
plane alone can be increased up to a factor of four to 
benefit from increased accuracy of the vector (a factor of 
four results from two divisions of two).

This function creates a greater limitation on possible 
invalid encodings as the remaining vertex pair \(b_1; b_2\) must be 
approximately equidistant. Figure 9 illustrates two cases 
360 and 365, the first case 360 which consists of a circle 
of points that form valid encoding whereby valid \((b_1; b_2)\) 
points appear on the circle (with a small degree of error)
and would also hold a minimum distance between them so they are not considered a chord. The false encoding is illustrated by unequal distances, this latter case is quite common and helps improve the effectiveness of encoding.

The unencoded triangle data are the horizon edges. The number of pixels which they occupy is too few to encode information embedded into the image. This is why the percentage of encoded vertices slowly increases at greater resolutions, the horizon edges are becoming detailed enough for the compression algorithm to work. Figure 10 shows a sequence of various resolutions of the bunny model.

Several more variation of this method are possible, the auxiliary point can be located a factor \(\frac{1}{3}\) instead of \(\frac{1}{2}\) from the midpoint, making it much closer to \(b_3\). The auxiliary point could also be shifted a predefined number of pixels from its intended location. These variations allowed the function to apply to different classes of triangles, namely they are skewed with one small angle and generally small area.

In one embodiment, once one or more of the above steps have been completed, the data describing each of the vertices, including vertex information and their relationships to draw or render the three dimension image on a two dimension image plane may then be compressed within an image file of a two dimensional format. Examples of the compression algorithms and their relative performance for the bit planes are now described. As described above, the resolution chosen will influence the
error rate, and thus the size of the compressed format. JPEG does not manage to achieve great efficiency as it remains best suited to representing continuous tones. PNG offers increased improvement over JPEG, however its prediction stage cannot approximate the sparse data of larger resolutions very well, it uses the zlib library which offers similar algorithms to that of gzip. GIF is very capable of managing a monochrome image bzip2 however surpasses each method with the smallest compressed size. The highest compressing algorithms also require the longest waiting time.

Figure 11 (a) shows a plot 370 illustrating the size (in bytes) as a function of resolution and Figure 11 (b) shows a plot 372 illustrating the compression ratio as a function of the resolution. The PNG image format uses a baseline comparison and the GIF demonstrates that it does not match the rate at which the uncompressed data grows. However, bzip2 can demonstrate how increasing the sparseness of the image can be efficiently encoded. The additional overhead of injecting zero values have shown to added no noticeable overhead.

Once the vertex information and their relationships to render a three dimensioned image are encoded, the encoded file may then be transmitted for further processing so that the necessary vertex information and its relationship may be recovered to render the encoded three dimension image. This embodiment of transmitting three dimensional image data is advantageous in that known methods of sending three dimensional image data is currently not bandwidth scalable. In another words, the amount of geometric-complexity of the image results in a
proportional cost of bandwidth. However, embodiments described herein may not require the same proportion of bandwidth for increased complexity.

In one embodiment, vertex information may be encoded using two cameras arranged as a stereo pair. In this embodiment, the first camera retains the original viewpoint of the image being rendered, while the second is transformed relative to the first camera such that it forms a stereo pair.

There is a given flexibility in choosing camera parameters for the purposes of acquiring the best depth estimation. The previous sections describe the enlarging of the viewpoint in the second view to have all data projected onto the screen surface. However, when viewing the data using a stereo camera pair, the second camera may require its view to be scaled to contain all the data seen from the first view. This may be known as the scaling parameter "s". This scaling camera parameter s is investigated to demonstrate how it can be chosen to yield a better translation of depth coordinate mappings from the k pixel line for a given rotation value θ. Figure 12 (a) illustrates a default render 380, Figure 12 (b) represents the image 385 as viewed from notional camera one and Figure 12 (c) represents the image 390 from notional camera two.

The darkest part of the line is the center of the cuboid, the grey part is the location of camera one. There are all data points contained within the cuboid that need to be visible in second view. One can either change the camera location, or change the shape of the camera. Changing the location by increasing the distance to the
points, the projected points condense into a smaller area. This only changes the level of resolution required and can vary greatly depending on how large the dynamic depth range is. Reshaping the camera via the projection matrix, would provide for a larger area to project onto without changing the distance to the point. Consequently, only the horizontal axis requires scaling. The extent of scaling required depends on both the dynamic depth range and the rotation of the second camera.

When rotating from the front view with projection axis [-1;1], the largest re-projection in second view occurs at the hypotenuse of the triangle formed by the horizontal screen width and maximum depth, as shown in Figure 13. Using object space coordinates are of relevance to compare untransformed depth data. The horizontal screen width distance can be obtained for the view at 0 rotation and XZ diagonal distance a, where

$$\tan(\alpha) = \frac{Z}{W}$$ (Znon-normalised)

The $0^{\circ}$ and $\alpha$ views represent lower and upper bound for change in screen size $W_2$.

In such a case, the numbers of pixels required for the $\alpha^{0}$ view are:

$$H_{pix} = \frac{W}{\cos(\alpha)}$$

the width, in pixels for $0 < \alpha$ view is:

$$W_2 = \left(H_{pix} - W_1\right) \times \frac{\theta}{\alpha} + 1$$

For the length of the view volume in the Z direction the number of pixels are referred to as:
the pixels for \( \theta > \alpha \) view is:

\[
W_2 = (Z_{pix} - H_{pix}) \cdot \frac{\theta - \alpha}{\pi} + \left( H_{pix} - W_1 \right) \cdot \frac{\theta}{\alpha} + 1
\]

5 The new pixel scale for the new projection point becomes 
\((W_2 + W_1)/W_1\).

Reducing this back to projective coordinates for \( \theta < \alpha \):

\[
s = \left( \frac{1}{\cos(\alpha)} - 1 \right) \cdot \frac{\theta}{\alpha} + 1
\]

10 for \( \theta > \alpha \):

\[
s = \frac{1}{2} \left( \frac{1}{\cos(\alpha)} - 1 \right) \cdot \frac{\theta - \alpha}{\pi} + \left( \frac{1}{2} \left( \frac{1}{\cos(\alpha)} - 1 \right) \cdot \frac{\theta}{\alpha} + 1 \right)
\]

Reducing this yield:

\[
s = \frac{1}{2} \left( \frac{1}{\cos(\alpha)} - 1 \right) \left( \frac{\theta \pi}{\alpha} - \alpha^2 \right) + 1
\]

15 The new coordinates need to divide the extending length \( W_2 \) by \([-s, s]\) to be represented in the same space as the first projection. This is applied to the definition of near and far depth planes of the projection also via the XZ plane. The boundary conditions to be considered here is the angle is based on the depth and width of the 3D volume of data points.

The parameter \( a \) greater than zero and smaller than \( \pi/2 \) otherwise this volume has no depth or width. The best angle \( \theta \) is dependent on the way in which the data is organised and the depth resolution from the bit plane.

The depth value can be recovered by performing
intersection tests in the world space. The values need to be transformed from the projective coordinate for each respective view. The intersection testing in this space can take many forms, for a reference solution, a 3D line to 3D plane test is used here. It is also possible to perform a 2D intersection test if the projected coordinates from one view are transformed into another.

The 2D line intersection test is possible as there is no dependence on the vertical axis of either screen. Figures 14 (a) and (b) show how the plane and vector are defined. In view one (Figure 14 (a)), a₁ is the point obtained from the vertex bit plane. The points a₂ and a₃ are simply translated to the extremes along the vertical and depth axes of the view, such that the three points form a 3D plane. In view two (Figure 14 (b)), the point b₁ is also obtained from given data. The second point b₂ is chosen such that b₂-b₁ is a vector parallel to the viewing direction.

After transforming all points to world space the plane, a normal n can be defined and the point on the plane p can be any of the original points a₁, a₂, a₃. The vector b₂-b₁ is also required in normalised form.

As the vertex v exists somewhere on the plane, the vector (v-p) is perpendicular to the normal, thus:

\[ n \cdot (v-p) = 0 \]

In the second view, the ray equation that passes through v begins from b₁ and moves along the vector (b₂-b₁) for some time t:

\[ v = b₁ + t(b₂-b₁) \]
Resolving these two equations gives:
\[ n \cdot (b_1 + u(b_2 - b_1)) = n \cdot p \]
\[ t = \frac{n \cdot (p - b_1)}{n \cdot (b_2 - b_1)} \]

The representation of depth using stereo images can offer potentially greater accuracy than simply sending the depth value alone. Stereo depth is a function of two variables, that is, the pixel position in image 1 and the corresponding pixel in image 2. As there is no change in height between these pixels they will be denoted \(x_1\) and \(x_2\), where \(x_1\) is in the set of \([0; W_1]\) and \(x_2\) is in the set of \([0; W_2]\). The case \(W_2 = W_1\) occurs where there is only a single depth value, otherwise \(W_2 > W_1\) is always true as the depth range increases.

For a given \(x_1\), the value of \(x_2\) will be dependent on both \(x_1\) and the depth value, which is illustrated in Figure 15. For each point, the re-projected pixel value was found for the second camera using a second camera \(\theta = 25\).

The geometric screen widths of each camera is different depending on the depth range and rotation \(\theta\). For non-normalised coordinate range used, the ratio \(W_2/W_1\) can be considered the scaling factor of the range with which this second coordinate will appear. In the instance presented \(W_2/W_1 = 1:174143\).

Figure 16 shows the ranges resulting from the re-projection of the vertex in the second view for all depths.

If only the \(x_1\) and \(x_2\) coordinate were sent, then for \(k\) bits
used in $x_2$, only $2^k$ unique depth values can be reproduced on the receiving side. By sending $x_1$, $dx$, the part of $x_2$ is already known from $x_1$, so only the difference is required to be encoded.

Figure 17 is an illustration of re-projecting a vertex with the whole depth range in using $x_1$ and $x_2$ separately when compared to $dx = |x_2 - x_1|$. Figure 18 (a) and (b) illustrates three scenarios for sending depth information, namely “unchanged”, “pixel coordinate in second view” and “difference between the pixel coordinate of view one appearing in view two”. The quantisation with $dx$ results in finer selection of depth values.

The relationship between $x_1$ and $x_2$ can be found in the equations used to derive each of the final values. Given a vertex $v$ to be transformed, the two camera instances carry out the following transformation:

$$w_i = P_i M_i v, w_2 = P_2 M_2 v$$

Here $P_1$, $P_2$ are the projection matrices and $M_1$, $M_2$ are the model view matrices. This method adjusts the second camera using the first.

In the second camera view, the projection matrix width is enlarged as to keep all the data visible, but only in horizontal axis. This corresponds to a scaling operation such that:

$$P_2 = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times P_1$$
s here is computed from the data using the equations previously described.

For the model view matrix, even as the rotation of the camera about the camera's up vector remains an arbitrary axis, this can still be represented in the normalised domain as a rotation about the y axis $R_y$:

\[
M_2 = \begin{bmatrix}
\cos \theta & 0 & \sin \theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

These scaling and rotation operators applied to the existing transformation are both linear transformations, as such the second projection also maintains a linear relationship with the first. This explains the linear dependence between the two variables.

There are numerous methods to gain the best compression result for a given set of floating point. A sparse bit plane of the depth values used is presented for reference. Uniform quantisation is a convenient method to achieve an image based representation with the bandwidth budget ($w \times h$ bits) of a sparse bit plane. The data can be transcoded from $k$ pixels = $2^k$ unique values to $k$ pixels = $k$ unique values. A depth value that has been quantised to the range $[0;16]$ (24 4-bit equivalent) will have reserved 16 pixels in the bit plane image. This representation is referred to as a pixel vector. For uniform quantisation, the sum of all quantised ranges should be close to the $w \times h$ value such that for $n$ points, the number of pixels are:

\[
n \times k \leq w \times h
\]
Using this technique, a sparse bit plane results, where each depth value corresponds to a single bit in the bit plane. While various orderings are possible, the values are encoded in order of scanline appearance such that they can be referenced to the vertex whose depth is being encoded.

Figure 19 shows an example of encoding the depth value as a sparse image using uniform quantisation. The image has resolution 8x5 = 40, for encoding six points, there are at most 40/6 = 6 bits available to use to encode a single depth value. Six (6) bits represent the amount of quantisation.

While the example shown in Figure 19 has many zeroes, the zeroes that appear after the one are considered redundant. As each depth value is read sequentially in the block of 6 pixels, once the appearance is found, the rest of the block of six pixels will always have zero values.

Figure 20 shows the result of removing redundancy. This method of non-uniform quantisation can achieve greater use of the available bits in the image such as to increase the number of pixels allocated. While the number pixels for each value remains data dependent, a summation of all pixels required can be expressed as:

\[ \sum_{i=1}^{N} v_p < w \times h \]

\( v_p \) represents the number of pixels required to represent the value \( v \). For a given quantisation level \( k \), the maximum required is as stated before, however under a uniform distribution of values from \([1;k]\), there will be at least
N/2 values with less than k/2 pixels required for representation.

By packing more values in the image, the bit plane will become less sparse, however, the information content per bit will be much higher. Should the values have bias for upper end values, it is simple to invert the sequence and signify this change as an additional single bit. Compression strategies such as Run Length Encoding can be applied to further reduce the size of the representation of depth data.

The uniform quantisation method can be applied without requiring a pass of the depth value data. In contrast, the non-uniform quantisation method will require a single pass for summation of the depth values. The optimal level of quantisation may require multiple passes to obtain for non-uniform distributions. A quantisation level can be pre-selected by using an upper limit estimation.

If the values are all normalised real numbers in the range [0;1] and uniformly distributed, then the frequency of occurrence is equal, then as values have a cost associated proportionally to their size, the area under this cost curve is the amount of space required. By scaling the cost axis, the quantisation level can be obtained in a simple manner:

$$\frac{\sum_{v=0}^{N} v}{N}$$

corresponds to the proportion of area used in the bit plane, where N is the number of values. For these N values to fit in the image, the quantisation is:
\begin{equation}
k = \frac{N-1}{\sum_{i} w \times h} \times \frac{N}{N}
\end{equation}

N-1 is used for slight over estimation.

Resolution Scalable Depth Representation

Sending the pixel difference \( dx = |x_2 - x_1| \) has no significant advantage than sending the depth \( w_{z1} \) as it is, in fact, with truncation errors of conversion, it has slightly worse recovery to \( w_{z1}^* \). The range that needs to be covered on the scanline covers the same range for a given level of quantisation. Instead, the pixel difference is encoded implicitly, by using the two vertex planes in both views, the median \( w_z \) value, and the closest point algorithm along the scanline.

The vertex bit plane is constructed using the first view, having coordinates for each vertex \( w_{x1}, w_{y1} \). The depth median \( w_{z\text{median}} \) is computed for all vertices. Using \( (w_{x1}, w_{y1}, w_{z\text{median}}) \), the corresponding projected points in the second view \( (w_{x2}^*, w_{y2}, w_{z2}^*) \) are obtained. The real projection points are found using \( (w_{x1}, w_{y1}, w_{z1}) \), as \( (w_{x2}, w_{y2}, w_{z2}) \). In the preceding section describes only \( w_{z2}^* \) being sufficient to dictate the depth, with \( w_{y2} \) having no change in the second view. There is now a real and guessed (as median) values for depth, as \( w_{x2}, w_{z2} \) respectively. The second vertex bit plane is used here, the point \( (w_{x2}^*, w_{y2}) \) is used as a seed location in the plane and a search for the closest point, along the scanline, produces a ranking of vertices ranked in order of distance. The correct depth will be of one of the vertices in the second bit plane, so the search space is of all the vertices in the scanline.
The rank for each vertex, in order of visit from the first bitplane, is saved in a 1 dimensional array to then be encoded as is, or as a bit plane. The remote site will receive, the two vertex bit planes for each view, the guess depth value \( w_{\text{median}} \), and a 1D array of ranks from the closest point algorithm.

The encoding of points in this fashion avoids the cost of explicitly encoding distance quantities, rather, the second bit plane contains depth information along the scanline.

This can be scaled horizontally with the same number of vertices, bits, essentially increasing the sparseness and hence accuracy of each vertex. The correct depth value is the furthest point, last of the closest points, requires encoding the value 6. For any given resolution this does not change, contrast this to the explicit distance \( dx \), which requires more bits to accurately specify the vertex along the scanline.

The median \( w_z \) value can also be changed, as the closest point can be referenced from anywhere along the line. The median has the least bias to any other point, the average however, would give lower rank distance to more frequently occurring depth values, meaning there are more bits required to represent the few.

In one example, the encoding algorithm for depth information may be summarised as follows:

1. Orthographically project 3D vertex data to window coordinates;
2. Resize orthographic projection such that all data fits into screen (use data lowest/highest Z);
3. Choose rotation angle θ and move camera accordingly as the second view;
4. Orthographically project actual vertices using the second camera view;
5. Orthographically project data with median depth value using the second camera view;
6. For each point, use the closest point algorithm to find the rank between the predicted and actual position along the scanline in the second view;
7. The 1D array of ranks from the depth representation are converted to a pixel vector with one bit active; and
8. Each pixel vector is placed in the third bit plane in the predefined order.

In one example, the decoding algorithm may be summarised as follows:

1. Obtain vertex bit plane of camera view 1 (w_{x1}, w_{y1});
2. Obtain camera parameter θ, and scaling factor s;
3. Obtain w_{zguess}, and compute the guessed depth bit plane using the second projected view (w_{x2}, w_{y2}, w_{zguess});
4. Obtain the vertex bit plane for the second view (w_{x2}, w_{y2});
5. Obtain the rank data for depth values;
6. Use iterative closest point, the guessed depth bit plane and rank data to determine the mapping from (w_{x1}, w_{y1}) to (w_{x2}, w_{y2}); and
7. For each point solve the 3D line-plane intersection of: camera location 1 through pixel (w_{x1}, w_{y1}) of view 1; and camera location 2 through pixel (w_{x2}, w_{y2}) of view 2. (where w_{y1} = w_{y2}).
In an alternative example, the decoding algorithm may be summarised as follows:

1. Transform each vertex into projective components \((w_x, w_y, w_z)\) which may be projected onto a plane;
2. From a camera position, determine a median \(w_z\) (depth) coordinate amongst all of the vertices. This median may be referred to as \(w_{\text{median}}\);
3. For each vertex, use only the \(w_{\text{median}}\) coordinate whilst ignoring the \(w_z\) coordinate;
4. Convert each vertex with coordinates \([w_x, w_y, w_{\text{median}}]\) back into the world space;
5. Reproject to a second view, each of the vertices with its corresponding coordinates of \(w_{\text{median}}\) and \(w_z\) so that on each scan line of the one vertex, there are two points. The first point being based on \(w_{\text{median}}\) and the other point on \(w_z\);
6. Operate a closest point algorithm to determine the number of hops from the vertex using \(w_{\text{median}}\) to the vertex using \(w_z\). The rank, may then be ordered by the number of pixels away from \(w_{\text{median}}\).

Referring now to Figure 21 there is a shown a schematic diagram of a system for processing information that is indicative of a shape in three-dimensional space. The system comprises a server 500. The server 500 comprises suitable components necessary to receive, store and execute appropriate computer instructions. The components may include a processing unit 502, read-only memory (ROM) 504, random access memory (RAM) 506, and input/output devices such as disk drives 508, input devices 510 such as
an Ethernet port, a USB port, etc. Display 512 such as a liquid crystal display, a light emitting display or any other suitable display and communications links 514. The server 500 includes instructions that may be included in ROM 504, RAM 506 or disk drives 508 and may be executed by the processing unit 502. There may be provided a plurality of communication links 514 which may variously connect to one or more computing devices such as a server, personal computers, terminals, wireless or handheld computing devices. At least one of a plurality of communications link may be connected to an external computing network through a telephone line or other type of communications link.

The service may include storage devices such as a disk drive 508 which may encompass solid state drives, hard disk drives, optical drives or magnetic tape drives. The server 500 may use a single disk drive or multiple disk drives. The server 500 may also have a suitable operating system 516 which resides on the disk drive or in the ROM of the server 500.

The system has a software application 520 residing on a disk or other storage device which is arranged to compress three-dimensional image data for the purpose of transmission of the data to one or more remote computing systems.

Although not required, the embodiments described with reference to the Figures can be implemented as an Application Programming Interface (API) or as a series of libraries for use by a developer or can be included within
another software application, such as a terminal or personal computer operating system or a portable computing device operating system. Generally, as program modules include routines, programs, objects, components and data files assisting in the performance of particular functions, the skilled person will understand that the functionality of the software application may be distributed across a number of routines, objects or components to achieve the same functionality desired herein.

It will also be appreciated that where the methods and systems of the present invention are either wholly implemented by computing system or partly implemented by computing systems then any appropriate computing system architecture may be utilised. This will include stand alone computers, network computers and dedicated hardware devices, such as smartphones, tablet computing devices, or the like. Where the terms "computing system" and "computing device" are used, these terms are intended to cover any appropriate arrangement of computer hardware capable of implementing the function described.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.
Any reference to prior art contained herein is not to be taken as an admission that the information is common general knowledge, unless otherwise indicated.
- 42 -

THE CLAIMS:

1. A method of processing information that is indicative of a shape in three-dimensional space, the processed information being usable in displaying a representation of the shape, the method comprising the steps of:
   projecting a plurality of vertices of the shape onto a plane; and
   determining reference information that is usable to associate the projected vertices with each other, the reference information being determined by applying a reversible function to the projected vertices;
   wherein information that is indicative of the projected vertices and the reference information can be used to represent the shape on a display.

2. The method of claim 1, wherein the reference information is indicative of a point in two-dimensional space.

3. The method of claim 2, wherein the point in two-dimensional space is bound by its associated projected vertices.

4. The method of any one of the preceding claims, wherein the reversible function is one of a centroid method or a bisector method.

5. The method of any one of the preceding claims, further comprising the step of applying a compression algorithm to compress the information that is indicative of the projected vertices and the reference information.

6. The method of any one of the preceding claims,
wherein the shape is one of a plurality of shapes that 
comprise a representation of an object in three-
dimensional space, and the method comprises the steps of:
projecting a plurality of vertices of each shape onto
the plane; and
for each shape, determining reference information
that is usable to associate the projected vertices of the
shape with each other, the reference information being
determined by applying a reversible function to the
projected vertices of the shape;
wherein information that is indicative of the projected
vertices and the reference information can be used to
represent the shapes that comprise the object on a
display.

7. A method of processing three-dimensional image data
comprising the steps of:
processing the three-dimensional image data to
provide a data set for a corresponding two-
dimensional image; and
adding at least one set of pixel data to the two-
dimensional image data set, the at least one set of pixel
data comprising additional data that is representative of
depth information of the three-dimensional image data.

8. The method of claim 7, wherein the at least one set
of pixel data is a bit plane incorporated into the two-
dimensional image data.

9. The method of claim 8, wherein the bit plane is a
matrix, wherein each element in the matrix represents a
pixel of the two-dimensional image data.
10. The method of claim 9, wherein each element in the matrix that represents a vertex of the three-dimensional image data is represented by a first value, and each element in the matrix that does not represent a vertex of the three-dimensional image data is represented by a second value.

11. The method of any one of claims 7 to 10, comprising the step of determining vertex information that is indicative of a two-dimensional position of a plurality of vertices of the three-dimensional data.

12. The method of claim 11, comprising the step of determining a connectivity between each vertex utilizing a connectivity algorithm and encoding the connectivity data in an additional set of pixel data.

13. The method of claim 12, wherein the connectivity data is derived using a generalised relationship:

\[ [a_{up}, a_{up}] = f(b_1, b_2, \ldots, b_n) \]

wherein: \( b_1, b_2, \ldots, b_n \) represent vertices of an \( N \)-sided polygon, \( f \) represents the connectivity algorithm applied to the vertices, and \( a_{up} \) is an auxiliary point derived from the algorithm.

14. The method of claim 12, wherein the connectivity data is derived using a generalised relationship:

\[ [a_{up}, a_{up}] = f(b_1, b_2, b_3) \]

wherein: \( b_1, b_2, b_3 \) represent vertices of a triangle, \( f \) represents the connectivity algorithm applied to the vertices, and \( a_{up} \) is an auxiliary point derived from the algorithm.
15. The method of claim 13 or claim 14, wherein the connectivity algorithm applied is one of a centroid algorithm and a bisector algorithm.

16. The method of any one of claims 7 to 15, comprising the step of including information that is indicative of a depth of a plurality of vertices of the three-dimensional image data in the two-dimensional image data set.

17. The method of claim 16, wherein the step of including the vertex depth information comprises including first vertex information associated with the three-dimensional data with respect to a first viewpoint, and including second vertex information associated with the three-dimensional image with respect to a second viewpoint.

18. The method of claim 17, comprising the step of utilising a correspondence parameter to provide a correspondence between the first viewpoint and the second viewpoint.

19. The method of claim 17 or claim 18, wherein the second viewpoint is selected by performing the step of analysing the three-dimensional data and the vertex depth information.

20. The method of any one of claims 7 to 19, comprising the step of compressing the sets of pixel data prior to transmission of the data.

21. A system for processing information that is indicative of a shape in three-dimensional space, the processed information being usable in displaying a
representation of the shape, the system comprising a processor arranged to:

project a plurality of vertices of the shape onto a plane; and

determine reference information that is usable to associate the projected vertices with each other, the reference information being determined by applying a reversible function to the projected vertices;

wherein information that is indicative of the projected vertices and the reference information can be used to represent the shape on a display.

22. The system of claim 21, wherein the reference information is indicative of a point in two-dimensional space.

23. The system of claim 22, wherein the point in two-dimensional space is bound by its associated projected vertices.

24. The system of any one of claims 21 to 23, wherein the reversible function is one of a centroid method or a bisector method.

25. The system of any one of claims 21 to 24, wherein the processor is arranged to apply a compression algorithm to compress the information that is indicative of the projected vertices and the reference information.

26. The system of any one of claims 21 to 25, wherein the shape is one of a plurality of shapes that comprise a representation of an object in three-dimensional space, and the processor is arranged to:

project a plurality of vertices of each shape onto
the plane; and

for each shape, determine reference information that is usable to associate the projected vertices of the shape with each other, the reference information being determined by applying a reversible function to the projected vertices of the shape;

wherein information that is indicative of the projected vertices and the reference information can be used to represent the shapes that comprise the object on a display.

27. A system for processing three-dimensional image data, the system comprising a processor arranged to:

process the three-dimensional image data to provide a data set for a corresponding two-dimensional image; and

add at least one set of pixel data to the two-dimensional image data set, the at least one set of pixel data comprising additional data that is representative of depth information of the three-dimensional image data.

28. The system of claim 27, wherein the at least one set of pixel data is a bit plane incorporated into the two-dimensional image data.

29. The system of claim 27 or 28, wherein the processor is further arranged to determine vertex information that is indicative of a two dimensional position of a plurality of vertices of the three-dimensional data.

30. The system of claim 29, wherein the processor further determines a connectivity between each vertex utilising a connectivity algorithm and encodes the connectivity data in an additional set of pixel data.
31. The system of claim 30, wherein the processor is arranged to derive the connectivity data using a generalised relationship:

\[ \text{[aup},_i,\text{aup},_j] = f(b_1, b_2, \ldots, b_N) \]

wherein: \( b_1, b_2, \ldots, b_N \) represent vertices of an \( N \)-sided polygon, \( f \) represents the connectivity algorithm applied to the vertices, and \( \text{aup} \) is an auxiliary point derived from the algorithm.

32. The system of claim 30, wherein the processor is arranged to derive the connectivity data using a generalised relationship:

\[ \text{[aup},_i,\text{aup},_j] = f(b_1, b_2, b_3) \]

wherein: \( b_1, b_2, b_3 \) represent vertices of a triangle, \( f \) represents the connectivity algorithm applied to the vertices, and \( \text{aup} \) is the auxiliary point derived from the algorithm.

33. The system of claim 31 or claim 32, wherein the connectivity algorithm applied is one of a centroid algorithm and a bisector algorithm.

34. The system of any one of claims 27 to 33, wherein the processor is arranged to include information that is indicative of a depth of a plurality of vertices of the three-dimensional image data in the two-dimensional image data set.

35. The system of claim 32, wherein the processor is arranged to include the vertex depth information by including first vertex information associated with the three-dimensional data with respect to a first viewpoint,
and including second vertex information associated with
the three-dimensional image with respect to a second view
point.

36. The system of claim 33, wherein the processor is
arranged to select the second viewpoint by analysing the
three-dimensional data and the vertex depth information.

37. A computer program, comprising at least one
instruction for controlling a computer system to implement
a method in accordance with any one of claims 1 to 20.

38. A computer readable medium providing a computer
program in accordance with claim 37.

39. A communication signal transmitted by an electronic
system implementing a method in accordance with any one of
claims 1 to 20.

40. A communication signal transmitted by a computer
system executing a computer program in accordance with
claim 39.
Figure 1

Projecting a plurality of vertices of geometrical shapes that are defined in three-dimensions onto a plane.

Determining reference information that is usable to associate the projected vertices with each other, the reference information being determined by applying a reversible function to the projected vertices.
Figure 8

8/22

b1

b2

b3

mid
dne

Figure 8
Comparison of image compression algorithms for increasing bit plane resolution

Figure 11 (a)
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

G06T 11/00 (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI : G06T11/-, G06T15/-, G06T17/-, G06T19/- & Keywords (3D, 2D, depth, vertex, centroid, matrix) and like terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category*</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
</table>

Documents are listed in the continuation of Box C

[ ] Further documents are listed in the continuation of Box C. [ ] See patent family annex

- Special categories of cited documents:
  - "A" document defining the general state of the art which is not considered to be of particular relevance
  - "E" earlier application or patent but published on or after the international filing date
  - "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
  - "O" document referring to an oral disclosure, use, exhibition or other means
  - "P" document published prior to the international filing date but later than the priority date claimed

- Later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

- Document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

- Document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

- Document member of the same patent family

Date of the actual completion of the international search
18 September 2012

Name and mailing address of the ISA/AU
AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
Email address: pct@ipaustralia.gov.au
Facsimile No.: +61 2 6283 7999

Authorised officer
Ross Stopford
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. 0262832177

Date of mailing of the international search report
18 September 2012

Form PCT/ISA/210 (fifth sheet) (July 2009)
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
</table>
This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Publication Number</td>
<td>Publication Date</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

End of Annex