[Hsiu90] Hsiung P.-K., Thibadeau R. H., Wu M.: **T-Buffer: Fast Visualization of Relativistic Effects in Spacetime**, March 1992, *Computer Graphics* **24**(2), pp. 83-88.

[Max83] Max N. L.: **Computer Graphics Distortion for IMAX and OMNIMAX Projection**, December 1983, *Proceedings of Nicograph '83*, pp. 137-159.

[Paet90] Paeth A. W.: **Digital Cartography for Computer Graphics**, in Glassner A. (ed.), "Graphics Gems", 1990, Academic Press, pp. 307-320.

[Port84] Porter T., Duff T.: **Composing Digital Images**, July 1984, *Computer Graphics (Proceedings of SIGGRAPH '84)* **18**(3), pp. 253-259.

[Suff93] Suffern K. G., Hopwood S., Sinclair I.: **Reflections in a Spherical Cavity**, in Magnenat-Thalmann N., Thalmann D. (eds.), "Models & Techniques in Computer Animation", 1993, Springer, pp. 206-213.

[Watt90] Watt A. (ed.), **Fundamentals of Three-Dimensional Computer Graphics**, 1990, Addison Wesley.

[Wolb90] Wolberg G. (ed.), **Digital Image Warping**, 1990, IEEE Computer Society Press.

[Wyvi90] Wyvill G., McNaughton C.: **Optical Methods**, 1990, *Proceedings of Computer Graphics International (CGI) '90*.

Extended camera specification may be used to realize more elaborate viewing techniques (e.g., non-linear, locally varying zoom). Applications of such generalized cameras include virtual cameras for virtual reality, arts and generation of special effects.

Acknowledgements

The authors would like to thank the following persons for providing the ray tracing test environment: Martin Pottendorfer, Albrecht Kadlec, Peter Hackl.

References

[Acqu93] Acquisto P., Gröller E.: **A Distortion Camera for Ray Tracing**, in Connor J. J., Hernandez S., Murthy T. K. S., Power H. (eds.), "Visualization and Intelligent Design in Engineering and Architecture", April 1993, Computational Mechanics Publications, Elsevier Science Publishers, pp. 105-118.

[Acqu92] Acquisto P.: **Virtual Irreality: Eine Verzerrungskamera** (translation: **Virtual Irreality: A Distortion Camera**), 1992, Diploma thesis, Institute of Computer Graphics, Technical University of Vienna, Austria.

[Cook84] Cook R. L., Porter T., Carpenter L.: **Distributed Ray Tracing**, July 1984, *Computer Graphics (Proceedings of SIGGRAPH '84)* **18**(3), pp. 137-147.

[Cruz93] Cruz-Neira C., Sandin D. J., DeFanti T. A.: **Surround-screen projection-based virtual reality: The design and implementation of the CAVE**, August 1993, *Computer Graphics (Proceedings of SIGGRAPH '93)*, pp. 135-142.

[Dors91] Dorsey J. O'B., Sillion F. X., Greenberg D. P.: **Design and simulation of opera lighting and projection effects**,
July 1991, *Computer Graphics (Proceedings of SIGGRAPH '91)* **25**(4), pp. 41-50.

[Fole90] Foley J. D., van Dam A., Feiner S. K., Hughes J. F. (eds.), **Computer Graphics Principles and Practice**, 1990, Addison Wesley.

[Glas89] Glassner A. S. (ed.): **An Introduction to Ray Tracing**, 1989, Academic Press.

[Gree86] Greene N., Heckbert P. S.: **Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter**, June 1986, *IEEE Computer Graphics & Applications* **6**(6), pp. 21-27.

[Hain92] Haines K., Haines D.: **Computer Graphics for Holography**, January 1992, *IEEE Computer Graphics & Applications* **12**(1), pp. 37-46.

realized with **xyzT** transformers, the left topmost image is a top view, which is changed to a horizontal view (ordering of the images is from left to right, top to bottom). The centers of the images approximate a perspective projection. Close to the corners strong distortions are visible.

Image 7: Scene 3 is rendered. This is an example for functions **pa**, **gen** and **xyzT**. The Picture Arrangers are the same as in image 6. The starting points lie on a line through the center of a sphere which is used for the definition of the directions. The nine different views are realized by rotating the camera model from left images to right images and by increasing the radius of the sphere from top images to bottom images. This case shows that even a simple extended camera definition can produce already spectacular distortions. An object may not only be distorted but show up more than once in the same image.

Future Work

Future work will include the implementation and testing of further mappings which make up an extended camera specification. Special effort will be taken on exploring the Parameter Space Transformer **uvT** as well as the Object Space Transformer **xyzT**. More complex objects like height fields, bezier surfaces or other free form surfaces will be implemented. We want to concentrate on transformations, which produce only slight distortions, because within this area we expect most of the usability of our approach.

The incorporation of the extended camera definition system into a more powerful ray tracing system than the one used in the current implementation is planned as well.

Conclusion

A system of an extended camera specification for image synthesis has been presented. By generalizing the simple camera definitions that are typically used in ray tracing systems and introducing a sequence of mappings the user gains more flexibility to tailor the viewing parameters to his specific needs. Additionally the system allows to integrate several images within one compound image. For each of these integrated images an extended camera is defined. The concept of viewpoint and projection plane is extended to more general projection surfaces in combination with a flexible specification of the viewing direction.

An extended camera thereby consists of a sequence of mappings which assign to each pixel of the calculated image a 3D ray in object space, which is then taken as primary ray in the following ray tracing step.

Aliasing becomes quite a massiv problem using extended cameras. Applying a sequence of mappings may produce vastly differing rays for adjacent pixels. The flexible specification of the directions of the rays is thereby the biggest problem. This obviously requires the user to be cautious while specifying an extended camera. Anti aliasing is done through the corresponding features of the ray tracing system, e.g., by oversampling. Adaptive sampling seems to be a good choice to take care of locally varying aliasing problems.

test system. The implementation so far comprises the functionality as shown in figure 1. A camera description language has been defined and a parser implemented. Using this parser, extended cameras can be defined as scripts.

Sample images were generated for three simple test scenes:

Scene 1: Scene 1 contains three spheres on a plane which simulates a water surface. The spheres are mirror reflectors.

Scene 2: Scene 2 contains two tents, with one door each, on a green plane and a cloudy background. The tents have different color for the outside, the inside and the door. The scene is lighted by four light sources which are positioned inside the tents and one light source above the scene.

Scene 3: Scene 3 contains three pyramids, a sphere and a cube, all positioned on a plane simulating a water surface. Objects are arranged in three groups, a brown pyramid, a green pyramid together with a green sphere and a dark red pyramid on top of an orange cube. Two light sources are used.

Image 1: Scene 1 is rendered with perspective projection.

Image 2: Scene 1 is rendered. This is an example for Picture Arranger **pa**. Thirteen extended cameras are used, one for the background (top view) and twelve arranged in a circle (center view) and clipped to an ellipse each.

Image 3: Scene 2 is rendered with perspective projection.

Image 4: Scene 2 is rendered. This is an example for functions **uvT**, **gen**, **xyzT**. Function **getDirPnt** and **getEyePnt** are specified using two cylinders (coinciding axis, perpendicular to the green plane, positioned at the center of the plane). The **xyzT** transformer scales the cross section of the cylinder for the starting points towards the interior of the tents. The **uvT** transformer is used to generate a non-linear zoom effect along the horizontal direction of the image, which offers a close-up look of the scene at the left and right edge of the image (depicting the outside of the tent with greater detail). The image shows an outside and inside view of one tent simultaneously. At the left and right edge of the image one views the tent from the outside, closer to the center one looks around the inside of the tent, whereas in the center one looks from the inside towards the second tent. The door of the tent is visualized twice, once from the outside, once from the inside of the corresponding tent. Combinator **getDirVec** is "relative".

Image 5: Scene 3 is rendered with perspective projection.

Image 6: Scene 3 is rendered. This is an example for functions **pa**, **gen** and **xyzT**. The Picture Arrangers **pa** position nine different views of the same object scene in a regular fashion. The starting points are taken from a small portion of a large sphere. This large sphere intersects a smaller sphere, a large portion of thereof is used to define the directions. Combinator **getDirVec** is "relative". The different views are

Clipping of the image may be done on a simple shape, e.g., circle, ellipse, polygon, slabs, strips. As clipping is done in discrete space, i.e., for each point it is determined if it lies within the clipping area, much more elaborate and geometrically complex clipping objects are applicable. One may want, for example, to take only those points of an image which taken as parameters in the complex plane belong to a fractal set. In this case the clipped image is delimited by a geometrically intricate fractal object. The second step of a Picture Arranger **pa** is to apply an affine transformation, e.g., for image reduction. Finally the clipped and affinely transformed image of an extended camera is positioned onto the compound image.

As more than one image may be incorporated into a single compound image, one has to decide on how to handle overlapping portions. One approach assigns priorities to each image of an extended camera. These priorities then determine visibility in the compound image. Porter and Duff describe digital image compositioning (alpha channel method), that makes use of opacity coefficients for each pixel of a raster image [Port84]. These opacity coefficients greatly increase the possibilities to handle overlapping portions of images within the compound image.

Figure 11 shows how the different components of an extended camera definition are integrated. In the most general case one starts out with a pixel position in the compound image. Applying the inverse of the corresponding Picture Arranger **pa** produces a position in one of the images associated with an extended camera. Then functions **getEyePnt** and **getDirPnt** - each a composition of Picture Mapper **pm**, Parameter Space Transformer **uvT**, Generator **gen** and Object Space Transformer **xyzT -** generate 3D points. The points returned by function **getEyePnt** define the starting points, the combinator **getDirVec** and 3D points returned by function **getDirPnt** define the directions of the rays that are then used in the following ray tracing step.

Implementation, Results

Mapping	*Functionality*
Picture Arranger **pa**	Ellipse, strips as clipping objects, affine transformation for placement within compound image, visibility due to priority.
Picture Mapper **pm**	Linear scaling and translation.
Parameter Space Transformer **uvT**	Affine transformations, non-linear parameterizations.
Generator **gen**	Point, line, circle, plane, sphere, cylinder, cone and torus as 3D objects.
Object Space Transformer **xyzT**	Affine transformations.
Combinator **getDirVec**	"relative", "absolute".

Table 1: Functionality of the implemented Test System

An extended camera definition system was implemented in C++ and integrated into a simple ray tracing system. The object oriented paradigm is well suited for incremental extension of the

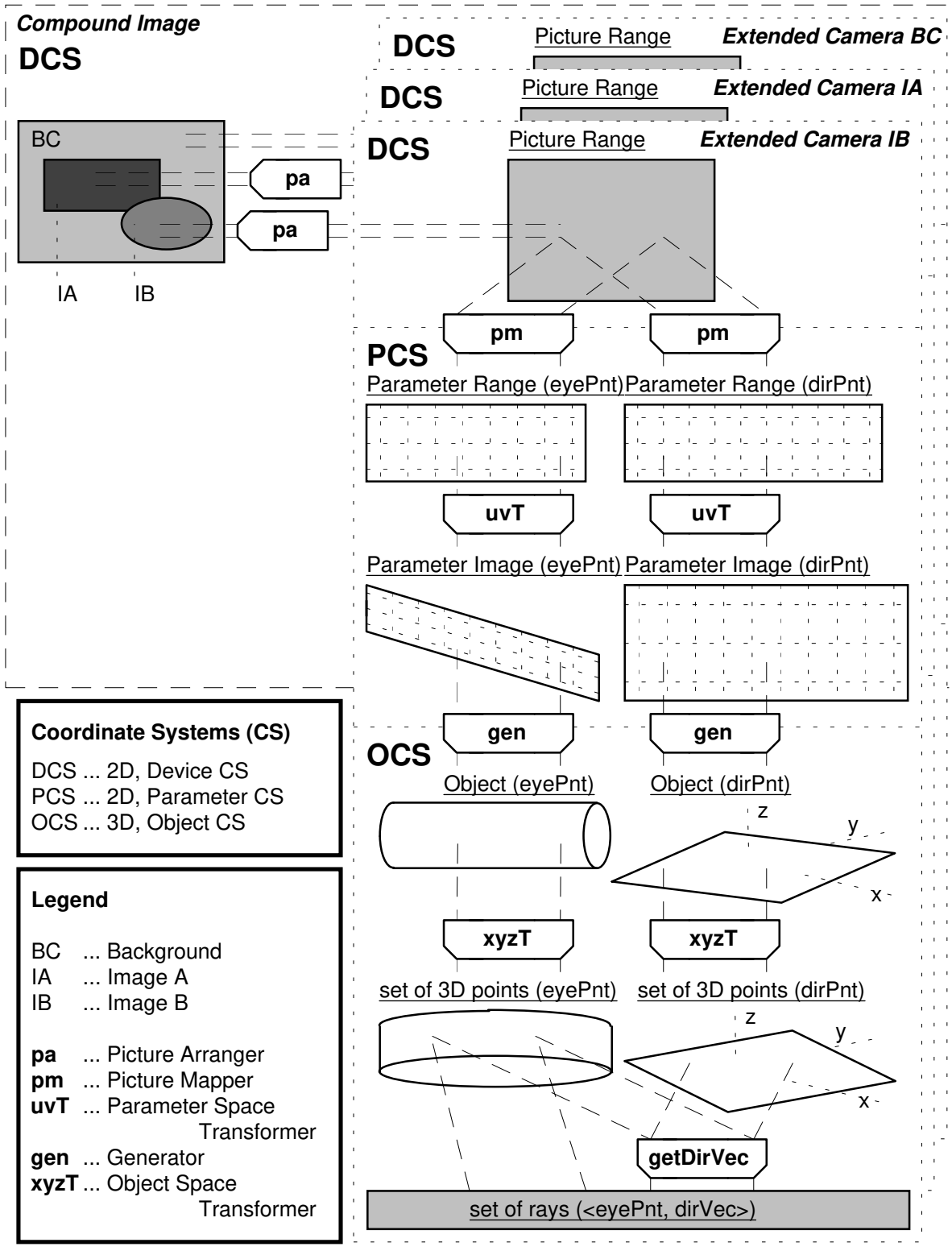

Figure 11: Extended Camera Specification including Picture Arranger **pa**

Thus for every image of an extended camera which is to be placed within the compound image a Picture Arranger is specified. Figure 11 illustrates the whole concept of an extended camera including the Picture Arranger **pa**. First a Picture Arranger **pa** performs a clipping operation.

camera at some specific location and do the exact positioning of the whole camera later on. For example a perspective projection is easier to define by assuming that the projection plane coincides with the xy plane and taking the viewpoint to lie on the -z axis. The exact positioning in object space is done only afterwards by specifying an affine transformation. An affine transformation as a **xyzT** transformer allows scaling and positioning of the camera model by changing the resulting 3D points of function **gen** globally in a uniform way. The second type of application of **xyzT** transformers again increases the flexibility of the extended camera model. In this case functions are used as **xyzT** transformers which change 3D points of function **gen** in a locally varying way. This realizes the concept of COINs (centers of interest) introduced by Acquisto and Gröller [Acqu93]. A COIN is a 3D position which marks a location of object space that is of special interest to the user. Rays are then, depending on their distance to the COIN, biased towards this center of interest. COINs produce non-linear zoom effects and are easily realisable as **xyzT** transformers. As another example a **xyzT** transformer may add noise to the 3D points of function **gen** in a stochastic fashion. Like the **uvT** transformer the **xyzT** transformer is again optional in the sequence of mappings.

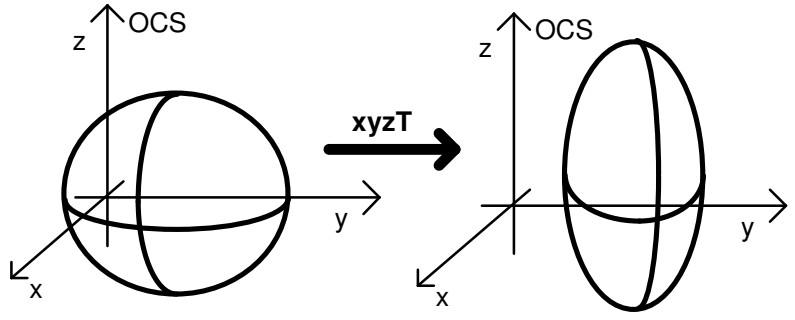

Figure 10: Object Space Transformer **xyzT**

The Picture Arranger pa

An extended camera defined so far enables the integration of different views within a single image. We finally introduce one more transformation, the so called Picture Arranger **pa** (DCS→ DCS). The Picture Arranger **pa** allows to integrate several images each due to a different extended camera within a single image (so called compound image). This makes it possible to compose several views of a scene into a collage. Although there exist already quite a number of stand alone image composition techniques like the so called alpha channel method introduced by Porter and Duff [Port84], we added this possibility directly to our approach. The advantages of this decision are that we don't have to calculate entirely each image we later want to combine. Only those portions of an image are calculated that are really needed in the compound image. On the other hand incorporating the **pa** into the camera model makes it possible to make the camera specification for the related picture depending on the **pa** itself.

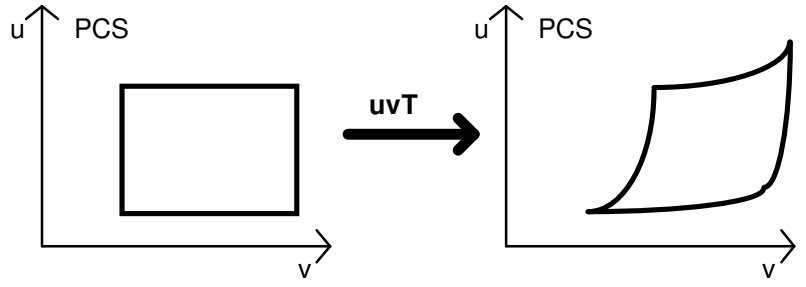

Figure 8: Parameter Space Transformer **uvT**

Generator Function gen

Generator function **gen** (PCS→OCS) defines a mapping from parameter space into object space. (see figure 9). **gen** may specify a three dimensional object which is then used to define the starting point or the viewing direction of the ray. Such an object could be a point, a line, a parametric curve, a plane, a quadric surface (sphere, cylinder, cone), a torus, a soft object, a free form surface (e.g., NURBS surface). Taking, for example, a free form surface as function **gen** to model the projection surface gives the user ample possibilities to control the object shape. One can think of even a larger class of functions to be taken as **gen** which may not even be continuos as is the case with the examples above. One, may be a little bit strange, choice, for example, would be to take height fields of fractal sets as generator function **gen**. Such choices enable the generation of a wide variety of special effects. Another aspect of the generator function **gen** is how to define the parameterization of the previous specified object. Different parameterizations heavily influence the resulting image. For example a sphere may be parameterized by longitude and latitude or by projecting a rectangular subset of parameter space onto the sphere. So again a wide variety of possibilities are user definable.

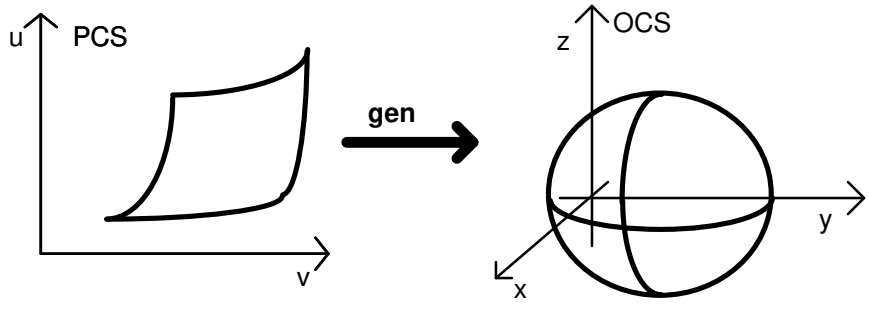

Figure 9: Generator Function **gen**

Object Space Transformer xyzT

The **xyzT** transformer (OCS→OCS) maps 3D points produced by the previously applied function **gen** into other 3D points within object space (see figure 10). There are two reasons to introduce such a transformation of the object space. Sometimes the user may want to specify the

can specify the parameter space according to his requirements without being restricted to have to use device coordinates.

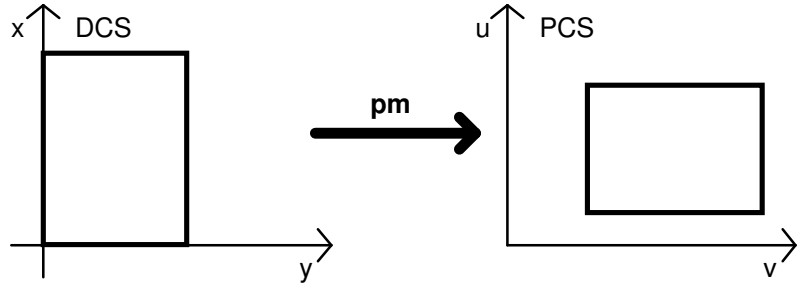

Figure 7: Picture Mapper **pm**

This function is inserted into the sequence of mappings, because often DCS and PCS are intuitively different. For example, on many screens the origin of the DCS lies in the top left corner with the x axis running to the right and the y axis to the bottom defining the Picture Range as rectangular area [0, picWidth]×[0, picHeight]. The default parameterization of the surface of the whole sphere might be seen as $[-\pi, \pi]\times[-\pi/2, \pi/2]$ using longitude and latitude as parameterization, whereby the origin of the PCS lies on the equator and the 0°-meridian with the u axis running to the east and the v axis running to the north. As neither the width/height ratio nor the orientation of the axises of these two rectangular areas are the same, it is easier for the user to specify the mapping between these coordinate systems first. In our example the affine transformation **pm** might be defined by the pictures of three points from the DCS: $\langle 0, 0\rangle \rightarrow \langle -\pi, \pi/2\rangle$, <picWidth, 0> → $\langle \pi, \pi/2\rangle$ and <picWidth, picHeight> → $\langle \pi, -\pi/2\rangle$.

Parameter Space Transformer uvT

The **uvT** transformer (PCS→PCS) maps the parameter space into itself (see figure 8). One can use the **uvT** transformer to specify more general parameterizations of those objects that are produced by the following function **gen**. Examples of functions that can be taken as **uvT** transformer are affine transformations (scaling, rotation, translation, shearing) and non-linear functions defined by a regular grid of control points (grid warping) [Wolb90].

The following two examples of extended cameras illustrate the usefulness and power of **uvT** transformers. In the first example the **uvT** transformer is used to produce non-linear zoom effects with continuously varying zoom factor. One may especially be interested in a certain portion of an image but still would like to retain an overview of the whole scene. By specifying a **uvT** transformer that enlarges the interesting portions in parameter space such an effect is easily realizable. In the second example the **uvT** transformer is taken to be a simple shear in v direction, and the objects of the scene are to be projected onto a cylinder (coordinate v runs around the cylinder, coordinate u is parallel to the cylinder axis). With this arrangement the viewpoint changes continuously from the top of the image to the bottom. Viewing a stick (which is parallel to the cylinder axis) with such an extended camera produces an image where the stick appears to be twisted. The Parameter Space Transformer **uvT** is optional in the sequence of mappings used for **getEyePnt** and **getDirPnt**.

is defined. Using a screen as output device, the values represented in the DCS are strongly related to pixels on the screen. The Object Coordinate System (OCS) is a three dimensional coordinate system which is used to represent the scene and objects of the extended camera. Such an object of an extended camera could be, for example, a sphere if the 3D scene is to be projected on this sphere or a part of it. The set of 3D points which result from applying functions **getEyePnt** and **getDirPnt** are represented in this coordinate system. Between the Device Coordinate System (DCS) and the Object Coordinate System (OCS) a two dimensional Parameter Coordinate System (PCS) is introduced. The Parameter Coordinate System (PCS) allows a device independent parameterization of the objects used for the extended camera. Furthermore transformations within the PCS increase the number of definable extended cameras.

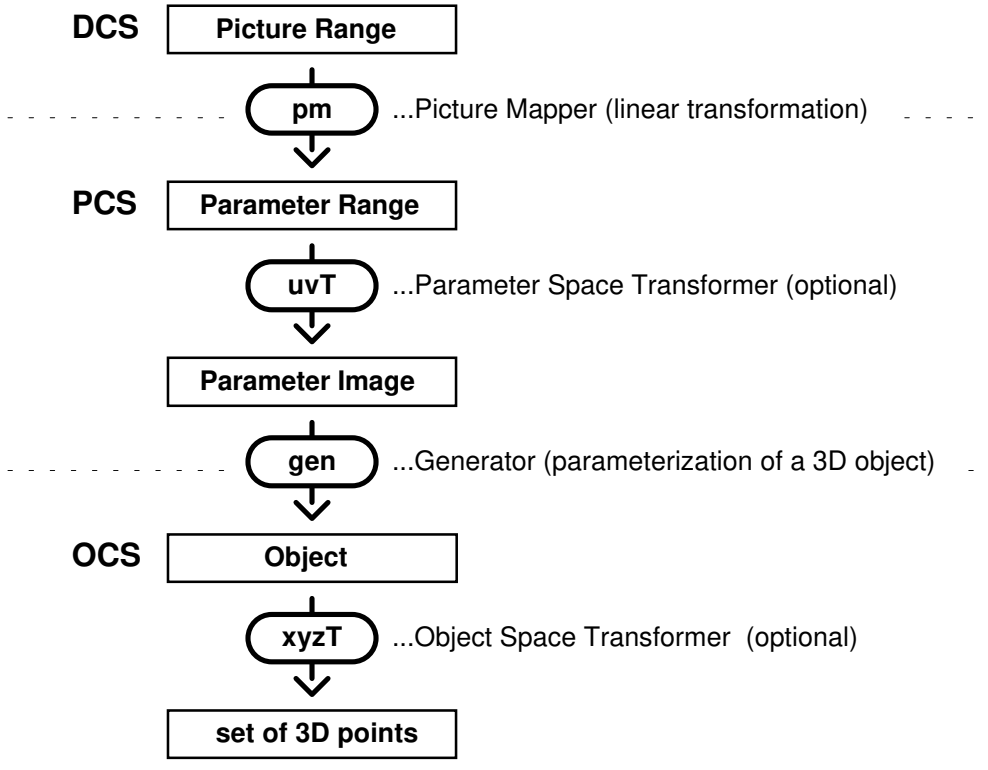

Figure 6: The Sequence of Mappings: **pm**, **uvT**, **gen** and **xyzT**

Four consecutive mappings make up the functions **getEyePnt** and **getDirPnt** (see figure 6). The Picture Mapper **pm** is a transformation from the DCS into the PCS to establish device independence. The **uvT** transformer performs a transformation of the parameter space. Function **gen** maps the parameter space into object space. This function specifies the parameterization of objects (e.g., sphere, cylinder) which define the projection surface and projection directions in the object space. Finally the **xyzT** transformer modifies the objects generated by the previous applied function **gen**.

Picture Mapper pm

Picture Mapper **pm** (DCS→PCS) is a simple affine transformation (scaling, mirroring and/or translation) to separate device coordinates from parameter coordinates (see figure 7). The user

defined through an extended camera specification. As another example, cartographic projections are included into the above scheme as well.

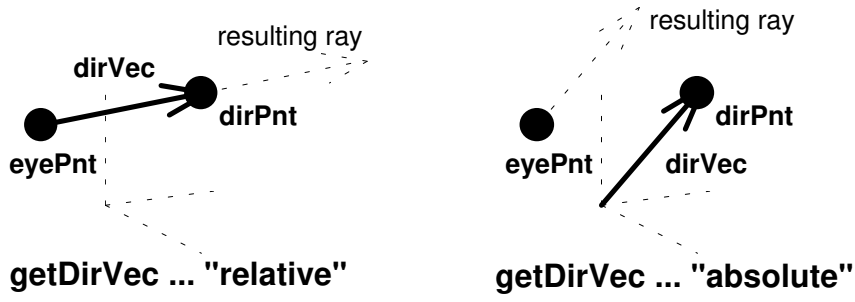

Figure 4: Two possible Combinators **getDirVec**

Figure 5 gives an example of an extended camera with the following characteristics: **getEyePnt** parameterizes a sphere, **getDirPnt** always returns the center of the sphere and the combinator **getDirVec** is "relative". As can be seen easily, this defines a camera, where one looks from the surface of a sphere right into the center of the sphere. According to the parameterization parts or the whole surface is used as projection surface. Such a type of camera allows to view the scene from different sides within one single image. The increased viewing range is paid for with distortion effects that show up in the final image.

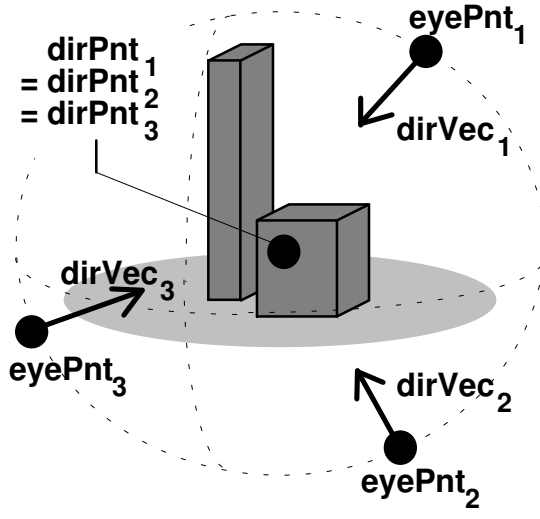

Figure 5: Extended Camera, Example: Looking from a Sphere towards the Center

Mapping the Picture Range to a Set of Rays

In the following sections we will discuss the functions **getEyePnt** and **getDirPnt**, which map the Picture Range into a set of 3D points each. Such a mapping is a combination of four consecutive mappings involving three different coordinate systems (see figure 6). The Device Coordinate System (DCS) is a two dimensional coordinate system (CS) where the Picture Range

ray, whose starting point is given as the center of the pixel on the projection plane and whose direction is taken to be constant (parallel projection lines). With perspective projection a pixel of the Picture Range is mapped to a ray whose starting point (viewpoint) is constant for all pixels and whose direction runs from this constant starting point through the center of the pixel on the projection plane.

The specification of an extended camera proceeds as shown in figure 3. As we use a slightly different definition of a ray as mentioned above, the definition of the Camera Function **getRay** is now given by three components: **getEyePnt**, which calculates the starting point **eyePnt** of the ray, **getDirPnt** and **getDirVec**, which together determine the resulting ray direction **dirVec**. The functions **getEyePnt** and **getDirPnt** are both of the same type, which has two advantages: It is easier for the user to specify an extended camera and provides more reusability for implemented mappings (see figure 3).

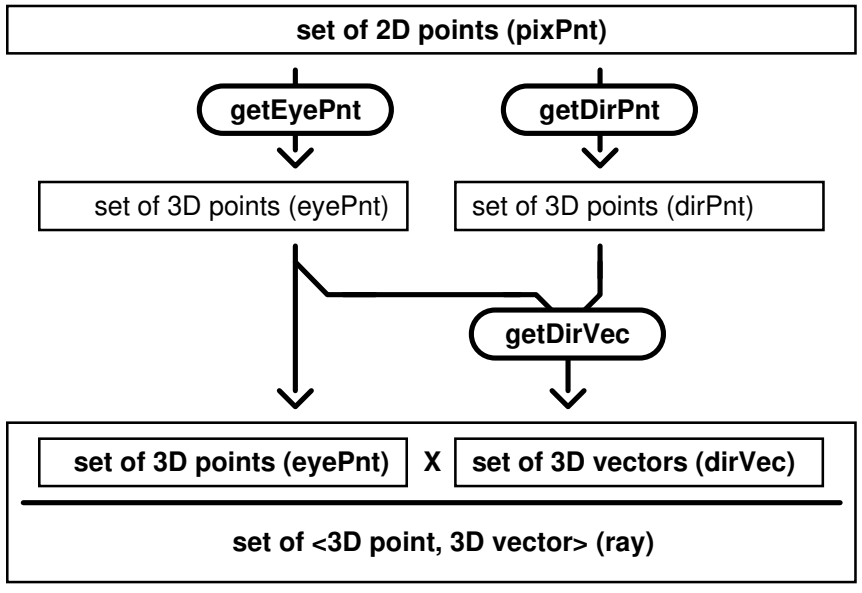

Figure 3: Extended Camera Model

Figure 4 shows how different sets of a function **getDirPnt** and a combinator **getDirVec** may specify the direction **dirVec** of a ray. In the first example **getDirVec** is "relative", i.e., the direction is given as **dirVec** ← **dirPnt - eyePnt**, and in the second example **getDirVec** is "absolute", which means that **dirPnt** itself is interpreted as direction vector (**dirVec** ← **dirPnt**).

The Extended Camera Model of figure 3 does not put any restrictions on the functions **getEyePnt**, **getDirPnt** and the combinator **getDirVec**. All the usual camera projections can be easily specified within this model. Thus a perspective projection is defined as follows: **getEyePnt** results in a constant **eyePnt** - the viewpoint, **getDirPnt** parameterizes the projection plane and **getDirVec** is so called "relative" (see above). A parallel projection is given as: **getEyePnt** parameterizes the projection plane, **getDirPnt** is taken to be a constant **dirPnt** - the projection direction. Therefore **getDirVec** has to be "absolute" (see above).

The functions **getEyePnt** and **getDirPnt** may specify parameterizations of more general objects than planes. Thus projections of the scene onto, e.g., spheres, cylinders, etc., are easily

Because the length of **dirVec** is finally normalized the direction vector **dirVec** can be specified by two values and thus this definition is equivalent to the one of Wyvill and McNaughton.

A sequence of mappings (**pm**, **uvT**, **gen** and **xyzT**), is introduced to map 2D points from the image to 3D points in object space. For each pixel **pixPnt**, we want to get a ray for, we now use two instances of this sequence, **getEyePnt** and **getDirPnt**, to calculate the 3D points **eyePnt** and **dirPnt**. Since there are several possibilities of constructing the **dirVec** from these two points, the type of combination is specified by **getDirVec**. Additionally an image composition, **pa**, is incorporated within this model to enable the combination of several different views within one picture without bloating the calculation effort.

Such an extended camera specification could be used, for example, for implementing non-linear zoom effects. This might be done, when some portions of the 3D environment are more interesting to the user than other parts and he would like to get a more detailed view of that portion with the remaining 3D environment still present in the final image. Leaving the restrictions of a perspective projection behind, with an extended camera specification viewing all six faces of a cube is not a problem. Thus within a single image one can get the whole topological information of a 3D object without the need to change the viewpoint. In just the same way it is possible to view the interior as well as exterior of a building within the same image.

Application areas of extended camera specifications are: creating elemental holographic images and raster OMNIMAX or IMAX images, more elaborate viewing techniques, visualization of scientific and mathematical data, background projections for opera stage settings [Dors91], projections in virtual reality [Cruz93], arts, special effects.

Extension of the Camera Specification for Ray Tracing

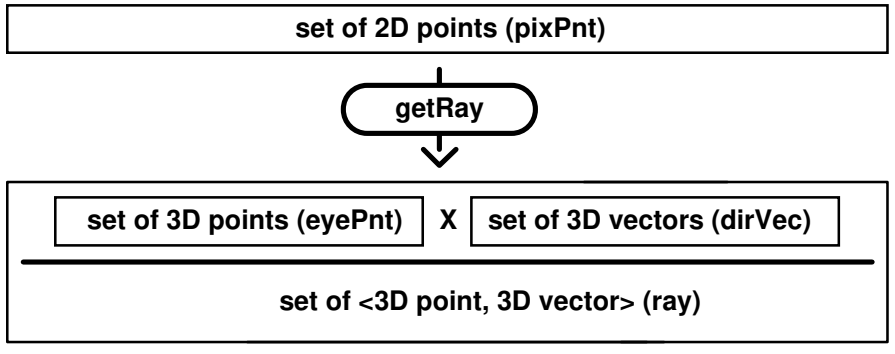

Figure 2: Standard Camera Model: Mapping the Picture Range into a Set of Rays

A standard camera model is integrated into a ray tracing system as follows: for each pixel of the resulting raster image at least one primary ray is traced through object space. A ray is thereby given by two components, i.e., a starting point **eyePnt** and a direction vector **dirVec** (see figure 2). The camera model thereby defines a mapping from the Picture Range, which is usually a rectangular grid of pixels and thus a finite subset of the 2D image space, to a Set of Rays, one for each point of the Picture Range. With this model perspective and parallel projections are easily specified. With parallel projection a pixel of the Picture Range is mapped to a

enhancements, the depth of field aspect to ray tracing [Cook84]. They use a sampling method with a stochastic distribution of the rays to simulate lens effects.

Different types of projections with widely varying characteristics have been already used for various purposes. Cartographic projections [Paet90], for example, map the surface of the earth onto 2D maps (e.g., stereoscopic projection, cylinder projection, Mercator projection). Artists sometimes make use of non-standard projections as well. In some of the paintings of Pablo Picasso more than one point of view is incorporated within a single image, which produces quite strange effects. M. C. Escher is one other prominent example of an artist that worked with unusual projections, distorted views and distorted images. Although we know, that we cannot be seen as artists, our work has aesthetic aspects comparable to art. Suffern, Hopwood and Sinclair point out that aspect in their work on reflections inside a sphere as well [Suff93].

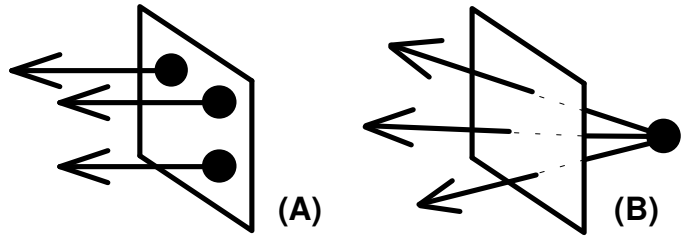

Figure 1: Primary Rays using Parallel (A) or Perspective Projection (B)

Multiple views within a single image are also applied for generating OMNIMAX or IMAX images [Gree86] [Max83] and holographic images [Hain92]. Hsiung, Thibadeau and Wu produced distorted images by visualizing relativistic effects [Hsiu90].

Important work on the area of extended cameras was done by Wyvill and McNaughton [Wyvi90]. They first generalize the mapping, which is performed by a camera function for usage with the ray tracing technique. Thus the camera function can be seen as mapping each point of the image to a primary ray (starting point and direction) for the following ray tracing step:

$$\text{Camera Function } \textbf{getRay}: \Re^2 \rightarrow \Re^5 \ (\Re^3 \times \Re^2).$$

They continue with the presentation of several possible functions **getRay** (e.g., super fish eye projections, all-around views) and present the idea to produce an animation, where the camera is transformed from a standard one into a non-standard one. Acquisto and Gröller worked out several extended camera functions and introduced a method with so called COINs, which stands for centers of interest [Acqu92] [Acqu93].

Due to the point sampling nature and pixel by pixel approach of ray tracing, i.e., the calculation for one pixel of an image can be done independently from other pixels, this image synthesis technique is well suited for investigating different projection methods. This paper presents a more detailed specification of the extended camera model based on the general ideas of Wyvill, McNaughton [Wyvi90] and Acquisto, Gröller [Acqu92] [Acqu93]. Slightly different to the definition of Wyvill and McNaughton we assume a ray to be represented by an eye point **eyePnt** and a direction vector **dirVec**:

$$\text{Camera Function } \textbf{getRay}: \Re^2 \rightarrow \Re^3 \times \Re^3.$$

Extended Camera Specification for Image Synthesis

Eduard Gröller, Helwig Löffelmann[1]

Institute of Computer Graphics, Technical University of Vienna
A-1040 Karlsplatz 13/186/2

Abstract

Typically a rather simple definition of the viewing parameters (e.g., perspective projection) is used for ray tracing, which is one of the most powerful rendering techniques for generating realistic images. In this paper this simple camera model is extended in several ways to increase user flexibility. A sequence of mappings defines a more general way to assign 3D rays for each pixel of the final raster image. These rays are then taken as primary rays in the following ray tracing step. An extended camera specification, for example, allows the projection of the object scene onto more general projection surfaces (quadrics, free form surfaces). Different views of the same object scene may be incorporated within one single image. A test implementation and sample images show the feasibility of the approach. The extended camera specification enables more elaborate viewing techniques which may be used, e.g., for virtual reality, arts and special effects.

Introduction

Ray tracing is a powerful and versatile technique for realistic image generation [Glas89]. Light propagation within a 3D environment is simulated by tracing rays to determine the amount and distribution of light perceived by a viewer. A raster image is calculated pixel by pixel, i.e., for each pixel rays are cast into the 3D object scene. Depending on the surface characteristics of the object hit by a ray and depending on the shading model used, additional rays are traced, for example, in the direction of reflection and refraction and towards the light sources. A ray tracing system thus consists of several distinct components. Some of these components are: the object model, which describes the 3D scene to be rendered, the lighting model, which describes light sources (point light sources, area light sources), the shading model, which describes the light-surface interaction, and the camera model, which defines how the 3D environment is projected onto a 2D image space [Watt90].

Typically rather simple camera models are used: parallel projection maps the 3D object scene onto a 2D image plane along parallel rays, whereas with perspective projection (see figure 1) these rays intersect in a single point, the viewpoint [Fole90]. There have been some approaches in computer graphics where the human visual perception is modelled more accurately than with perspective projection. Cook, Porter and Carpenter added among other

[1] mailto:groeller@cg.tuwien.ac.at
mailto:helwig@cg.tuwien.ac.at, http://www.cg.tuwien.ac.at/~helwig/