

CAMPLAN: A Camera Planning Agent

Nicolas Halper^{1*} and Patrick Olivier^{**}

*Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg,
Universitätsplatz 2, Magdeburg D-39106, Germany
nick@isgnw.cs.uni-magdeburg.de

**Department of Computer Science, University of York, Heslington, York, YO10 5DD, UK
patrick.olivier@cs.york.ac.uk

From: AAAI Technical Report SS-00-04. Compilation copyright © 2000, AAAI (www.aaai.org). All rights reserved.

Abstract

Camera planning agents address the problem of positioning a camera within a world, such that the resulting image has some predefined set of visual properties. We are developing a graphical presentation planning system which incorporates CAMPLAN, a camera planning subsystem for polygonal graphics. The motivations for CAMPLAN are outlined and a set of shot properties has been established and implemented from which the user can select communicative goals. CAMPLAN uses a genetic algorithm to optimise the camera with respect to the specified set of image objectives. An informal evaluation of the system is presented in which we show how successfully more restrictive objectives can impose stylistic regularity in a graphical scene and demonstrate other features of declarative camera planning using CAMPLAN. We conclude with a discussion of our current work and outline future directions for graphical presentation planning.

Introduction

There is an increasing need for algorithms that compute an appropriate shot of a three-dimensional graphical model. For example, a number of multimedia systems which aim to combine text and graphics have been proposed (Seligmann, 1993; Wahlster et al, 1993; Zhou & Feiner, 1998), yet all have been limited by their ability to relate linguistic references to the graphical content of a shot. Complex simulation environments, such as production simulation, have many events happening concurrently, and optimally positioned camera views maintaining visualization of the important aspects in the simulation (for example, where a particular component is located, and where it is heading) would increase awareness of the situation.

In an attempt to provide a generic camera agent suitable for such a variety of applications we present CAMPLAN, the camera planning component of a graphical presentation planning system currently in development. The first part of the paper will describe a

brief overview of existing work on graphical presentation planning and current limitations of such systems. The CAMPLAN system will then be introduced and evaluated (sections 3 and 4 for a summary). To conclude the presentation, in section 5 we will point to current and future directions for this research.

Motivation

Existing techniques for camera planning tend to be limited to specific application contexts (Seligmann, 1993; Blinn, 1993) or largely based on point abstractions (Christianson et al, 1996) and cannot therefore account for the range of visual effects that arise from the fact that real scene elements have finite extents (e.g., occlusion between scene elements). Whereas a recent attempt by Bares and Lester was successful in constructing a real-time camera visualization interface for dynamic 3D worlds (Bares et al, 1998; Bares and Lester, 1999) it consequently suffered in the expressiveness of its viewing constraints (which were limited to viewing angle, distance, and occlusion avoidance).

Drucker (Drucker, 1994) formulated the task of finding an image with a particular set of scene element position properties. This was treated as a constrained optimization problem whose solution was sought numerically using Newton's method. The range of properties that Drucker allowed was limited to the point idealizations of scene elements in the image, and relative orientations between scene elements and the camera. Due to the local nature of such numerical methods we have empirically demonstrated that this approach is less effective as a solution method for more complex scenes and large sets of properties.

CAMPLAN is an attempt to address the principal shortcomings of these approaches: the restriction placed on the range of image properties that may be specified and the unrealistic point-based characterizations of scene elements. To provide for a wide range of applications CAMPLAN is to solve problems for camera planning residing in the latter two

¹ Original work produced on CAMPLAN was completed while the first author was at the University of York.

stages of the following graphical presentation pipeline characterized in (Seligmann, 1993):

- (a) *Generation of communicative goal*: decide what it is that the image should accomplish.
- (b) *Selection of presentation strategy*: determine the visual effect that will be used to satisfy the communicative goal.
- (c) *Selection of presentation method*: given the presentation strategy specify the different ways by which it may be realized.
- (d) *Image generation*: based upon the selection of the presentation methods, the graphical model of the artifact must be modified to achieve the specified visual properties.

The CAMPLAN System

Our aim is to incorporate CAMPLAN as the camera planning subsystem of Seligmann's graphical presentation system. Following Seligman's proposal we have thereby arrived at a number of requirements for a camera planning module which can be stated as follows:

- *selection of shot objectives*: CAMPLAN must establish a set of shot properties that a user may select in order to achieve desired visual goals.
- *evaluation of objectives*: each shot objective must be well defined and efficient to evaluate with respect to the underlying graphical modeling paradigm.
- *acquisition of a camera state*: an image must be generated by modifying the camera state (location, orientation and field of view) such that the fulfillment of the specified objectives is maximized.

The strength of the system will be in planning expressive static visual shots of scenes intended to achieve a specific visual communicative goal. In the following subsections we describe the requirements and operation of CAMPLAN through a brief account of the resolution of each of these problems.

Selection of Shot Properties

In this section we establish a set of shot properties which the user can select for a presentation. The elements of the set of shot properties can be classified according to their type, quantitative character (absolute or relative) and the reference frame with respect to which they are characterized.

A classification of shots

Studies of human cognition yield several possible reference frames, for example, relative to which spatial

references (e.g. the use of spatial prepositions) may be made. In general these are the intrinsic (object centered), deictic (viewer centered), and allocentric (world centered) reference frames. Whilst a camera is essentially a deictic point of view, specified by its orientation, position in space, and field of view, the properties of a shot (for example, the position of an object in a shot) for a camera state can be further characterized with respect to the viewplane, viewport and viewpoint of the camera:

- *Viewplane*: the surface which extends infinitely in all directions perpendicular to the view direction of the camera. Often it is useful to specify objects in off-screen space (e.g. relate part of an object outside the screen with those in the screen).
- *Viewport*: subset of the viewplane with a defined bounded region for the view with centre-point of projection (0.0, 0.0) and upper-left corner (-1.0, 1.0). This specifies the view and comprises the image output.
- *Viewpoint*: the location of the camera. Camera placement can be controlled in an environment by using properties defined over elements surrounding the viewpoint by specifying positional measures such as angle or distance.

Specification of shot property values and tolerance settings can be made either explicitly or implicitly to these reference frames. Hence, two additional characteristics of shot properties:

- *Absolute*: The shot property types are defined explicitly, as an absolute relationship (e.g. object A is to be projected 20% the size of the viewport).
- *Relative*: The shot properties for objects are defined by reference to the properties of other objects (e.g. object A is to be projected 20% bigger than object B in the viewport).

Shot types

A type of image property is that which is most characteristic of the constraint imposed on a scene element. For an object in a scene this includes its position, size, visibility and orientation. Ideally, these types should be independent of knowledge about the topology of objects and the spatial arrangement of the scene so that they can be applied generically.

- *Position*: the location an element of the scene, with respect to the reference frame (viewpoint, viewport or viewplane). Thus, a user may characterize both the physical location of an object (position with respect to the viewpoint), visual location of an object (position with respect to the viewport) and the off-screen visual location of an object (position with respect to the viewplane)

Position	Viewplane	Viewport	Viewpoint
Absolute	CenterX/Y/XY BetweenX/Y InFrontOf Behind	CenterX/Y/XY BetweenX/Y InFrontOf Behind	AngleX/Y BetweenAngleX/Y
Relative	CenterX/Y/Z BetweenObjectsX/Y CloserThan ExtentsX/Y/Z	CenterX/Y/Z BetweenObjectsX/Y CloserThan ExtentsX/Y/Z	AngleExtentsX/Y/Z AngleX/Y Distance

Table 1. Position properties, text in light grey for the viewport image functions are identical in function to the viewplane image functions. In the case of BetweenX/Y, the extent values are clipped to the viewport.

Size	Viewplane	Viewport	Viewpoint
Absolute	ProjectedArea ProjectedLengthX/Y SpanXorY	ProjectedArea ProjectedLengthX/Y SpanXorY	ProjectedArea ProjectedLengthX/Y AngleWidth AngleHeight
Relative	ProjectedArea ProjectedLengthX/Y SpanXorY	ProjectedArea ProjectedLengthX/Y SpanXorY	ProjectedArea ProjectedLengthX/Y AngleWidth AngleHeight

Table 2. Size properties, viewplane and viewport will give different evaluations on size. The viewport is a subset of the viewplane in that it gives a clipped portion of the objects projected area in the viewing window. The viewpoint calculates projected areas onto a sphere surrounding the camera.

Orientation	Viewplane	Viewport*	Viewpoint
Absolute	Angle	Angle	Angle
Relative	N/A	N/A	N/A

Table 3. Orientation properties, viewplane and viewport describe identical image properties. No relative relationships exist since orientations of scene elements to other scene elements are independent of camera position.

Visibility	Viewplane	Viewport	Viewpoint
Absolute	Occluded	Occluded InViewport	Occluded
Relative	Occluded OccludedBy	Occluded OccludedBy	Occluded OccludedBy

Table 4. Visibility properties, viewplane and viewport will give different evaluations on size. The viewport is a subset of the viewplane in that it gives a clipped portion of the objects projected area in the viewing window. The viewpoint also has to compute projected areas, although in this case they are projected onto a sphere surrounding the camera.

- *Size*: how large an object is to appear. The projected size of an object is dependent on the actual physical size of the object, the field of view of the camera, and the distance to the point of projection. This also provides an indirect, but often more natural, means for the user to specify distance to an object and its visual impact.
- *Orientation*: each scene element can have a specified orientation in the final image. Orientation shot properties enable the user to view certain sides of an object that might be difficult to specify otherwise.
- *Visibility*: this is usually defined as the requirement that some portion of an object must be viewable in the image. Visibility properties may be used to specify the general visibility of an object, or that two or more objects are in a specific occlusion relationship (thereby indirectly characterizing the orientation of a scene to a camera).

Generation of shot properties

A taxonomy of image properties was created by considering permutations of the image property types

with relationships in the various dimensions (viewpoint, viewpoint, viewplane, absolute and relative). Tables 1-4 show the current set of image properties implemented. In addition, the graphical models allow the specification of the part-whole structure of each scene element, thereby allowing the specification of scene properties over named parts of scene elements. The user may also apply properties across a group of specified named parts. Furthermore, the user must be able to specify tolerance on these properties.

Evaluation of Shot Properties

CAMPLAN uses a polygonal representation of the scene elements. Some image properties are very inexpensive to evaluate, such as determining the position of an object's center. In this case, a representative bounding sphere for the object is pre-computed for the object, and its position is evaluated as the location (or projected location) of the center of the bounding sphere. Most image properties are more complex to evaluate, though in turn they may be simplified and optimized with respect to user's demands on the output. The algorithms presented in this section aim to exploit the user-sensitivity of shot property requests based on given user tolerance settings, and also cull redundant information for particular shot properties.

Efficiency of the evaluation algorithms can be optimized with respect to the current user preferences. For instance, if the user is known to be viewing the scene in a lower resolution, then an evaluation of a number of the properties may themselves be performed at a lower (and therefore less expensive) resolution. Similarly, cruder evaluations can be chosen based upon user tolerance settings, by considering higher tolerances as an implicit statement to the importance of a property (the greater the tolerance the less important the property). The following subsections briefly outline the shot property evaluation algorithms.

Projected Extents

Often the projected extents of an object need be determined in order to evaluate whether an object lies in some specific limits (e.g. `PositionBetweenX`). Complex quantitative and qualitative relationships between the projected extents of scene elements may also be specified using the `RelObjectsPosition` property, including all of the thirteen qualitative relations that can hold between two one dimensional intervals (Allen, 1983), and quantitative variations on these which we parameterize by the magnitudes of the extents of the objects. It is our belief that qualitative relations between the visual properties of scene elements are a more natural way to specify shots.

The projected extents of elements in a scene are determined by initially computing the convex hull (as

a pre-processing step) using a quick-hull algorithm. We then pick a random point on the hull and run along the edges to the hull extents by choosing the vertex neighbors which maximize the local progression. The projected extents for an object may be requested several times over the set of specified shot properties, and therefore results are tagged and stored to eliminate re-computation.

In the case that part of an object cuts the viewplane, the object must first be clipped according to the original object (the convex hull may give erroneous results), and we may need to revert to a brute-force algorithm finding the extents over all the vertices comprising the clipped object. However, in some cases calculations based upon the projected extents of the clipped convex hull of elements will be sufficient for solution convergence since approximate values of tolerable error are returned (often clipped projected extents comprise large values due to perspective distortion).

The results returned are at sub-pixel accuracy. However, often sub-pixel accuracy is not necessary for properties with imprecise specifications and it would be just as good to use coarser evaluations. For example, specifying an object to be positioned somewhere to the left of another object, without specifying the magnitude of the relationship, enables the use of simplified bounding sphere calculations rather than more computationally expensive projected vertex extent calculations to return the desired evaluation.

Evaluation Window

In order to maximise the accuracy of a discretised evaluation (such as visibility calculations performed at pixel accuracy) an evaluation window may be zoomed in over the elements comprising the shot property. This enables us to maximise the precision of a shot property evaluation and minimise the cost of the evaluation by increasing the resolution of the window over the desired areas of interest.

The evaluation window is set to enclose only the combined projected extents of the objects under evaluation. The resolution of the window is dynamically modified based upon the number of pixels comprising the projected region as displayed in the user window. Given the dimensions of the user's viewing window, we may estimate the required resolution to give an appropriate evaluation residing within the user tolerance values for the shot property. If the objects in the shot property are only a few pixels large, we lower the resolution of the evaluation window to the visible size in the users view. If the objects for evaluation are projected larger than a certain threshold, we increase the resolution of the evaluation window. We have found by basic experimentation that an evaluation window of 32x32 pixels for full views of between 400x300 to 800x600

resolution results in acceptable accuracy for tolerance settings of $\pm 10\%$. Minimising the resolution of the evaluation window gives significant performance improvements since much of the time taken for computation is the rendering and transfer of pixel information from the graphics buffer to the CPU.

The evaluation window enables calculations of properties in regions lying outside the viewport required for certain evaluations and partial fitness results (e.g. what percentage of an object is projected inside and outside of the viewport).

Visibility

Occlusion constraints are evaluated first over a hierarchical bounding sphere approximation of the polygonal objects, followed by an optimized visible-surface determination algorithm. The visible-surface algorithm takes advantage of the fact that disjoint occluding objects from the occluded object need not determine their polygon depth order, only those objects which intersect the occluded objects bounding spheres in 3D-coordinates need a closer z-buffer determination of which polygons are visible.

An optimised procedure serves to report back two variables (the number of projected pixels, and the number of visible pixels for the object/s) from which the variety of occlusion properties can be evaluated.

Initially, a pre-computation step determines which parts of objects may mutually overlap other parts of objects (i.e. objects whose polygons reside in each others bounding convex hull). These objects are stored in each object's list of potentially overlapping objects whose visible-surface needs to be determined by the z-buffer.

The calculation of these occlusion variables is performed in 3 stages. In the first stage we determine the number of potentially occluding objects. This is done by performing a hierarchical bounding sphere operations over the occluded object and tagging any objects whose lowest-level bounding spheres overlap the lowest-level bounding spheres of the objects for evaluation. If there are no potentially occluding objects, then the algorithm returns with value fully unoccluded.

The second stage clears the frame-buffer (with value 0) and renders the objects to evaluate to the frame buffer using flat-shaded polygons of uniform color (which is greater than zero). The frame-buffer is then read and each pixel with value greater than zero is counted. This gives the projected pixel count of the objects.

The third stage renders the potentially occluding objects with color value of zero to the frame buffer. The list of occluding objects which intersect the occluded objects are rendered with depth-testing enabled. Visible-surface determination is necessary only between the evaluated occluded objects and the occluding objects, hence depth-testing is disabled for

the rest of the potentially occluded objects since they have already been proven disjoint and lie closer to the viewplane.

The pseudocode for this algorithm is shown below:

1. Generate a list of the potentially occluding scene elements, using pre-computed intersection sets and hierarchical bounding sphere occlusion determination.
2. If number of potentially occluding objects > 0
 - 2.1. Clear frame-buffer
 - 2.2. If intersection occluders: enable z-depth
Else: disable z-depth
 - 2.3. Render evaluation objects with color value 1.
 - 2.4. Count the total projected number of pixels T
 - 2.5. Render intersection occluders with value 0
 - 2.6. Render disjoint occluders with z-depth comparison disabled with color value 0.
 - 2.7. Count visible pixels V with value > 0
 - 2.8. Return unoccluded = V / T
3. Else return unoccluded = 1.0

The algorithm exits early if the shot property has already been satisfied (i.e. more than a value specified in the shot property). Variance between evaluation resolutions of 32×32 pixels and 640×480 resolutions has been found in most cases to be insignificant, although for deliberately very low values (and tolerances) this is a factor that remains to be addressed.

More involved visible-surface determination techniques (such as advanced culling techniques using octrees) could improve the rendering speed of objects to the frame-buffer. However, the current bottleneck for these operations is the transfer of pixels from the graphics card buffer to the CPU. Here we have two buffer transfers and the threshold at which the graphic card improves upon software rendering is at about 5000 polygons per scene and does not degrade significantly for scenes comprising up to a hundred thousand polygons.

Image Generation

Given the selection of the shot properties, we must produce an image that maximizes the fulfillment of these objectives. This is cast as a standard optimization problem. At present only the camera state is modified to achieve the final output. Possibilities for modifying lighting or introducing cut-away shots of objects (to achieve further visual properties) will be applied in a future extension.

CAMPLAN uses a genetic algorithm (a non-deterministic optimization method) to find the optimal camera position whereby all seven elements of the camera state vector (position/3, angles/3 and field of view) are encoded in the chromosomes of each gene.

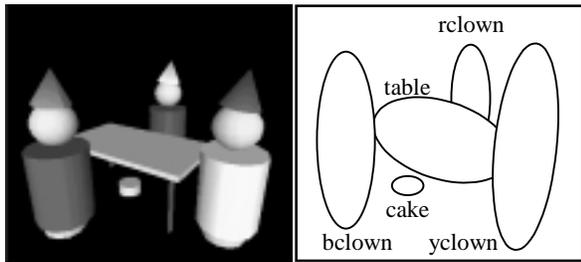


Figure 1. Test scene and names of scene elements.

A population of cameras are randomly distributed in a pre-specified bounded search space. A fitness value, derived from a linear combination of normalized values returned from the evaluation of each fitness function, is computed for each camera state. The top 90% of the population survive to the next generation, and the remaining 10% are re-generated by random crossover and/or mutation of chromosomes. Each successive generation is produced until the user aborts the search or an optimal combination of chromosomes is found for the camera state.

The fitness functions pertains to each shot property and is evaluated in sequence given each new gene (camera state). In order for a gene to survive to the next generation it must give a higher fitness value than the existing top ten percent of the population. This allows the program to check the current accumulated progress of the shot properties for the gene and if it is not possible for the gene to survive to the next generation then all remaining shot evaluations may be evaluated under cruder, more efficient algorithms. Shot properties can be arranged such that the most computationally expensive evaluations are executed last (e.g. OccludedInViewport).

System Evaluation

The evaluation of camera planning systems is difficult and often prone to the subjective biases of the evaluators. However, CAMPLAN may be evaluated according to two criteria: (1) the use of shot properties to achieve communicative goals, and (2) the performance of the optimization of shot properties.

Evaluation of the use of shot properties requires a higher-level goal and pertains more to the eventual applications of CAMPLAN. We need applications that attempt to establish and generate a communicative goal or effect and apply the image functions currently available in CAMPLAN to determine if they are sufficient to achieve the specified goal. We have previously demonstrated that stylistic consistency can be achieved when applying the same set of image functions across object isomorphisms in scenes with similar spatial properties (Olivier et al., 1999).

The first set of the following examples observe CAMPLAN's satisfaction of progressively more restrictive sets of image properties for the emergence of common styles. Next, we demonstrate that

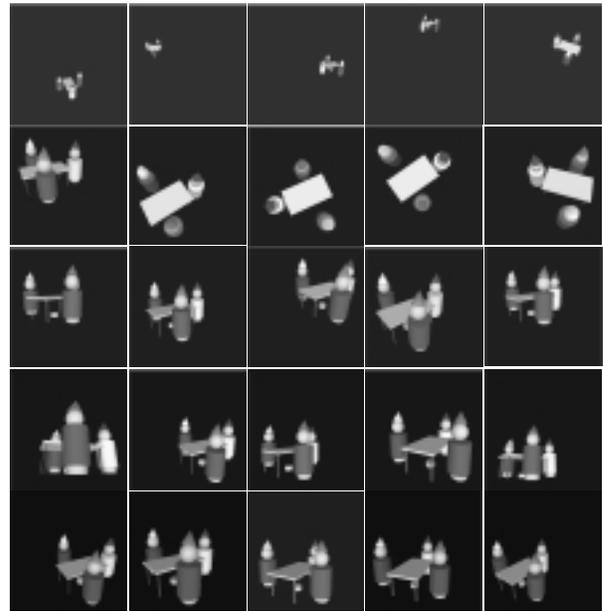


Figure 2. Each horizontal row corresponds to the output for a progressively more restrictive set of shot properties

CAMPLAN is able to find a solution in a tightly constrained space and that it can provide a natural way of describing a view to generate an image. We also look at how well the genetic algorithm integrates into the system at this stage of the project.

Progressive restriction

Here we consider the output of CAMPLAN for the simple scene shown in figure 1 for which solutions are found for a set of shot properties that is progressively restricted. For each set of properties the first five solutions are shown in figure 2.

The first set of properties require only that all objects be contained in the viewport. The location of the camera may take any value within the search space, which is large compared to the dimensions of the objects in the scene, thus it is predictable that the scene elements will most likely occur in the distance in the resulting output images.

The undesirably small size of the scene elements in the first row can be addressed by introducing a constraint on the size of a particular element in the scene (e.g. the largest) to have certain dimensions in the screen space. The second row shows the newly generated images with the additional fact that the table is required to be between 40% and 60% of the viewport width. The scene elements now appear larger although there remains significant variability both in the direction from which the scene is viewed, and in the positions of the scene elements in the viewport.

We can restrict the orientation of the view by requiring one of the clowns to be closer to the camera than the two others. The resulting images, shown in the third row, have an added degree of compositional regularity. However, there is still some variation in

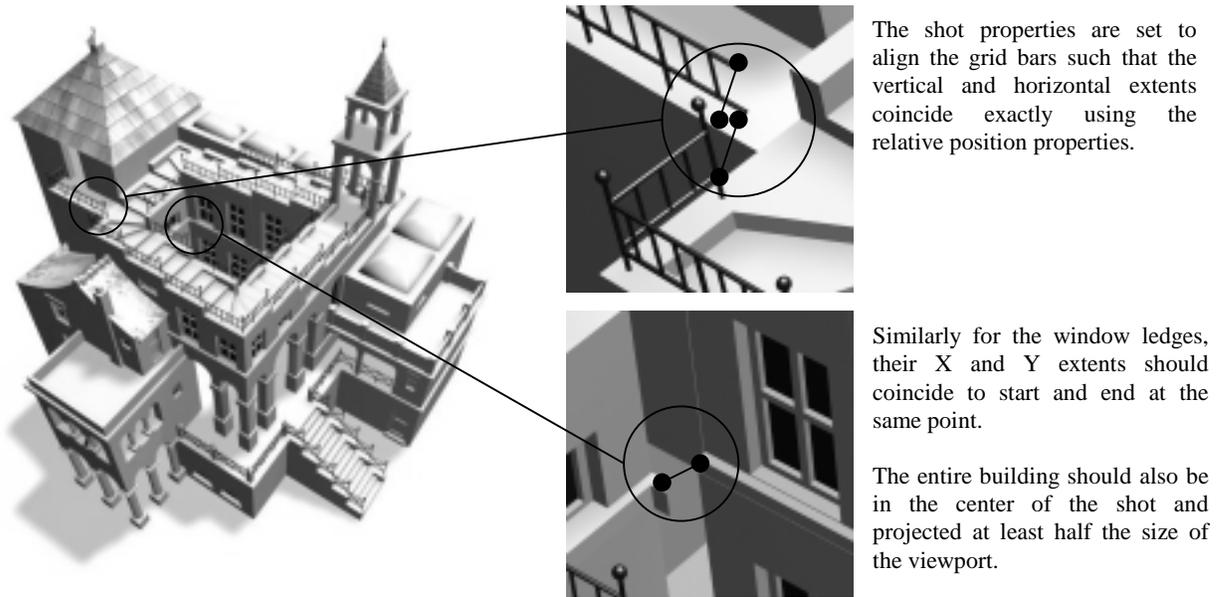


Figure 3. Camera state found in a unique solution region which satisfies a set of tightly constrained properties

the position of the elements in the viewport. By additionally specifying that the coordinates of the table must lie in the lower half of the viewport, the position of scene elements are further restricted (as shown in the fourth row).

Observing the images shows that some objects may still be fully, or nearly fully, occluded by others. The final objective is the removal of the possibility that any of the scene elements are more than 80% occluded. When applied to the scene typical image results are shown in the bottom row of figure 2.

Escher scene

We have demonstrated the system's performance on a more complex scene (a virtual reconstruction of Escher's "Ascending and Descending") comprising over 60,000 polygons. Figure 3 shows how CAMPLAN has found a tight unique solution region over a set of properties designed to produce the correct perspective view of the Escher building by aligning certain visual keys. Note that an additional constraint was used so that the center of the building's bounding sphere is forced to be located in the center of the view. This removes 3 degrees of freedom (the camera orientation is fixed to face a target given its position) and the convergence of the GA towards a solution is improved dramatically. CAMPLAN is able to find a solution on average within a few seconds. Without forcing the camera to focus on the center of the building the GA converges much slower (taking minutes or hours). This suggests that removing degrees of freedom and introducing methods to restrict the search space will result in significant improvements for the search time.

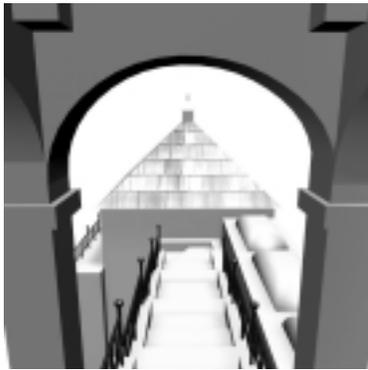
Another set of properties demonstrates how the user can specify a view in a natural way. Figure 4 shows a view where the camera was desired to be placed in a position to give an unoccluded shot of the roof of the

building leading to the steps as seen between two pillars of the small tower. The model itself has a grid fence which is discontinuous, and the user may hide this by specifying that this grid should be overlapped by the pillar which is to be placed to the left.

To show the advantages of a declarative approach to shot specification, and in order to give a clearer shot of the models construction and illustrate how the illusion of the infinitely ascending and descending steps is created, a small number of properties breaking the main features of this illusion are specified. The two steps which are disjoint in the cycle should be in some part visible, and that they should not overlap each other. A typical result of the images produced in satisfying these properties is shown in figure 5.

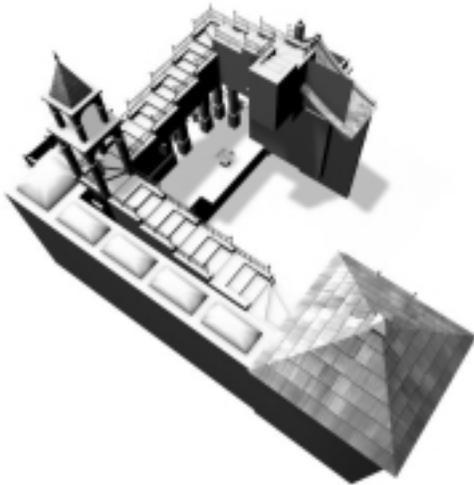
Genetic algorithm

Although little experimentation has been conducted with respect to the configuration of the genetic algorithm, its strength and practicality at this stage can not be understated. For example, modification to an existing image function or the inclusion of additional shot properties can be achieved by simply defining the evaluation function for the property, without concern for the optimization process. In fact, GA functions work well with crude evaluation values because the progressing population is based upon relative discrepancies between gene fitness values. Although no deterministic time estimate is possible for the duration of the search (our examples typically take between 3 seconds and 5 minutes), we have found it to be a reliable method that (eventually) finds the optimum camera state to satisfy the fitness functions. As the image properties become more established, we hope to integrate alternative solution search methods with the aim of achieving more efficient mechanisms for acquiring the desired camera state. The drastic



```
CenterShot "roof"
Unoccluded "roof" 100 tolerance 0
SpanXorY "roof" 2 tolerance 1.2
PositionRelativeX "pillar01" before before "roof"
PositionRelativeX "pillar03" after after "roof" "grid"
OccludedInViewPortBy "pillar01" 100 tolerance 99
```

Figure 4. Image generated by directly specifying the desired compositional properties of a view



```
CenterShot "Escher building"
EntirelyInViewPort "Escher building"
OccludedInViewPort "step high" 0 80
OccludedInViewPort "step low" 0 80 "step high"
NotOccludedBy "step low" "step low"
NotOccludedBy "step high"
```

Figure 5. Highlighting the relationship between the top and bottom step.

decrease in time required to generate a solution by restricting the GA search space highlights that progress in this direction will most likely produce the most promising results.

Current and Further Work

Current work is largely focused on the technical aspects of CAMPLAN as this forms the basis upon which everything else will be constructed. Optimization implementations currently undertaken exploit OpenGL 3D graphics hardware acceleration, level-of-detail structures, and more sophisticated hierarchical occlusion and frustum culling. Alternative solution methods or optimization strategies to genetic algorithms are also being investigated. In particular, partitioning of the solution space into potentially satisfiable regions based on the selected shot properties, reducing the degrees of freedom in the search space, and more intelligent search methods given known properties are expected to give the most significant performance boost by many orders of magnitude.

Concurrently, we are considering enlarging the current set of image functions placing more emphasis on empirical motivations for the image functions. Higher level issues such as style rules and cinematic idioms can then be established. Additional features such as moving cameras and including lighting models

to allow image functions for visual effects such as shadows are under development.

References

- James F. Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26:832-843, 1983.
- William Bares, Joel Gregoire, and James Lester, Real-time Constraint-Based Cinematography for Complex Interactive 3D Worlds, Proceedings of the Tenth National Conference on Innovative Applications of Artificial Intelligence, pp. 1101-1106, Madison, Wisconsin, July 1998.
- William Bares and James Lester, Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds, Proceedings of the International Conference on Intelligent User Interfaces (IUI-99), pp. 119-126, Los Angeles, 1999.
- Jim Blinn, Where are I? What an I looking at? *IEEE Computer Graphics and Applications*, 8(4):76-81, 1988.
- David B. Christianson, Sean E. Anderson, Li-wei He, David Salesin, Daniel S. Weld, Michael F. Cohen: Declarative Camera Control for Automatic Cinematography. Proceedings of the National Conference on Artificial Intelligence (AAAI), Vol. 1, pp. 148-155, 1996.

- Steven M. Drucker, Intelligent Camera Control for Graphical Environments, PhD Thesis, Media Lab., MIT, 1994.
- Nicolas Halper, Camera Planning for Polyhedral Worlds, MEng Dissertation, Department of Computer Science, University of York, 1999.
- Patrick Olivier, Jon Pickering, Nicolas Halper and Pamela Luna, Visual composition as optimisation, AISB Symposium on AI and Creativity in Entertainment and Visual Art, Edinburgh, 9th April, pp. 22-30, 1999
- Doree D. Seligmann, Interactive Intent-Based Illustrations: Visual Language for 3D Worlds, PhD Thesis, Department of Computer Science, Columbia University, 1993.
- Wolfgang Wahlster, E., André, H.J., Profitlich and T., Rist. Plan-based Integration of Natural Language and Graphic Generation. Artificial Intelligence, 63:387-427, 1993.
- Zhou, M.X., and Feiner, S.K. Efficiently Planning Coherent Visual Discourse, Journal of Knowledge-Based Systems, 10(5):275-286, March, 1998.