A Model for Constraint-Based Camera Planning

William H. Bares Somying Thainimit Scott McDermott
Center for Advanced Computer Studies
University of Louisiana at Lafayette
PO Box 44330
Lafayette, LA 70504-4330
whbares@cacs.usl.edu

Abstract

Automatically planning camera shots in virtual 3D environments requires solving problems similar to those faced by human cinematographers. In the most essential terms, each shot must communicate some specified visual message or goal. Consequently, the camera must be carefully staged to clearly view the relevant subject(s), properly emphasize the important elements in the shot, and compose an engaging image that holds the viewer's attention. The constraint-based approach to camera planning in virtual 3D environments is built upon the assumption that camera shots are composed to communicate a specified visual message expressed in the form of constraints on how subjects appear in the frame. A human user or intelligent software system issues a request to visualize subjects of interest and specifies how each should be viewed, then a constraint solver attempts to find a solution camera shot. A camera shot can be determined by a set of constraints on objects in the scene or on the camera itself. The constraint solver then attempts to find values for each camera parameter so that the given constraints are satisfied. This paper presents a work in progress snapshot of the virtual camera constraint model that we are currently developing.

Introduction

Automatically planning camera shots in virtual 3D environments requires solving problems similar to those faced by human cinematographers. In traditional cinematography parlance, a *shot* refers to a continuous stream of *frames* (individual images) recorded by a given camera. In the most essential terms, each shot must communicate some specified visual message or goal. Consequently, the camera must be carefully staged to clearly view the relevant subject(s), properly emphasize the important elements in the shot, and compose an engaging image that holds the viewer's attention (Alton 1947, Katz 1991, Mascelli 1965, Millerson 1994). Given a director's shot request in a *storyboard* (sketch of the frame) or in writing (e.g. medium profile shot), the *camera operator* must position the camera to satisfy the given "constraints" in the context of real-world physical settings, frequently beyond the camera operator's control (Hines 1997). The constraint-based approach to camera planning in virtual 3D environments is built upon the assumption that camera shots are composed to communicate a specified visual message. As observed by Drucker, it would often be more efficient to specify what should appear in the camera shot and have an "intelligent camera module" compute the shot rather than having to manipulate camera controls to obtain the desired shot (Drucker 1994). The desired visual message of a camera shot is expressed in the form of constraints on how subjects appear in the frame. A human user or intelligent software system issues a request to visualize particular subjects of interest and specifies how each should be viewed, then a constraint solver attempts to find a solution camera shot. A camera shot can be determined by a set of constraints on objects in the scene or on the camera itself. The constraint solver then attempts to find values for each camera parameter so that all given constraints are satisfied.

A general-purpose camera constraint system should support the following design guidelines if it is to be effective in a wide-range of applications and virtual 3D environments. In some cases, specific applications may permit simplifications of these guidelines.

- *Arbitrary Viewing Goals:* At any time, viewers or software modules may request views of any set of subjects, and they may do so by stipulating *constraints* on how each subject should appear in the shot. These viewing goals may be specified with respect to user preferences or user task performance needs.

- *Environmental Complexity:* Objects are arranged in non-trivial configurations, and they may feature high geometric complexity, holes, or transparent surfaces. Furthermore, the virtual camera planner should be able to analyze this environment to find a solution.

- *World Non-Interference:* The virtual world should not be modified to simplify the visualization problem. In some cases, out of position objects may be moved (Li-wei He *et al.* 1996) or occluding objects may be "cutaway" (Feiner and Seligmann 1992).

- *Failure Handling:* Systematic methods are needed to produce "next-best" camera shot(s) when a satisfactory one is not possible so that the viewer can be given an acceptable view even under unforeseen difficulties

Current State of the Art

Most of the 3D virtual environment applications deployed today rely on either manual camera control or very rudimentary methods of automated control. Typical virtual camera control methods include first-person point-of-view shots, switching the camera between several predefined shots, moving the camera in lock-step to track a moving object, flying the camera along motion paths, or rotating the camera to look around obstructions.

Prior research works in automated virtual cinematography are relatively recent and represent a brief series of notable contributions. One family of systems employs camera positions pre-specified relative to the subject(s) being viewed (Feiner 1985, Seligmann and Feiner 1991, André *et al.* 1993, Bares and Lester 1997, Karp and Feiner 1993). This approach fails when the camera must view arbitrary combinations of subjects with specific goals or constraints, or when unanticipated motion or obstructions occlude the subjects of interest. IBIS can overcome viewing failures by using multi-view illustrations and cutaways of occluding obstructions, and CATHI has a facility for transparency (Butz 1997, Feiner and Seligmann 1992).

Idiom-based systems encode knowledge of cinematography to sequence shots of commonly occurring actions such as conversations between small groups of players. Idioms may be encoded using Finite State Machines to designate shot transitions (Christianson *et al.* 1996, Li-wei He *et al.* 1996). Or, idioms may be encoded in the form of film grammars, which direct a top-down planner in generating a sequence of shots (Butz 1997, Karp and Feiner 1990, Karp and Feiner 1993). Existing idiom-based systems use variants of the pre-specified relative camera method in lieu of more complex camera placement solvers. The Virtual Cinematographer employs Blinn's method to stage the camera to project an object to a given point in the frame (Blinn 1988). Consequently, such idiom-based systems can often fail to find acceptable shots when multiple subjects occupy unanticipated relative spatial configurations, or structures in the world occlude the subjects of interest, or users wish to view specific types of shots not encoded by the idioms.

In contrast, the constraint satisfaction approach to automated camera planning casts viewing goals as constraint satisfaction problems, which are then solved by a constraint solver to compute camera placements. We have adopted this approach due to it's versatility and natural correspondence to the way human cinematographers specify what they want to see in each shot using storyboards or notes and leave the job of actually placing the camera to an operating cinematographer (Hines 1997). CAMDROID, the first major work in this area, supports a broad and powerful set of camera constraints, but employs a numerical constraint solver that is subject to local minima failures (Drucker 1994, Drucker and Zeltzer 1994, Drucker and Zeltzer 1995). The CAMPLAN system utilizes genetic algorithms to intelligently generate-and-test candidate camera shots by "mating" those sets of camera parameters which best satisfy the constraints of the desired shot. This system supports thirteen constraint types including shot distance, view angle, occlusion, and projection location and relations between objects (Patrick Olivier *et al.* 1998). A constraint-solver developed by the author provides a systematic solution for handling constraint failures that occur in non-trivial virtual environments. However, this initial constraint-based camera planner supported only four types of constraints including viewing angle, distance, occlusion-avoidance, and camera inside an enclosed space (Bares *et al.* 1998, Bares and Lester 1999). This initial effort has served as the starting point for the ongoing work described in this paper.

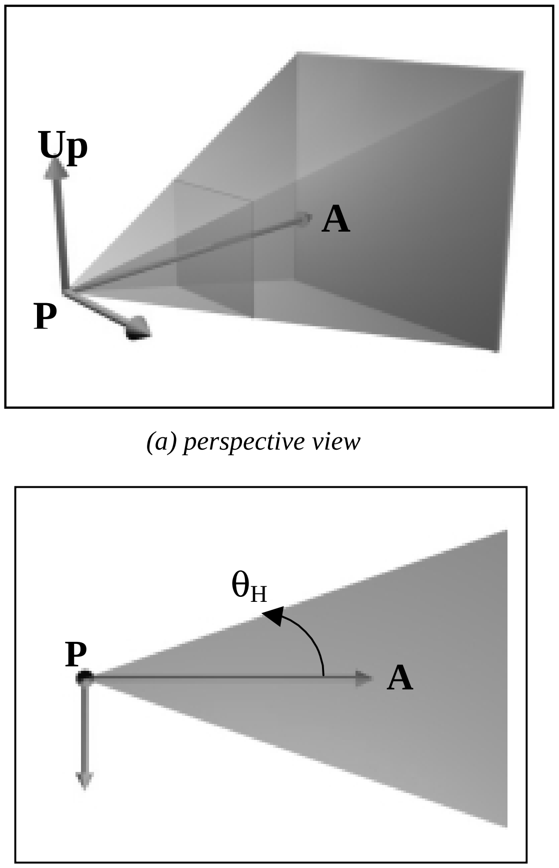
Automatic camera control assistants vary camera position to avoid occlusions of a goal object or satisfy screen-space constraints on how subjects appear on-screen (Gleicher and Witkin 1992, Phillips *et al.* 1992, Jardillier and Languénou 1998). Automated camera navigation assistants adjust camera motion speed based on distance to the target or guide the camera along specified optimal vantages as a user navigates over a terrain (Hanson and Wernert 1997, Mackinlay *et al.* 1990, Ware and Osborn 1990, Ware and Fleet 1997). Neither automated viewing nor navigation assistants can address the problem of goal-driven camera control because they focus on specific subsets of this much larger problem or on controlling relatively low-level parameters and frequently require considerable user inputs.

Virtual 3D Camera Parameters

A *camera placement* in a virtual 3D environment is defined by the following parameters, most of which are illustrated in Figure 1. A computed constraint solution assigns values to the virtual camera parameters.

- *Camera position* **P**(x, y, z): This is the point at which the virtual camera is located.
- *Aim direction* vector **A**(dx, dy, dz), or *aim point* **P**(x, y, z).
- *Field of view* angles: The horizontal and vertical field of view angles model the effects of the so-called wide angle and telephoto lenses. In computer graphics, the "lens angle" is determined by two parameters, the horizontal field of view angle (θ_H) and vertical field of view angle (θ_V).

- *Up orientation vector* **Up**(dx, dy, dz): This vector controls the degree of tilt or rotation (roll) of the camera about its aim direction vector.



(a) perspective view

(b) Top view

Figure 1: Virtual camera parameters.

Camera Constraints

The camera constraint system being developed supports fifteen different types of constraints on either the camera's attributes or how objects in the scene appear in the camera shot. One or more constraints may be specified to define a camera constraint problem. If constraints C1, C2, ...CN are given, then C1 *and* C2 *and* C3 ... *and* CN must be satisfied to produce a *camera placement* that satisfies the complete set of constraints. Each constraint also includes a specified relative priority with respect to the other given constraints.

Constraints may be applied to one or two objects or to the camera itself. Constraint types beginning with "OBJ" apply to an object in a 3D scene and must specify one "primary" object to which the constraint applies. Some "OBJ" constraints also require that a "secondary" object be specified. For example, the OBJ_DEPTH_ORDER constraint can be used to require that the primary object is closer to the camera than the secondary object. Constraint types beginning with "CAM" apply only to the camera and no objects are specified. For example, the CAM_POS_IN_REGION constraint requires that the camera position lie within the given region of 3D space. Constraint types not beginning with either "OBJ" or "CAM" may apply to either an object or the camera.

Some object constraints, notably those dealing with how an object's projection appears in the frame, can specify that the constraint operate upon a designated point or region of space (bounded by a sphere) displaced from an object's midpoint. This optional construct is designated a *locus modifier*. For example, the LOOK_AT_POINT constraint may aim the camera at a locus modifier point displaced slightly ahead of a moving object. Or, OBJ_PROJECTION_SIZE constraint can include a locus modifier in the form of a sphere that can be positioned and sized to coincide with the head of a humanoid figure to constrain the projection of the head to fill a given fraction of the frame.

Constraint Types

The general format of each constraint specifies a range of allowable values, an optional optimal value, and a value indicating the relative importance of that constraint. Some types of constraints apply to a specific object or pair of objects, designated by the specified primary and secondary objects. Several constraint types permit the specification of additional parameters specific to that type of constraint. The following is a brief summary of several of the currently implemented constraint types.

1. **LOOK_AT_POINT**
The LOOK_AT_POINT constraint causes the camera to aim at a particular point. The aim point may be displaced relative to a designated object or to any given point in global space.

2. **OBJ_IN_FIELD_OF_VIEW**
The OBJ_IN_FIELD_OF_VIEW constraint requires that the specified primary object be entirely within the camera's field of view formed by its left, right, bottom, top, near, and far clipping planes.

3. **OBJ_OCCLUSION_MINIMIZE**
The OBJ_OCCLUSION_MINIMIZE constraint requires that no more than the specified maximum allowable fraction of the given primary object is occluded by other opaque object(s). The given fraction *maxOccl* ranges between 0.0 and 1.0, inclusive. Occlusion fractions are measured using the ratio of occluded pixels to total number

of pixels used to draw an object. For example, if an object's projection covers 1000 pixels of the frame and 250 of those pixels are occluded by an opaque object, then the occlusion fraction is 0.25.

4. **OBJ_VIEW_ANGLE**
The OBJ_VIEW_ANGLE constraint requires the camera to be positioned at a specified orientation relative to the untransformed primary object. The relative orientation of the camera to the object is expressed in spherical coordinates (only the theta and phi angular components are used). The spherical coordinates orientation angles (theta, phi) represent the horizontal and elevation angles respectively. Horizontal angle values range between -180 and 180 degrees. Elevation angle values range between -90 and 90 degrees. Figure 2 provides an example of how the horizontal and elevation angles are specified.

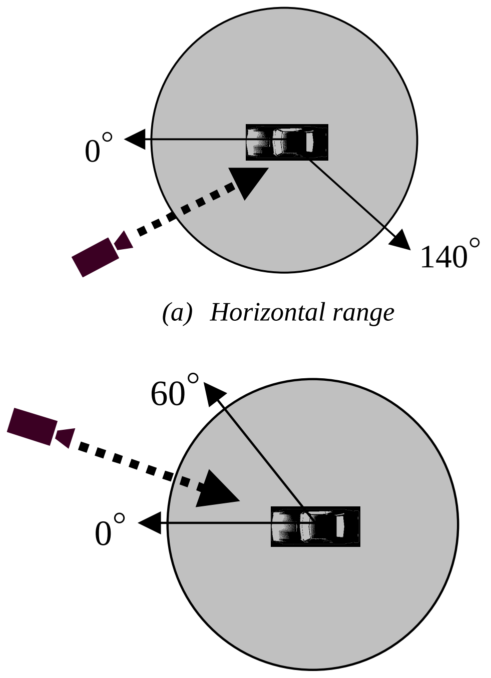


Figure 2. Viewing angle example.

The orientation angles are measured from the object's midpoint (if no locus modifier is given) or from the locus modifier point if a locus modifier is specified.

5. **OBJ_DISTANCE**
The OBJ_DISTANCE constraint requires that the distance from either the object's midpoint or the specified optional locus point to the camera lie within the given allowable minimum-maximum distance range. An optional optimal distance may be specified. The distance from the camera position is measured from the object's midpoint (if no locus modifier is given) or from the locus modifier point if a locus modifier is specified.

6. **OBJ_PROJECTION_SIZE**
The camera's distance to the subject and the field of view angle determine the size at which the subject appears in the frame. Nearer camera positions and narrower lens angles result in larger subject sizes, for example as in the close-up shot. Farther camera positions and wider lens angles result in smaller subject sizes and can include vast expanses of territory as in the extreme long shot above. In cinematic terms, shot "distances" are specified by the on-screen size of the subject(s) relative to the size of the frame. Common shot sizes include close up, medium, and long, arranged in increasing order of distance and decreasing order of subject size (Mascelli 1965). The OBJ_PROJECTION_SIZE constraint requires that the projection of the primary object's source cover a specified fraction of the frame. The object's projection source may be either its BoundingBox or the optional LocusSphere. The primary object's source is projected onto the frame to form a bounding rectangle (in normalized frame coordinates), whose area is then divided by the total area of the frame.

7. **OBJ_PROJECTION_ABSOLUTE**
Project a given portion of an object to a given region of the 2D frame. This projection region is expressed in normalized frame coordinates. The 3D source of projection may be a point, sphere, or bounding box. The projection source parameter value may be Center, LocusPoint, LocusSphere, or BoundingBox. If the source is "Center," then the object's midpoint is projected. The 3D source point, sphere, or bounding box, is to be projected so that it lies inside the given rectangular region in normalized frame coordinates.

8. **OBJ_PROJECTION_RELATIVE**
The projection of the primary object has the specified relationship to the projection of the secondary object in the frame. The projection relationship is expressed in terms of the relative positions of the primary and secondary objects in the frame. The 3D source of projection for both objects can be a point, sphere, or bounding box. The projection relationship between the primary and secondary objects may be one of the following:

Left: The projection of the primary object's source lies entirely to the left of the projection of the secondary object's source.

Right: The projection of the primary object's source lies entirely to the right of the projection of the secondary object's source.

Above: The projection of the primary object's source lies entirely above that of t1he secondary object.

Below: The projection of the primary object's source lies entirely below that of the secondary object.

Inside: The projection of the primary object's source lies entirely inside of the projection of the secondary object's source. The projection source must be either LocusSphere or BoundingBox.

Outside: The projection of the primary object's source lies entirely outside of the projection of the secondary object's source. The projection source must be either LocusSphere or BoundingBox. Informally, this constraint means that the projections of the two objects not overlap.

DirectionVector: The vector from the center of the primary object's projection source to the center of the secondary object's projection source must have the specified direction in normalized frame coordinates. An amount of tolerance is specified by the *Max Angle* parameter. The constraint is satisfied if the angle between these two vectors is less than or equal to the given threshold.

9. **OBJ_DEPTH_ORDER**
The OBJ_DEPTH_ORDER constraint requires that the given primary and secondary objects have the specified depth (or distance) relationship with the camera. Relationships include NearerThan, FartherThan, and SameDistance. For example, the NearerThan constraint is satisfied if the midpoint of the primary object is nearer to the camera than the midpoint of the secondary object.

10. **CAM_POS_IN_REGION**
This constraint requires that the camera position lie within the specified region of Three Dimensional space. Currently, this region is specified by the minimum and maximum endpoints of an axis-aligned bounding box.

11. **CAM_FIELD_OF_VIEW**
This constraint sets the allowable ranges on the camera's horizontal and vertical field of view angles in addition to the aspect ratio of the frame. It's necessary to also specify limits on the aspect ratio since allowable values of horizontal and vertical field of view angles can combine to undesired aspect ratios. Wide field of view angles are used to include as much of a subject's surroundings as possible or to reduce the subject's size in the frame. Narrow field of view angles are useful for eliminating clutter around a subject or filling the frame with the subject.

Measuring Constraint Satisfaction

When constraints are evaluated or solved, it's essential to determine how well a given camera placement satisfies each constraint. The degree of satisfaction for each constraint is computed as a value in the range 0.0 to 1.0. Fractional values are useful in distinguishing which satisfactory camera placement values are nearest to the specified optimal values. A cumulative constraint satisfaction rating is computed from a weighted sum of the satisfaction measures of all individual constraints as shown in the below equation, where P_i is the specified relative priority of the i<u>th</u> constraint and S_i is the satisfaction rating of the i<u>th</u> constraint.

$$satisfaction = \sum_{1}^{N} \left(Pi \times Si \right)$$

Constraint Solutions

The initial prototype of the constraint solver employs an exhaustive generate-and-test process to determine the camera placement having the highest cumulative constraint satisfaction rating. Candidate camera placements are generated via iteration in discrete steps over all possible values of camera position, aim direction, and field of view angle. The prototype exhaustive search solver serves as a testbed for designing the constraint types and constraint-satisfaction evaluators. We are currently investigating other more efficient constraint solution algorithms.

Example

The constraints can be used to compose camera shots similar to those found in motion pictures. One of the most common types of shots involves filming two persons in conversation. For example, we might want a shot of player1's face as he speaks, in which player1 is viewed from a camera position behind and over-the-shoulder of player2. Consider the sample 3D scene populated by four players standing around a table in a room ringed by stone columns (Figure 3). The virtual camera will need to be carefully staged in order to capture the desired over-the-shoulder shot of player1 and player2.

In order to compose the desired over-the-shoulder shot of player1 and player2, we could specify the following set of camera directions or constraints. The constraint type used to implement each directive is given in parentheses.

- View the face of player1 who is speaking (Object view angle).
- Player1 should be farther from the camera than player2 (Object depth order).
- Player1 should appear slightly to the right and above player2 (Object projection relation expressed by a given direction vector).
- Player1 should appear in the middle of the frame so as to be the center of focus (Object projection location).
- Player1 and player2 should be framed at a medium shot size (Object projection size).
- Player1 and player2 should not be occluded (Object minimize occlusion).

Figure 4 illustrates a shot that partially satisfies the shot constraints. However, a column occludes player2.

Figure 5 presents a satisfactory shot computed by the constraint solver prototype searching camera positions over a 20x20x20 grid, aim directions at 15° intervals, and field of view angle over a range of 10 values. A total of 13,891,500 shots were evaluated in approximately 30 minutes on a Pentium II 400 MHz computer to determine this solution shot, which had a cumulative constraint success rating of 89%.

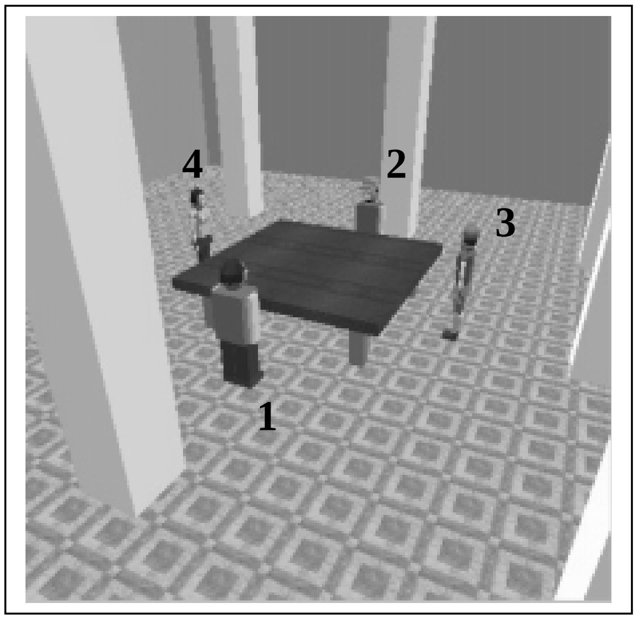


Figure 3: Room interior scene featuring four players.

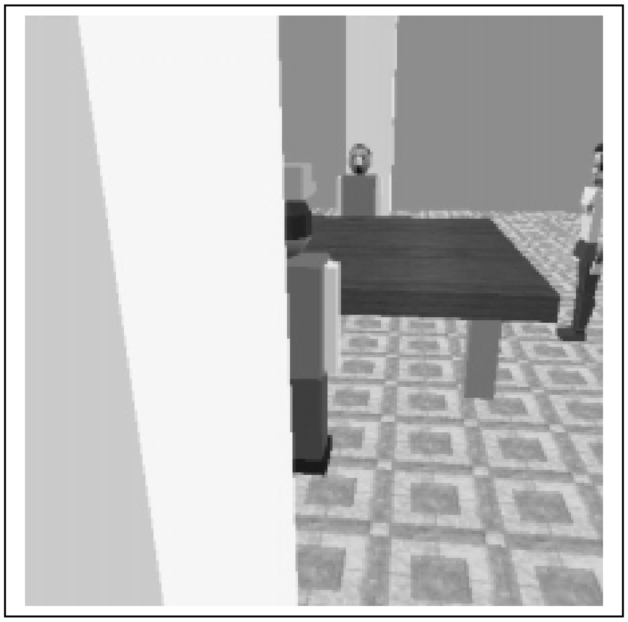


Figure 4: Unsatisfactory over-the-shoulder shot.

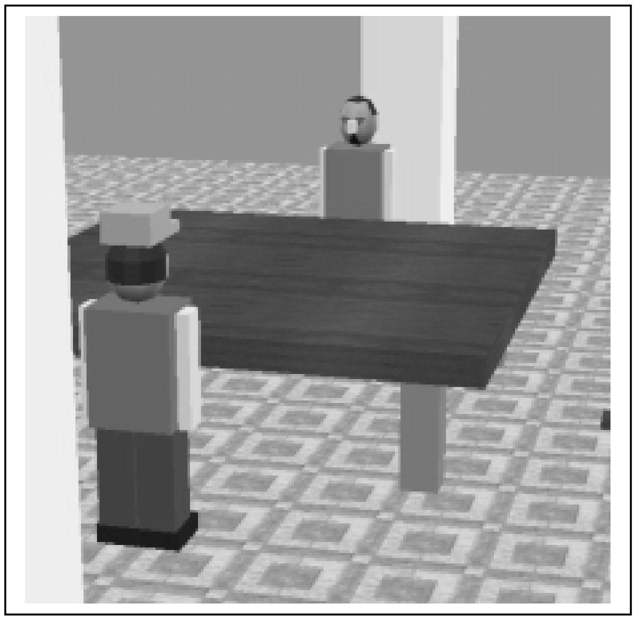


Figure 5: Successful over-the-shoulder shot.

A portion of the constraint input script used to create the shot in Figure 5 appears in the code listing in Figure 6 on the following page.

```
Constraint CAM_FIELD_OF_VIEW
{
 Parameters {
 minVert 9.75 optVert 30.0 maxVert 78.6
 minHoriz 9.75 optHoriz 30.0 maxHoriz 78.6
 minAspect 1.0 optAspect 1.0 maxAspect 1.0 }
 Priority 1.0
}

Constraint OBJ_IN_FIELD_OF_VIEW
{  PrimaryObj 1        Priority 1.0 }

Constraint OBJ_PROJECTION_SIZE
{
 PrimaryObj 1
 Parameters {
 Source BoundingBox
 MinSize 0.017456   OptSize 0.0400  MaxSize 0.117181 }
 Priority 1.0
}

Constraint OBJ_PROJECTION_ABSOLUTE
{
 PrimaryObj 1
 Parameters {
 Source BoundingBox
 BottomLeft 0.500115 0.327080
 TopRight 0.972977 0.988225 }
 Priority 1.0
}

Constraint OBJ_VIEW_ANGLE
{
 PrimaryObj 1
 Parameters {
 optHoriz 15.591192
 optElev 19.223467
 AllowedHorizRange -20.408808 51.591194
 AllowedElevRange 1.223467 37.223469 }
 Priority 1.0
}
```

Figure 3: Constraint input script fragment.

Future Work

We are in the process of developing more efficient solution search algorithms that can compute solutions of comparable quality to the impractical exhaustive search by examining a carefully selected subset of the space of all possible camera parameters. The current constraint system requires the user to edit the constraints in a text script, which is loaded into the constraint solver prototype. Intuitive What You See Is What You Get graphical interfaces for constructing constraint sets could help constraint-based intelligent camera systems to find productive use by visual artists, cinematographers, or interactive 3D software developers. Consequently, we are also developing a visual interface to automatically generate constraint scripts for a desired camera shot.

Acknowledgements

Thanks to visual artists Yeon Choi, Robert Russett, and Patrick FitzGerald for their creative suggestions. Thanks also to James Lester and the North Carolina State University IntelliMedia Initiative for their continued support. This work is partly supported by the Louisiana State Board of Regents through the Board of Regents Support Fund contract LEQSF(1999-2002)-RD-A-48.

References

Alton, John. 1947. Painting with Light. Macmillan, 1947.

André, Elisabeth , W. Finkler, W. Graf, T. Rist, A. Schauder, and W. Walhster. 1993. WIP: The automatic synthesis of multimodal presentations. In M.T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 3, AAAI Press, 1993.

Bares, W. and J. Lester. 1997. Realtime Generation of Customized 3D Animated Explanations for Knowledge-Based Learning Environments. In *AAAI-97: Proceedings of the Fourteenth National Conference on Artificial Intelligence*. Providence, Rhode Island, 1997, pages 347-354.

Bares, W., J. Grégoire, and J. Lester. 1998. Realtime Constraint-based Cinematography for Complex Interactive 3D Worlds. In *IAAI-98: Proceedings of the Tenth Conference on Innovative Applications of Artificial Intelligence.* Madison, Wisconsin, 1998, pages 1101-1106.

Bares, W. and J. Lester. 1999. Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds. In *IUI-99: Proceedings of the 1999 International Conference on Intelligent User Interfaces*, Los Angeles, California, 1999, pages 119-126.

Blinn, James. 1988. Where Am I? What Am I Looking At? In IEEE Computer Graphics and Applications, July, 1988, pages 76-81.

Butz, Andreas. 1997. Anymation with CATHI. In *IAAI-97: Proceedings of Innovative Applications of Artificial Intelligence*. Providence, Rhode Island, July 1997, pages 957-962.

Christianson, David, Sean Anderson, Li-wei He, David Salesin, Daniel Weld, and Michel Cohen. 1996. Declarative camera control for automatic cinematography. In P*roceedings of the AAAI-96: Proceedings of the Thirteenth National Conference on Artificial Intelligence*, August 1996, pages 148-155.

Drucker, Steven. 1994. *Intelligent Camera Control in Graphical Environments*. Ph.D. thesis, 1994, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1994.

Drucker, Steven and David Zeltzer. 1994. Intelligent Camera Control in a Virtual Environment. In *Graphics Interface '94*, 1994, pages 190-199.

Drucker, Steven and David Zeltzer. 1995. CamDroid: A System for Implementing Intelligent Camera Control. In *1995 Symposium on Interactive_3D Graphics*, pages 139-144, 1995.

Feiner, Steven. 1985. APEX: An Experiment in the Automated Creation of Pictorial Explanations. In *IEEE Computer Graphics & Applications*, September, 1985, pages 29-37.

Feiner, Steven and Dorée D. Seligmann. 1992. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. In *The Visual Computer*, August 1992, pages 292-302.

Gleicher, Michael and Andrew Witkin. 1992. Through-the-lens camera control. In Edwin E. Catmull, editor, *Computer Graphics (Proceedings of SIGGRAPH '92)*, 1992, pages 331-340.

Hanson, Andrew and Eric Wernert. 1997. Constrained 3D navigation with 2D controllers. In *Proceedings of IEEE Visualization '97*, 1997, pages 175-182.

Hines, William. 1997. Operating Cinematography for Film and Video: A Professional and Practical Guide. Ed-Venture Publishers, Los Angeles, California, 1997.

Jardillier, Frank and Eric Languénou. 1998. Screen-Space Constraints for Camera Movements: the Virtual Cameraman. In *Eurographics 1998 Computer Graphics Forum*, 17(3), 1998.

Karp, Peter and Steven Feiner. 1990. Issues in the Automated Generation of Animated Presentations. In *Graphics Interface '90*, 1990, pages 39-48.

Karp, Peter, and Steven Feiner. 1993. Automated Presentation Planning of Animation Using Task Decomposition with Heuristic Reasoning. In *Graphics Interface '93*, 1993, pages 118-126.

Katz, Steven. 1991. *Film Directing Shot by Shot*. Studio City, CA, Michael Wiese Press, 1991.

Li-wei He, Michael F. Cohen, and David H. Salesin. 1996. The virtual cinematographer: A paradigm for automatic realtime camera control and directing. In *Computer Graphics (Proceedings of SIGGRAPH '96)*, 1996, pages 217-224.

Mackinlay, Jock, S. Card, and G. Robertson. 1990. Rapid controlled movement through a virtual 3D workspace. In *Proceedings of ACM SIGGRAPH '90*, pages 171-176, 1990.

Mascelli, Joseph. *The Five C's of Cinematography*. Silman-James Press, Los Angeles, California, 1965.

Millerson, Gerald. 1994. *Video Camera Techniques*. Focal Press, Oxford, England, 1994.

Olivier, Patrick, Nicolas Halper, Jon Pickering, and Pamela Luna. 1998. Visual Composition as Optimisation. In Artificial and Simulation of Intelligent Behavior (AISB) Workshop '99 Symposium on AI and Creativity in Entertainment and Visual Arts. Edinburgh, UK, April 1998, pages 22-30.

Phillips, Cary B., Norman Badler, and John Granieri. 1992. Automatic viewing control for 3D direct manipulation. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, March 1992, pages 71-74.

Seligmann, Dorée, and Steven Feiner. 1991. Automated Generation of Intent-Based 3D Illustrations. In *Computer Graphics*, July 1991, pages 123-132.

Ware, Colin, and S. Osborn. 1990. Exploration and virtual camera control in virtual three-dimensional environments. In *1990 Symposium on Interactive 3D Graphics*, pages 175-184, 1990.

Ware, Colin and Daniel Fleet. 1997. Context Sensitive Flying Interface. In *Proceedings of the Symposium on Interactive 3D Graphics*. Providence, Rhode Island, 1997, pages 127-130.