A Lightweight Intelligent Virtual Cinematography System for Machinima Production

David K. Elson[1], Mark O. Riedl[2]

[1]Columbia University, Computer Science Department,
New York City, New York, USA
[2]University of Southern California, Institute for Creative Technologies,
13274 Fiji Way, Marina Del Rey, California, USA
delson@cs.columbia.edu; riedl@ict.usc.edu

Abstract

Machinima is a low-cost alternative to full production filmmaking. However, creating quality cinematic visualizations with existing machinima techniques still requires a high degree of talent and effort. We introduce a lightweight artificial intelligence system, Cambot, that can be used to assist in machinima production. Cambot takes a script as input and produces a cinematic visualization. Unlike other virtual cinematography systems, Cambot favors an offline algorithm coupled with an extensible library of specific modular and reusable facets of cinematic knowledge. One of the advantages of this approach to virtual cinematography is a tight coordination between the positions and movements of the camera and the actors.

Introduction

Narrative is a powerful modality for communication, especially when realized in a visual medium. Creating a film, however, is a costly and time-consuming endeavor that requires trained writers, producers, directors, actors, editors, and others. Recently, hobbyists have turned to off-the-shelf software for mitigating many of these costs. A portmanteau of machine and cinema, *machinima* refers to the innovation of leveraging video game technology to greatly ease the creation of computer animation. Rather than building complex graphical worlds, machinima artists carefully manipulate the behavior of 3D games. By choreographing their characters as avatars, they can "perform" for a player whose perspective represents the camera, record what the camera player sees, and edit the clips into a narrative film (often adding dubbed dialogue).

Some game manufacturers have embraced machinima by adding authoring tools in recent versions. For example, machinima creators can sometimes replace the art assets of a game with character models, sets, and animations more suitable to their narrative settings. Additionally, some games provide special modes for scripting camera angles.

Despite tools to support machinima production, it is still complicated and costly to design and produce cinematic narratives. Authors must, through the constraints of software, manually position cameras and subjects. For example, the LEADERS project (Gordon et al., 2004) used computer-controlled avatars in a virtual environment and cinematic camera shots to create an interactive leadership development application centered around a fictitious military overseas food distribution operation. LEADERS alternated between non-interactive "cut scenes" and decision points where the user could affect the direction of the story. The use of machinima simplified the creation of the cut scenes by avoiding high-cost film production. However, choreographing and encoding the cinematic camera shots still required over $800 in labor costs per minute of machinima video (Gordon, personal communication).

There are two central challenges to machinima creation. First, a script must be created that describes dialogue, movements and gestures for computer-controlled avatars to perform in a video game environment. Second, the script actions must be visualized as an aesthetically coherent visual narrative, rendered as a 2D projection of activity occurring within a 3D graphical environment.

In this paper we focus on the problem of automatically selecting camera angles to capture the action of an *a priori* unknown script as aesthetically appropriate cinema. There are a number of challenges therein, including visual composition, occlusion avoidance and coordination between virtual camera and computer-controlled avatars. We describe a virtual cinematography system, Cambot, which acts as a virtual director.

Background

The Model Process: Real Filmmaking

In order to computationally craft aesthetically acceptable movies, we closely modeled our approach after that of actual filmmaking; in particular, the simultaneous solving of several types of interconnected constraints. In this section, we describe the pertinent aspects of the real-world process.

```
EXT. KABUL CITY STREET - NIGHT

SERGEANT SMITH, 29 y.o. male, is standing in a
street, gun at his side.  CAPTAIN JONES, 34 y.o
male, approaches him.

                    JONES
       What's your condition, Sergeant Smith?

                    SMITH
    Captain Jones, sir, road Beta One is secure.

We see a few Afghan civilians chatting before them.

                    SMITH (cont'd)
          The city's pretty cold tonight.

                    JONES
   Perez tells me you have a message from a local?
```

Figure 1. A sample fragment of a script, showing location constraints, blocking constraints, and dialogue actions.

Filmmaking begins with a script. Figure 1 shows a script in a format similar to that used in the film industry. The script describes the beginning of a *scene,* in which a segment of action and dialogue takes place in a continuous span of time and in a single location. A *beat* (McKee, 1997) is the smallest divisible segment of a scene, typically encompassing one line of dialogue or a moment of action.

The core of any scene consists of actions and dialogue acts that advance the narrative. The other elements of the script specify how the scene should look and sound:

- **Location.** The *scene heading* ("EXT. KABUL CITY STREET – NIGHT" in Figure 1), restricts the locations in which the director may shoot the scene.
- **Blocking.** The *blocking* of a scene refers to the positions of the actors relative to key points on the set and to one another. In Figure 1, Smith is standing still and observing the street as Jones walks toward him.
- **Viewpoint.** A script may restrict the camera angle (i.e., the *shot)* used to visualize a particular beat. In Figure 1, the script specifies that the camera should at one point focus on what Smith is looking at. In the absence of such constraints, the director is free to use whichever shots he or she sees fit.

Though the director has to satisfy many overlapping constraints, there is a generous "search space" from which he or she can craft the best visualization of the script. There may, for example, be many streets in the world (or a studio lot) that resemble those in Kabul closely enough to satisfy the location restriction. There are also several ways the director can block the actors. It is important to note that these decisions affect one another. Ideally, the director keeps all the constraints in mind while choosing from among any of the options.

For example, Figure 2 shows a set during the filming of a shot; in this case, the movements of the camera and the character are closely coordinated so that the former leads the latter down the street. In order to achieve this shot, it was necessary for the director to select a location with sufficient space for the actor and camera to move, and to

Figure 2. The capturing of a cinematic shot requires coordination between actor and camera placement.

build special tracks. The production crew pulls the camera away from the boy at the same speed at which he walks. For this to succeed, the actor must walk a particular path parallel to the tracks at a consistent pace.

The director cannot finish shooting the scene until he or she has obtained "coverage" of at least one shot for each beat. Gathering more than one shot for a beat generates more flexibility in the editing room; while the director can then choose from among more variations of the complete scene, this approach takes a great deal more time and expense on set.

Virtual Cinematography

Virtual cinematography refers to the cinematic projection of scenes occurring in a 3D graphical environment onto a flat screen, with a virtual camera serving the role of a physical one. There are several inherent differences between virtual and real cinematography. One is that a virtual camera may be instantly teleported from place to place, creating the same "cutting" effect that can only be achieved through editing in conventional cinematography. Thus a virtual camera may essentially shoot and edit simultaneously. Moreover, a virtual camera has the freedom to move anywhere, at any speed, which is often difficult or impossible for a physical camera.

Regardless of differences, in both real and virtual cinematography, the movements of both camera and actors must be closely coordinated in order to achieve satisfying results. In a virtual equivalent of the situation in Figure 2, the tracks would not be needed, but the movement of the avatar and camera would still need to be synchronized in order for the shot to succeed as intended.

The simultaneous shooting and editing and the ability to place the camera at discretion make virtual cinematography well suited for interactive systems where it is not possible to predict the movement of a user's avatar or autonomous agents. However, this versatility comes at the expense of aesthetics, as tight control over both avatar and camera is not always possible.

Related work in virtual cinematography is as follows. Drucker (1994) planned paths for a virtual camera through

visually complex environments such as virtual reality museums. He et al. (1996) use a finite state machine (FSM) of common cinematic patterns called idioms to capture commonly occurring patterns of user behaviors in a 3D graphical chat environment. Christianson et al. (1996) describe how a hierarchical language of idioms can be used to plan cinematic shots. Tomlinson et al. (2000) approach virtual camera control by creating a reactive, autonomous camera "creature" driven by motivations and emotions. Halper and Olivier (2000) use a genetic algorithm to satisfy viewpoint constraints such as occlusion avoidance and relative position of objects in the viewport. Bares and Lester (1999) describe a real-time constraint-based approach to dynamically select the best camera perspective in a dynamic and unpredictable world. Jhala and Young (2004) use hierarchies of camera idioms from which a planner selects sequences of shots to use to cover a script.

The constraint satisfaction, FSM, and reactive approaches are suited for unpredictable camera control scenarios such as interactive systems because shot optimality can degrade gracefully to avoid occlusions. However, if real time interactive performance is not a requirement, a more deliberative process can globally optimize camera placements.

Cambot

Cambot is a thin, stand-alone application that closely models the real filmmaking process to function as a virtual director for offline machinima production. Given a script, it blocks characters, identifies possible shot compositions, and edits the available shots into a final *reel*. A reel contains time-indexed dialogue and gesture commands contained in the script and adds positioning commands for virtual avatars and camera. These commands are rendered by a separate visualization engine.

Input Parameters: Script and Set

There are two types of input that Cambot needs to realize a scene. One is a *set*, a digital environment annotated with labels that describe the types of locations that can be evoked in each constituent space. For example, certain areas might be labeled "indoors," "outdoors," "road," "desert," or "office." In this manner, the set acts as a studio back-lot, which can be re-used from film to film.

The other input is a symbolically encoded script that is analogous to a real-life script, such as the example in Figure 1. Structurally, the script is divided into a number of scenes, each of which consists of at least one beat. Scenes and beats contain the following types of information:

- **Character declarations.** A scene must declare which characters are present, and which of the available avatars Cambot should invoke for each character.
- **Actions.** Actions include lines of dialogue the characters must say and gestures such as nods of the head. Each action is indexed as occurring a certain number of seconds after the beginning of the scene to provide a temporal framework.

Given no other information, Cambot is able to realize a scene from these elements alone. The restrictions that a real script uses to guide the look of the scene, as discussed above, are supported by Cambot in the form of optional constraints. There are four dimensions of constraints that a script may use to guide Cambot's aesthetic treatment of a scene:

- **Location constraints.** Analogous to the scene headings of a real script, location constraints indicate to Cambot how to select an area of the digital back-lot for shooting the scene. For instance, a location constraint might require a scene to appear to take place on a "street" with the declared character of Smith blocked near a point labeled as an "intersection."
- **Blocking constraints.** Each beat may be annotated with constraints on the movements and locations of declared characters relative to one another. From the example in Figure 1, Smith is required to be standing still while Jones is required to move toward him. Blocking constraints are persistent, so a blocking specified in one beat applies to all subsequent beats unless new blocking constraints are provided.
- **View constraints.** The script may recommend (or require) that Cambot cover a certain beat with a certain type of shot, e.g., a close-up of a particular character, a shot that Cambot knows to be "intense," or a shot in which the camera moves.
- **Scene constraints.** With these, the script may guide Cambot's aesthetic choices in assembling complete reels, e.g., to use as few cuts as possible, or to avoid the jarring effect created when the camera crosses the "180 degree line" which runs between two characters on a set.

Cinematic Knowledge

Cambot uses each of the script's constraints to select among all the assets available for realizing the scene. The constraints are matched against a hand-authored library of bits of cinematic knowledge called "facets." Facets fall into the following types:

- **Stages.** A *stage* is an area of space that Cambot assumes to be free from occlusions and obstructions. It functions as the frame on which the other elements of a scene (characters and cameras) are mounted. A stage can be a rectangle, octagon, or other polygon.
- **Blockings.** A *blocking* is a geometric placement of abstract characters relative to the center point of a stage. A blocking must be invoked along with a stage that is sufficiently large to contain each blocked character. A blocking can contain character movements.
- **Shots.** Similarly, a *shot* is defined to be the position, rotation and focal length of a virtual camera relative to the center point of a stage. Like characters in a blocking, a camera can move within a shot.

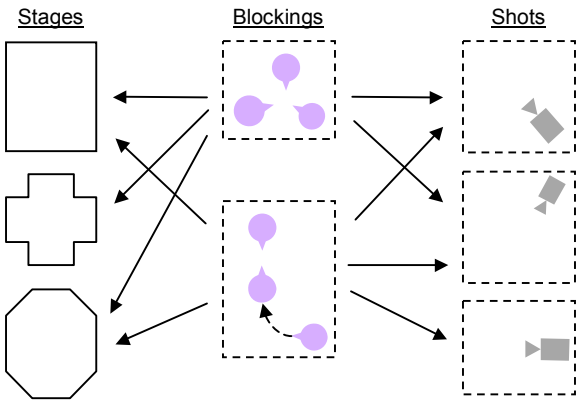

Figure 3. The relationships between facets of Cambot's cinematic knowledge base.

Stages, blockings, and shots are used in conjunction with one another, superimposed by aligning their respective center points. Not all elements are compatible; that is, there is a many-to-many, but incomplete mapping between stages, blockings, and shots that can be combined (see Figure 3).

Cambot uses stages to "package" shots and blockings together in a way that guarantees freedom from occlusions and obstructions. As described in the following section, stages are *anchored* onto the set according to their shape and size in order to instantiate an abstract shot and blocking at a certain location. A stage specifies a bounded region inside of which the set must not contain obstructions or occlusions. For instance, suppose the input scene calls for three characters to be conversing close to the intersection of the two streets, and that the set contains the two intersecting streets. Figure 4 shows how Cambot superimposes stage, blocking, and shot facets for this scene. The stage, represented by the darker rectangle, has been anchored to the set near the intersection of two streets at a position and rotation such that none of its borders intersect the outside edges of the streets. This ensures that no occlusions, such as the corners of buildings, will interfere with the ultimate visualization of the scene.

As Cambot's library of facets grows in size and richness of annotation, so too does the range of available constraints. Some annotations, such as whether a camera is moving or static, are detected automatically; features that are more aesthetic, such as the "intensity" of a shot, rely on manually annotation.

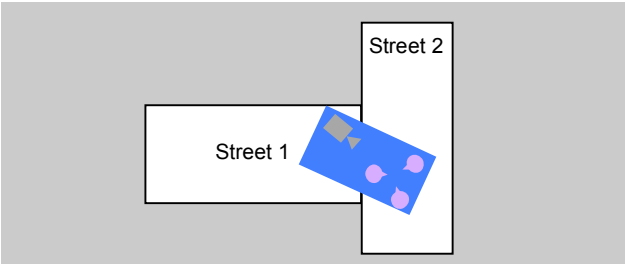

Figure 4. A stage – associated with a blocking and shot – anchored in the set.

Given: A specification for a scene, including participating characters, location annotations, annotated character actions and dialogue.

- For each possible sequence of blockings that satisfies the scene:
  - For each stage that can be applied to the blocking sequence:
    - Find best location on the set
    - Compile all the shots eligible to cover each beat
    - Select the shots to use for each beat to create a reel
    - Take the reel with the highest overall score

Figure 5. Cambot search algorithm.

Cinematic Search

Cambot utilizes its knowledge base to realize an input scene using the algorithm in Figure 5. The first step is to find blockings in the library suitable for the scene. For each beat, Cambot iterates through its blockings and selects those that feature slots for characters resembling the instantiated ones declared in the beat. For instance, there might be two blockings with human adults conversing, one with them facing side by side and one with them facing each other. Cambot binds the script's declared characters to these slots according to the blocking constraints. In the case where more than one binding is possible, Cambot considers each. Given an appropriate blocking and binding for each beat, Cambot assembles a sequence of blockings for the entire scene, called a *scene blocking*, which ensures consistent bindings across all beats. The knowledge base contains information about which blockings can follow one another to avoid consistency problems.

Given a scene blocking as a sequence of blockings mapped to beats, Cambot selects a stage. Each beat blocking in the scene blocking is compatible with one or more stages. Since blockings that can follow one another are associated with compatible stages, there is at least one stage that satisfies the entire scene. If there are several matches, Cambot favors the one that provides the most flexibility for satisfying the remaining scene constraints.

Given a scene blocking and a stage, Cambot attempts to anchor the stage onto the set. This location search maximizes over the location constraints described earlier. For example, if a character is to be shown in a doorway, Cambot places and rotates the stage so that the character falls as close as possible to a point on the set that has been annotated as a doorway. Cambot invokes a brute-force search here, considering all *anchor points* where the stage fits on the set. To keep the geometric search space relatively small, the search space is pruned by constraining locations to certain parts of the set. For example, Cambot will only consider anchoring a street scene on those areas of the set annotated as being streets.

Just as the blocking search considered all beats in a scene, so too does the location search ensure that the location serves all the beats of the scene. The other phases of search, such as choosing a shot, are independent of location selection. That is, Cambot completes its location search prior to searching for shots and reels, rather than conflating these dimensions.

Figure 6. Cambot screenshots: two runs of the same scene using different heuristics.

The next search phase considers each beat independently. Cambot compiles a list of all the shots that are compatible with the previously selected stage and blocking. It evaluates the shots according to the given view constraints. Cambot uses dynamic programming to search for the sequence of shots (i.e., the reel) that best covers the beats in a scene. This technique reduces an exponential search space of shots – given n beats and m shots available, there are m^n possible scenes – to polynomial time, O(mn). This assumes that the property of *optimal substructure* holds for reel selection: that the optimal solution to a large problem contains within it the optimal solutions to the smaller sub-problems on which it is built.

The resulting reel contains the sequence of shots, blockings, gestures and dialogue acts which can be sent to a visualization engine for final rendering. Cambot then repeats the reel-finding algorithm over each satisfactory stage and scene blocking to ensure that it has maximized over these variables. Once the overall best-scoring reel is found, Cambot moves on to the next scene in the script.

The phases of Cambot's search algorithm are serialized rather than nested whenever doing so does not compromise its ability to find an optimal combination of elements. We believe this strategy will allow Cambot to scale effectively.

Cinematic Execution

Once all scenes in a script have reels, Cambot instructs the visualization engine to begin the rendering process via network socket. Cambot instructs the visualizer to place avatars and the camera at particular Cartesian coordinates, to play avatar animations or have avatars speak dialogue, and to move the avatars or the camera along particular trajectories. It associates scheduling information with each instruction to ensure proper synchronization between characters and camera. The visualization engine responds with confirmation when actions are completed. This process continues until the reels for all scenes are rendered.

Currently the visualization engine is Unreal Tournament™ with art assets from the LEADERS project (Gordon et al., 2004). We modified Unreal Tournament to

accept temporally parameterized character and camera positions, a technique based on that of Young and Riedl (2003) in which specialized blocks of control code called "action classes" are triggered through a socket connection.

Figure 6 shows the output of Cambot on a scene involving an American Soldier and an Afghan civilian visiting a U.S. Army base. The script specifies that the Soldier should approach the Afghan and then engage him in conversation. It also requires that the final beat of the scene, containing particularly intense dialogue, be shot in close-up. The script was processed twice by Cambot using distinct heuristics modeling different directorial styles in order to demonstrate aesthetic flexibility.

In the first run, Cambot uses a heuristic emulating a directorial style that prefers to cut as frequently as possible, keep shots as static as possible, view characters' faces when they speak, avoid repeating the same shot twice, and minimize compositional variation. Hence the solution to the scene cuts along every beat boundary and spans many of the shots in Cambot's library available for this blocking.

The second run, by contrast, uses a heuristic that aims to cover the scene with as few shots as possible while still increasing intensity, avoiding repetition, and viewing characters who are speaking. First, Cambot covers the beat where the Soldier approaches the Afghan with a shot in which the virtual camera leads the Soldier down the road – similar to the situation in Figure 2 – until the Afghan emerges from the left edge of the frame. Cambot then covers the bulk of the conversation with a lengthy single shot that moves laterally, parallel to the axis between the two characters, from a dramatically low angle.

Conclusions

We describe Cambot as an intelligent virtual cinematography system that closely parallels the real filmmaking process. We refer to it as lightweight because it relies on a library of modular, human-authored facets of cinematic knowledge to search for an aesthetic cinematic realization of a parameterized script, independent of a graphical visualization. The current state of our implementation includes a library of approximately 50 shots, two stages and half a dozen blockings. This amount of detail enables us to create reasonable-looking movies from short scripts containing several scenes.

By modeling Cambot on the real filmmaking process, we enable it to control a range of creative decisions similar to that available to a human director. Cambot is responsible for choosing a location on a virtual back-lot on which to shoot, blocking the virtual actors, placing the camera, and editing the scene together. As a result, Cambot (a) provides the close control and coordination over virtual actors and camera necessary to reproduce (and extend, where possible) many aesthetic cinematic effects, and (b) eliminates the possibility of occlusions and sub-optimal camera placement. The trade-off is that Cambot is not a real-time camera planning system; its offline search algorithm requires complete scenes as input.

We believe that as a computational model of a real-world movie director, Cambot can be an effective tool for creating machinima. Its cinematic expertise makes it valuable for assisting human-authored machinima or industry pre-visualization. If coupled with a narrative generation system, Cambot can also be part of a larger system capable of automatically generating cinematic narrative.

Acknowledgements

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred. Special thanks to Andrew Gordon.

References

Bares, W.H. and Lester, J.C. 1999. Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds. In *Proc. of the 1999 Int. Conf. on Intelligent User Interfaces.*

Christianson, D., Anderson, S., He, L., Salesin, D., Weld, D., and Cohen, M. 1996. Declarative Camera Control for Automatic Cinematography. In *Proc. of 13th National Conf. of the AAAI.*

Drucker, S.M. and Zeltzer, D. 1994. Intelligent Camera Control in a Virtual Environment. In *Proc. of Graphics Interface '94.*

Gordon, A.S., van Lent, M., van Velsen, M., Carpenter, P., and Jhala, A. 2004. Branching Storylines in Virtual Reality Environments for Leadership Development. In *Proc. of the 16th Innovative Application of Artificial Intelligence Conf.*

Halper, N. and Olivier, P. 2000. CAMPLAN: A Camera Planning Agent. In *Proc. of the AAAI Spring Symposium on Smart Graphics.*

He, L., Cohen, M.F., and Salesin, D. 1996. The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing. In *Proc. of the 23rd Int. Conf. on Computer Graphics and Interactive Techniques.*

Jhala, A.H. and Young, R.M. 2005. A Discourse Planning Approach for Cinematic Camera Control for Narratives in Virtual Environments. In *Proc. of the 20th National Conf. of the American Association for Artificial Intelligence.*

McKee, R. 1997. *Story: Substance, Structure, Style, and the Principles of Screenwriting.* HarperCollins, New York.

Tomlinson, B., Blumberg, B., and Delphine, N. 2000. Expressive Autonomous Cinematography for Interactive Virtual Environments. In *Proc. of the 4th International Conf. on Autonomous Agents.*

Young, R.M. and Riedl, M.O. 2003. Towards an Architecture for Intelligent Control of Narrative in Interactive Virtual Worlds. In *Proc. of IUI 2003.*