Longboard: A Sketch Based Intelligent Storyboarding Tool for Creating Machinima

Arnav Jhala, Curtis Rawls, Samuel Munilla and **R. Michael Young**
Digital Games Research Center
Department of Computer Science, North Carolina State University
Raleigh, NC - 27695
{ahjhala, cgrawls, srmunill}@ncsu.edu, young@csc.ncsu.edu

Abstract

Real-time 3D game environments provide a compelling medium for cinematic storytelling. Professional filmmakers have started using them for pre-visualization. They provide a low-cost learning environment to students. Longboard is an intelligent storyboarding tool that provides content authors access to 3D graphical environments through an intuitive sketch based interface. Users can author scripts and visualize them on a game engine. Longboard provides a pen-based interface for authoring scenarios and sketching 2D storyboard frames that are rendered in 3D on a game engine. A hierarchical causal link planner Darshak encodes cinematic shots and idioms as plan operators and automatically completes unspecified shots from the user generated scenario and executes script and story actions. A real-time geometric constraint solver written on the Unreal game engine is used for rendering the shots. Longboard is developed as a platform for integrating research in interactive storytelling, intelligent cinematography, and intelligent user interfaces for creative professionals. This paper describes the architecture and implementation of Longboard.

Introduction

Virtual environments are developing into a prolific creative medium for cinematic storytelling for professional and amateurs alike. For professionals, they provide a low-cost real-time medium for pre-visualization of movie scenes in full 3D with help from a graphics programmer and a modeling artist. For film students and amateurs, they provide a sandbox environment for exploring the creative space of possible visualizations. Machinima (Lowood 2006) is a recently developed art form in which artists create movies from game engines. The machinima community has developed tools for amateurs to create movies using these game engines. Until recently only programmers with extensive game programming experience had the expertise of working on machinima movies due to the complexity of implementation of camera moves on largely undocumented code bases of existing game engines. In the last few years game engines like the unreal engine have provided users with tools to create movies from these engines without writing complex code. These tools require artists to adapt their workflow to the specific interactions that the tool affords. Most of the camera planning still requires programming knowledge. Controlling camera movements and coordinating them precisely with story actions requires communication between programmers and graphics artists. This introduces an overhead for the director who also has to focus on other aspects of directing, like the script, blocking, communicating cinematic intent, etc. To reduce the overhead of dealing with technology issues and adding an extra communication layer with programmers and artists we have created a storyboarding tool that transparently communicates with a game engine and renders actions in a script and 2D sketches provided by a director through a pen-based interface. Our system has the following features:

- A familiar pen-based interface for storyboarding with support for storyboarding idioms and iconic conventions.
- The automatic mapping of 2D storyboards to 3D visual constraints.
- The automatic execution and coordination of characters and camera on the game engine.
- A transparent communication between user interface and game engine.
- Integration with intelligent shot planning algorithm for automatic completion of unspecified or underspecified storyboards.

The motivation for Longboard comes from the desire to allow an individual user to create a fully realized film solely with a computer interface that doesnt require them to have familiarity with graphics engines or programming. Traditionally, films have been a highly collaborative effort. Production of a finished film requires the cooperation of a director, actors, cameramen, and editors, each of which entails an extensive and specialized skill set. A script must be created, sets must be constructed, actors must act out their scenes in front of a camera, and the finished product must then be editing to create the illusion of a continuous movie. It is possible for individual users to create movies on game engines by importing freely available content without the use of expensive equipment. The use of world editing tools and APIs for game engines for making movies is unintuitive and is hard

to learn. Some commercial software systems are available to help users create 3D visualizations by providing specific interfaces for making movies on game engines.

FrameForge 3D is a computer-based film visualization system (Innoventive-Software 2007). Users model sets and place actors within them. Then virtual cameras are placed within the set to create a storyboard. Users do not need any experience creating sets or using a film camera to make a detailed plan for a film.

Machinima (Lowood 2006) is a technique used for producing movies from game engines. Machinima movies are either created by recording players controlling avatars in the game environment and then adding voice over to the resulting video or by controlling character and camera actions through scripts. If the former technique is used then the recorded movies have to be edited using traditional movie-editing tool such as Adobe Premiere (Adobe-Software 2007). For the latter technique several research systems have been proposed for storyboarding (Jhala, Bares, & Young 2005)(McDermott, Li, & Bares 2002) and developed for automatically directing characters and camera (Elson & Riedl 2007). The constraint-based approaches (Bares 2006) for automated camera placement do not have intuitive interfaces to specify visual constraints and require the users to hand craft each constraint specification file. Automated directing approaches (Jhala 2004)(Elson & Riedl 2007) do not provide flexibility in constraining shots frame-by-frame as their focus is on automated direction with limited or no user intervention.

Most systems currently available for generating movies from game engines either require specific skills like programming or have complex interfaces that are hard to learn and master. Longboard system attempts to alleviate this problem by providing a) an intuitive Tablet PC based interface for specifying content b) a set of tools that help users easily create and modify their visualizations on a 3D game engine c) access to intelligent algorithms for completing underspecified or unspecified shots and assisting with cinematic composition.

Longboard Architecture

Longboard application is designed to run on the Zocalo service oriented architecture (Young *et al.* 2004). Zocalo is a set of tools that provide remote APIs to various artificial intelligence algorithms used for storytelling and execution environments, like game engines, through a web-service interface. Longboard incorporates three major components: a user interface implemented on a Tablet PC, an AI planning engine for creating story and camera plans, and a rendering engine. These components communicate in XML through the service oriented software infrastructure 1 depicts the components of the system and their relationships

Longboard Application on the Tablet PC

Storyboarding is an important and effective means of communication used in filmmaking. Filmmakers use pencil/pen and paper frames for storyboarding scenarios. In recent times, storyboarding artists have used pen based graphics

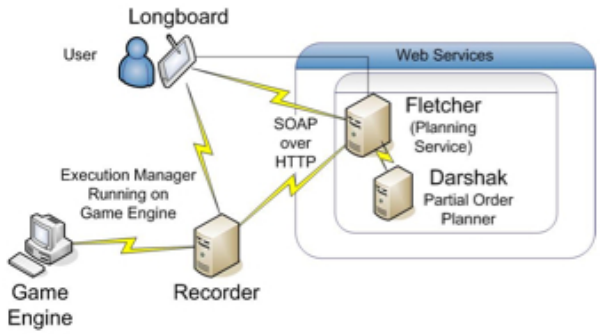

Figure 1: Components of the Longboard System.

tablets for visualizing and effectively communicating their ideas. In the Longboard system, users specify movie directives through a storyboarding interface. The Tablet PC provides a natural interface for drawing storyboard frames. Users have the freedom to sketch characters and objects through the pen using the built-in drawing functionality. This avoids making the user learn manipulation of 3D models in the interface. Pen input provides a shallow learning curve for users, interpreting both stick-figures and professionally drawn characters. The stroke recognition system also allows users to annotate the storyboard frames with text and stage directions in addition to characters and objects in the 3D world.

User Interface. Longboard user interface is designed to represent the typical workflow of a storyboarding session. Users first define a setting and create a script. They then create storyboard frames and associate them with story actions. They can either create fully specified storyboards or can leave storyboard unspecified. Unskilled users especially may not know or care what type of shot or action they desire in a given part of the script. Some users may want to explore different ways of filming the action sequence that they want to visualize. Users can either accept or reject the suggested shots by the planner by specifically adding constraints on certain frames or by adding new frames to the storyboard. The user can then send the completed plan to the Renderer. The Renderer takes advantage of game engine technology to create a video of a rich 3D environment. The Longboard Renderer uses the Unreal Tournament engine to create a virtual world that has a library of sets for setting up the film environment. The Renderers Execution Manager manipulates the actors and camera to follow the action sequence specified by the Longboard interface. An external program records the execution of the scene on the game engine. The recorded scene is then sent back to the user in the form of a compressed video file.

Translating storyboard frames to action representation for the planner and execution environment. One of the key features of Longboard UI is the mapping of 2D storyboard frames to action classes. This mapping is useful in two ways. Mapped action classes serve as input to the

shot planning algorithm and constrain the search space of possible rendering plans, and it can be associated with parameterized procedural objects within the execution environment. As shown in Figure 2, plan operators and execution objects are automatically created from information that is either explicitly specified by the user or is inferred from the sketch. Formalization of operators based on standard film terms (e.g. Close-Shot) enables us to have consistent parameterization of actions. Some of the parameters can be inferred automatically from the script and 2D storyboard sketch. For instance, in Figure 2, the primary actor is inferred from the action associated with the frame, shot distance is inferred from the amount of frame that the bounding box (shown as a dotted line in the picture) of the character occupies, the location of the bounding box in relation to the thirds line determines shot style and the arrow that is placed on the frame sets the tracking property to true which indicates the direction that the character is moving during this shot. The planner, described in detail in a later section, takes as input a story represented as a plan and a partial plan for camera actions. The goals of the planner are to visualize the story actions and events that are not specified in the storyboard and complete the camera plan. In the current implementation of Longbaord, causal and temporal relationships between story actions are not explicitly specified. The camera planner needs this information to reason at a higher level than primitive shots. Longboard camera plans are currently limited to primitive actions and do not take into account discourse level organization of camera shots and higher level scene idioms.

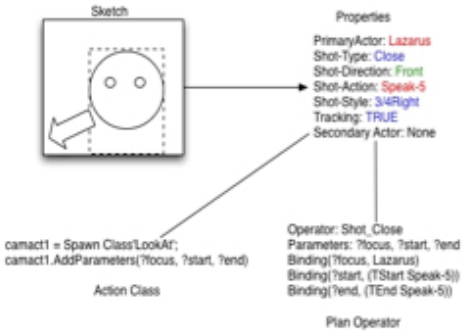

Figure 2: Mapping a storyboard sketch to a plan operator and an execution object

Planning Algorithm

The discourse planner in Longboard is a modified version of Darshak (Jhala & Young 2006) planner. The planner takes domain information comprising characters, objects, locations, initial state of the story-world and story actions, and a set of goals. The planner implements a Decompositional Partial Order Causal Link planning algorithm. Each action in the planner is represented as a parameterized operator containing name, type of operator, parameters, preconditions, constraints, and effects. Action operators are instantiated as steps during the plan construction process. The plan is constructed by adding a step that satisfies one of the open conditions. Whenever a steps effects affect another steps enabling condition the planner detects it as a threat and adds ordering constraints on the execution of steps. The planning algorithm terminates when no open preconditions remain in the plan. We use the plan representation described in (Jhala & Young 2006). The story consists of steps, causal links, and ordering links on the steps. Causal links specify the causal dependence of one step over another through an enabling condition. Ordering links represent the strict order in which certain actions execute in the world. Parameters of the steps specify the actors and objects or locations involved during the execution of these actions in the world. The discourse actions relate to the actions in the story world through temporal constraints on the execution times of the story world actions. Action classes on the game engine described in the previous section are based on the operator library for the planner. This ensures that the planned actions execute correctly as expected on the game engine. Actions defined in the script and storyboards are sent to the shot planner. The goals of the planner are to visualize all story actions through camera actions. Storyboards provided by the user act as partial plans. The planner resolves flaws (Script actions that do not have corresponding communicative shot operators) by constructing a space of plans that have appropriate shots for filming such actions. Actions that are added by the planner along with the shots provided by the user for the script actions. Users can either leave the storyboard unspecified or provide shots for each action in the script. In the former case, the planner fills out the complete shot sequence and in the latter no shot is automatically added.

Recorder Application

Complete script and storyboard plans are converted into an XML action representation based on the standard schema used by the Zocalo architecture. Having a standard schema based representation allows easy integration of different applications within the architecture. The Longboard user application communicates with the game engine via the recorder application. The recorder application is responsible for receiving story and camera actions in XML format, starting up game engine and video recording programs, opening socket communication with game engine, sending the actions with a control message for starting execution when video recorder is ready, compressing the video after execution finishes and saving the time-stamped video file on a web server, sending the web address of the video file to the tablet application for download. This is accomplished by capturing the servers video buffer while the engine executes designated actions. The third-party application, FRAPS (Beepa-Software 2007), runs in the background and outputs a high quality video of the game environment. Due to high resolution, the resulting file is far too large to be sent over an internet connection. Another third-party application, VirtualDub (Lee 2007) is used for compressing the resulting video file before sending it back to the Longboard user interface Figure 3.

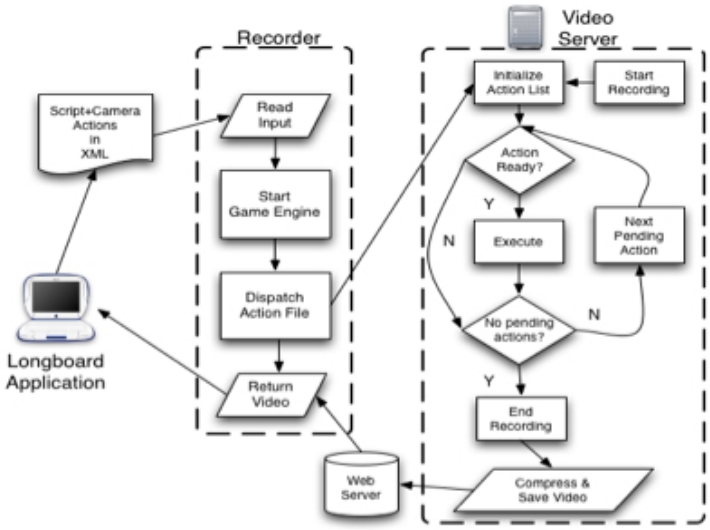

Figure 3: Recorder and Execution Management Flowcharts

Execution Management on the Game Engine

The Execution Manager resides on the game engine and parses the plan file that it receives from the Longboard UI. It instantiates game objects from the information provided in the plan file Figure 3. The actions generated can include story actions, such as character movement, as well as camera actions, such as still shots or pans. These actions are then placed in a pending action queue. Each tick of the 3D engine, the constraints and preconditions of every action in the queue are checked. If all constraints and preconditions are met, the action is removed from the action queue sent to the 3D engine to be executed, and placed in a separate executing action queue. Each action in the queue corresponds to a concrete Action Class within the engine.

Action Classes An action class in the game engine is a procedural implementation of the abstract actions in the Longboard UI Figure 4. For each action in the script available to the Longboard user, a corresponding action class object is written on the game engine. As these classes are parameterized and variable bindings occur at instantiation, they need to be written only once. The action library is extensible and programmers can provide patches for increasing control and adding more expressive cinematic conventions. Story Action Classes control objects in the world, such as characters and world events. Camera Action Classes control in-game camera movement. Each Action Class consists of a set of constraints and preconditions, items that must be true in the story world before the action can begin execution, and effects, which assert the effect of the action's execution on the story world.

```
class Shoot extends Action
 var ZController Agent;
 var Pawn MyTarget;
 function CheckPreconditions(){
   if(Agent.Pawn.Weapon != MyWeapon)
   { return FALSE; }
   if(!(Agent.Pawn.Weapoon.hasAmmo()))
   { return FALSE; }
 }
State Executing {
function Execute() {
Agent.Pawn.SetPhysics(PHYS None);
Agent.Pawn.SetRotation(rotator(MyTarget.Loc
                        -Agent.Pawn.Rotation))
  }
}
```

Figure 4: Action Class in Unreal

3D Game Engine

The 3D engine used by the Renderer is Epics Unreal2 engine. Using the information received from the Execution

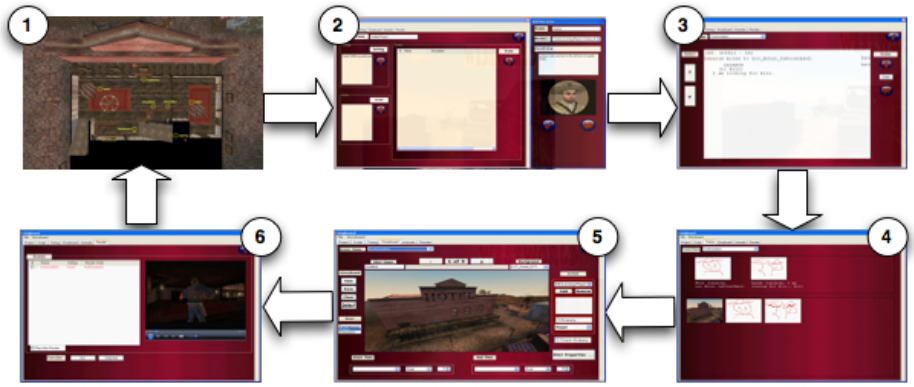

Figure 5: Longboard Workflow

Manager, a pre-made level is loaded into the engine as the setting. For every Actor specified in the plan file a Pawn (class name of the object representing the 3D model and animations of the character) is created and placed in the specified position within the set. The engine receives commands into the execution manager from the recorder in the form of abstract actions. These actions are instantiated as Action Classes (described in the previous section) that can execute in the environment. While the actions execute within the story world, the output of the engine is captured into a video by an external video recording program.

Longboard Workflow

Longboard has been designed with the intention that it can be incorporated into the traditional workflow of storyboarding and movie-making. It consists of a tabbed interface in which each tab represents a step in the process of creating a movie. The user picks a setting and actors, and then constructs a script for actors by adding dialogues and actions for characters. The script editor provides a facility of annotating the script with notes. The storyboard overview lists all script actions and allows the user to create individual storyboard frames and associate them with story actions. An animatic view control provides a timed slideshow of each user drawn storyboard frame and the rendering tab allows the user to select the scenes and send them to the recorder for rendering. The embedded media player control plays the 3D rendering of the scene. Rendered videos are stored on a web-server and are accessible through the web browser for remote reviewing without the Longboard application running. In the following paragraphs we describe in detail a typical step-by-step interaction session for the user with the system.

1. Create a project. The first step is to create a new project, or load an existing one. After naming the project the user selects a setting from a list of available locations. Settings are locations that have been imported into the game engine. In the current implementation a number of settings based on the Western theme are available. Each setting consists of labeled markers that can be used for moving the characters precisely to different parts of the set. A top down image of the labeled set is provided on the Longboard interface and it matches with the location of actor path nodes in the game engine as shown in Figure 5(1).
2. Add scenes. Scene in Longboard is defined as a set of actions and events that happen at the same physical location. Each scene has a location associated with it which is specified by the user in the scene tab. Each scene has associated actors that are part of the scene Figure 5(2). Actors can be added to scenes from a library of available 3D meshes that are accessible on the game engine. All the actors have a standard set of animations that are available. The Unreal engine used in this work provides the facility of importing user generated characters and animations. Art content for the engine is also available from several 3rd party artists and free converter plugins are available for popular 3D modeling toolkits.
3. Add a script. A script is a sequence of actions that takes place within a scene in the story. In Longbard users can construct a script by adding actions from a library of available actions Figure 5(3). Actions are provided in a library because each action has a corresponding execution class in the game engine and a plan operator in the planner. Actions have parameters that can be set from a range of available choices. Currently parameters cannot be left unspecified for any action. Automatic binding of unspecified parameters can potentially be offloaded to the story planner if additional domain information (causal relationships between actions, co-designation constraints, etc.) is gathered from the user. Actions of type dialogue, for instance, have the speaking actor, listening actor and dialogue text as parameters.
4. Add storyboards. A storyboard is a visual guide to how an action should be captured by the camera. The user creates a storyboard by drawing directly onto the storyboard canvas Figure 5(4, 5) with the pen (or mouse). The user can draw background, actors, and stage directions on the storyboard. A set of stock establishing shots for the locations are provided and can be used for setting the 3D context within the storyboarding environment. Actors that are included in the scene are added to each storyboard

and associated with their storyboard representation which is a bounding box around the sequence of strokes drawn by the user. An optional rule-of-thirds grid aides in storyboard composition. The user can also manipulate various properties of the camera including tracking and panning, transitions, zooms, slow-motion. Storyboard sketches can be saved in JPEG format.

5. Review Animatic and Render the scenes. The user selects the scenes that are ready to be rendered. Users can also review a times slideshow of all the hand drawn frames in the animatic control tab. The partial film plan is sent via XML to the story planner. The completed plan is then returned to the user interface. Advanced users may choose to review this raw XML-formatted plan. The scene plans are then sent to the Renderer. The actions are recorded as a video file, one for each scene. The rendered videos are stored on a web-server and a download link is sent back to the user interface Figure 5(6). The user can then play the newly rendered scenes, along with any previously rendered scenes in the embedded media viewer. The user can go back and modify any part of the project and re-render the scenes. Specifically, if the user wants to override the choices of the planner, new actions and storyboards may be added to the unspecified parts of the scene or modifications could be made to the script. When re-rendered, the planner is constrained by the added storyboards and a new planning problem is created with a different solution.

Conclusions and Future Work

Longboard is an intelligent storyboarding tool that provides film students, amateurs, and professionals access to 3D graphics capabilities of a game engine through an intuitive and familiar interface. Longboard connects to a planning algorithm that intelligently generates complete 3D visualizations with unspecified or underspecified 2D storyboards. The camera shots implemented in Longboard and the game engines action library are based on cinematic conventions. Real-time geometric constraint solver ensures that the visualizations are occlusion-free. Longboard attempts to integrate and leverage research in intelligent user interfaces, intelligent cinematography, and intelligent storytelling in a package that is accessible to artists as well as developers.

Longboard is a working tool that incorporates a cinematic shot planner that automatically complete unspecified camera shots. The shot planner is limited by the information about the story that is provided as input. With more information on the rhetorical relationships between actions in the script and additional input for cinematic properties such as tempo, mood, emotion, etc., the cinematic planner can be extended to reason about higher-level phenomena like frame coherence, continuity, and composition. Automatic photographic composition (Bares 2006) is also an ongoing area of research and is being implemented for providing composition assistance and evaluation for hand drawn storyboard frames in the application.

Evaluation of Longboard is being carried out across three dimensions: Usability, Effectiveness, and Expressiveness. Use of Longboard as a medium for evaluating intelligent camera planning and composition systems is also being investigated. We are incorporating richer cinematic idioms in the shot planner library for evaluation of Longboard with a variety of users from experts to novices.

Longboard is a step towards providing artists with a tool that by automatically mapping 2D concept sketches to 3D realizations in real-time - allows them to create 3D visualizations that make it easier for them to communicate their vision more effectively and provide an alternate 3D space of exploring a traditionally 2 dimensional concept space of storyboard frames.

Acknowledgements

The authors would like to thank William Bares for the insightful discussions and constructive comments, and Amanda Macik for implementing the storyboard animatic control. This work was supported by a gift from Microsoft Research.

References

Adobe-Software. 2007. Adobe premiere software suite.

Bares, W. H. 2006. A photographic composition assistant for intelligent virtual 3d camera composition. In *ACM Symposium on Smart Graphics*, 172–183.

Beepa-Software. 2007. Fraps-real time video capture and benchmarking tool.

Elson, D., and Riedl, M. 2007. A lightweight intelligent virtual cinematography system for machinima generation. In *Artificial Intelligence in Interactive Digital Entertainment (AIIDE)*, volume 3. Palo Alto, CA: AAAI.

Innoventive-Software. 2007. Frameforge 3d.

Jhala, A., and Young, R. M. 2006. Representational requirements for a plan based approach to virtual cinematography. In *Artificial Intelligence in Interactive Digital Entertainment (AIIDE)*. Marina Del Rey, CA: AAAI.

Jhala, A.; Bares, W.; and Young, R. M. 2005. Towards an intelligent storyboarding tool for 3d games. In *ACM Advanced in Computer Entertainment 2005*.

Jhala, A. 2004. A cinematic camera planning system for dynamic narratives. *Master's thesis, NCSU*.

Lee, A. 2007. Virtualdub, video capture and processing utility.

Lowood, H. 2006. High-performance play: The making of machinima. *Journal of Media Practice* 7(1):25–42.

McDermott, S.; Li, J.; and Bares, W. H. 2002. Storyboard frame editing for cinematic composition. In *Intelligent User Interfaces*, 206–207.

Young, R. M.; Riedl, M.; Branly, M.; Jhala, A.; Martin, R. J.; and Saretto, C. J. 2004. An architecture for integrating plan-based behavior generation with interactive game environments.