A Machinima-like 3D Animation Production System

Ming-Yuhe Chang, Chih-Ming Chiu, Shao-Shin Hung

Abstract

Aiming to eliminate the shortcomings of existing machinima production platforms, this research implements a new machinima production system by providing more degrees of freedom for camera manipulation, flexibility, and extensibility in production. Our system is composed of three levels: cinematic script editor, presentation, and production system. Through the script editor's interface, one not only can edit flow of script but also can preview and adjust the final result in real-time. Therefore, users can produce matching scripted cinema more efficiently. Furthermore, we provide more flexible camera modes with the capability of integrating user-build level components, characters and so forth. Our system eventually is flexible enough to meet the dynamical demands of a cinematic production environment.

Keyword: Machinima, Game Engine, Animation, FSM, real-time,

1. Introduction

The definition of machinima is, according to [3], that to produce cinema or animation clips by using 3D engine like computer games or similar virtual 3D environment. Owing to this special sort of its creation, the production is called "machine cinema" which contributes to the word "machinima" [4].

Machinima, a new kind of multimedia, originated from the computer gaming video recorded by gamers during game playing. Because of using the resources built in games, it requires lower skill level for users. Thus, one can effortlessly use frame grabber software and video editing tools to create animations. On the contrary, due to the constricted camera modes and special style of game arts provided by the game used, our creativities may be confined.

2. Related Work

We describe several existing machinima related platform and implementation in this section.

2.1 3D Animation and Machinima

The original of machine cinema only presents the humor of games through special camera handling and the recorded images alone without any post-production for sound and video. Later, accompanied by the character's motions, the captions and sounds were added into the video. Unlike traditional computer animations creation procedures, one does not need to be familiar with special animation tools and build the 3D game art, thus it turns into a trend for producing and sharing videos. Nowadays, a few games even provide users with primitive tools to easily create their own videos; by which it also promotes the game itself. Furthermore, to enable uploading and sharing videos, many web sites and platforms are dedicated specially for machinima.

As mentioned before, one must use the built-in resources and manipulate game characters during production of machinima. However, it is not without its problems. The drawbacks can be categorized as follows. First, the number of characters can be simultaneously controlled are restricted by the type of game used. Second, the camera modes provided by the game can be limited. Third, we cannot use the external resources other than that inside the game. Therefore, aiming to solve above problems, we determine to use game engine to implement the machinima production system. By providing basic artificial intelligence for the characters, abundant camera modes and, enabling the importing of game arts into the game engine; we eliminate the limitations of existing machinima production systems and, thus, help improving users' productions for better machine cinema and animations.

The remainder of this paper is organized as follows. Section 2 considers related work on machinima and our references of implementation. In section 3, we present the advantages and disadvantages of existing machinima system and then our system framework and implementations. We describe the user interface, system workflow, storage structure, and present a case study in section 4. Finally, we conclude in section 5 and discuss the future work.
machinima is that, instead of using time-consuming off-line rendering like traditional 3D animation does, machinima utilize real-time 3D virtual environment to create videos [5]. Furthermore, users do not need high-end graphics workstation or special rendering hardware to produce animation clips which can take long time to render. The strength of machinima lie in its real-time recording of game playing video and it's not necessary to wait for the rendering. In addition, due to the popularity of FRAPS (a real-time game play video recording software) [6] and Windows Movie Maker (an audio-video post production software), more and more gamers begin to create their own machinima.

In order to create 3D animations, one usually should learn software packages like 3DS Max or MAYA and it takes long time to familiarize themselves with the sophisticated user interface and features provided by the software. However, it suffices to learn the entry level tools like FRAPS and Windows Movie Maker for machinima. Furthermore, users do not have to construct the levels, characters and so on; instead, they can just use the game resources and post-produce with audio or special effects.

The well-known MACHINIMATION.COM machinima website, in one of its articles, mentioned that, “an art-form comes out of nowhere; films that use computer game technology and look like ‘Toy Story’. Zero-budget film-makers making films that would stretch the biggest of Hollywood studios.” [1] Although it seems to brag about itself, but as 3D game engine being more powerful, the real-time made motion pictures can match in quality with traditional 3D animation clips.

2.2 Application of Camera

In addition to script and art assets, the most essential element is manipulation of the cinematic cameras. Therefore, according to how easily we can control the cameras, it forms the basis of selecting an appropriate machinima platform.

2.2.1 Dynamic Camera Modes

There are so called dynamic cameras other than fixed camera modes, [7] categories several dynamic camera modes as follows.

- **Boom**: Change vertical Z coordinate.
- **Dolly**: Change horizontal X or Y coordinate.
- **Pan**: Change Horizontal rotation Z-axis.
- **Tilt**: Change Horizontal rotation X-axis.
- **Tumble**: Perform unconstrained movement and rotation.
- **Track**: Move camera along tracks.
- **Zoom**: Change the field of view.
- **Pull Force**: Change the depth of field.

2.2.2 Related work in camera mode for machinima

CamBot [8], is a thin, stand-alone application that closely models the real filmmaking process to function as a virtual director for offline machinima production. As a standalone tool, the CamBot takes a script as input, and generates a machinima film, choosing scene locations, character blocking, and camera shots from constraints provided by the author.

This type of technology appeared during the latter period in stages of machinima development. Thus, it is the good criterion to judge whether a platform is capable of future extensions or not.

2.3 Machinima Production Platform

According to [9], existing machinima production platforms can be categorized as four types: Pure Machinima Platforms, Hybrid Games, Pure Games and Modded Games, we summarize each of them in Table 1.

<table>
<thead>
<tr>
<th>Type</th>
<th>The summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure Machinima Platforms</td>
<td>Dedicated tools for machinima.</td>
</tr>
<tr>
<td>Hybrid Games</td>
<td>The games that contain assets and auxiliary shooting tools.</td>
</tr>
<tr>
<td>Pure Games</td>
<td>Plain game platforms.</td>
</tr>
<tr>
<td>Modded Games</td>
<td>The games that modify source codes or original modules from other published games.</td>
</tr>
</tbody>
</table>

Table 1. Existing machinima platforms and their types.

- **Pure Machinima Platforms**: Are created either using a game mod or a custom-built software application that contains no game-related content. These platforms exist only as machinima producing tools. Examples of these include Machinimation, Virtual Stage, and the Matinee in Unreal Development Kit.
- **Hybrid Games**: Are also used as machinima platforms include mechanisms that are built into the interface to make the production of machinima easier. Such games include The Movies, the Sims 2, and Second Life.
- **Pure Games**: Have no explicitly built-in mechanisms for the production of machinima, but can be used for it nonetheless. Games used in this
way include *Halo 2* and *World of Warcraft*, but can also include any other platform in which a producer wishes to make machinima.

- **Modded Games**: Platforms where machinima producers use source code, toolkits, and whatever other applications the platform developers provide to produce their machinima. Commonly produced machinima mods are made using platforms such as *Half-Life 2* and *Neverwinter Nights’* Aurora toolset.

### 2.4 The Structure of Animation Clips

When implementing a machinima system, one has to consider the relationship between the structures of animation clips and system storage. Therefore, we discuss related issues in this subsection.

According to [10], a film consists of many Shots as shown in figure 1. The Shot contains camera-setup, start-time, end-time, target-actors and events. Among them, an event represents the actions that actor performs during some period.

![Figure 1: A film is composed of many shots in which each of them contains various camera types and events.](image)

Therefore, we can treat a Shot as the combination of events during some time interval. For example, if we need a shot for the dialogues among three actors in ten seconds and it consists of three events as follows:

- Actor 1 starts an action in the third second and ends in the eighth second.
- Actor 2 starts an action in the fifth second and ends in the ninth second.
- Actor 3 starts an action in the second second and ends in the third second.

As illustrated in figure 2, through a time-line representation, we can clearly describe the appearing order of the events during a shot. Moreover, it is also beneficial for the user to arrange the events in such an interface.

![Figure 2: Use a time-line to describe a shot and the events in some duration.](image)

### 3. System Overview

In this section, we will first evaluate current machinima production platforms by analysis of their advantages and disadvantages. According to this, we give the main considerations of our research and, then, propose the system architecture and data structures.

#### 3.1 Analysis of Existing Platforms

In this subsection, we evaluate the four machinima platforms mentioned in section 2.3 and give the result of analysis.

- **Machinimation and Virtual Stage**

  The above tools not only provide flexible camera control features including camera zoom-in, zoom-out, panning and so on but also offer saving the camera's position and lighting control. However, users are unable to build their own levels and character models other than select from the default resource database.

  According to [9], the built-in resources are quite limited and hardly meet users' demands except that coincidently match what one needs. In general, those tools cannot provide enough flexibility and extensibility.

- **The Model Viewer of World of Warcraft (WoW)**

  The tool, from the developer of famous *World of Warcraft*, can be utilized to make game related video. The features include abundant art assets, allowing user to pick items, weapons and so on, and plentiful character motions to choose from. However, its workflow needs lots of post-production efforts to synthesize many video clips by blue screen compositing character animations and the exported background images. Although offering rich art assets, it is confined in the WoW framework and does not use 3D environment as the basis in production. Therefore, with limited camera usage, one can only utilize 2D techniques to simulate 3D-like movements.

  In general, this tool is also not highly flexible and hard
to extend, but can be beneficial for the users who intend to shoot WoW related videos.

- **Unreal Development Kit (UDK)**

  The free development kit is from the famous Unreal Game Engine developer. The built-in Matinee tool can be utilized to make cut-scene animation clips and, thus, is useful for creating machinima. Furthermore, Matinee also allows a user to import his own art assets and that make it more flexible for production. However, it is the tool primary for Unreal Tournament game and, thus, one need to be familiar the user interface to create videos.

- **SimCity**

  The advantages to use the SimCity series for making machinima lie in the abundant character assets, levels, and built-in motions. Although unable to import his own models, one can easily modify the textures of the original characters to make new ones. Furthermore, this game also provides primitive furniture for users to build their own levels. The game can output videos, but due to the need for human-controlled camera, the produced video has to be heavily edited to fit the scripts.

- **Pure Games:**

  Due to the similarity of contents in pure games, we separate the discussion here in single-player and multi-player games.

  - **Single-Player Games**

    Current Machinima made by this type of games, most of them are recorded game video supplemented by some post-productions. Because of this type of games is not designed for Machinima and, thus quite limited in usage.

  - **Multi-Player Games**

    This type of games has the advantages that one can look for real human to control the avatars in making machinima. Although the game provides more social motions for characters, the camera manipulation is the main concern because you have to switch to first-person for shooting and, thus, the usage is limited. Furthermore, during the shooting period, one also needs to consider another on-line players or NPCs that, in general, is not very convenient for machinima.

- **Summary of the Drawbacks in Existing Platforms**

  - Do not support art asset importing.
  - Limited art assets resource and usage.
  - Lacking facial emotions presentation.
  - Confined camera manipulation.
  - The number of avatars concurrently controlled is limited.
  - It is not easy to shoot and modify videos.
  - The creativity of script is confined by the platforms.
  - The more videos are shot, the more similar contents produced.

### 3.2 System Implementation

Based on the analysis presented in section 3.1, we can discover that, except UDK, other platforms only provide limited art resources although SimCity can offer finite capability for modification of its models. Eventually, if one wants to shoot a science fiction video then SimCity would be insufficient to fulfill that mission.

Machinima is not without its problems until nowadays. Due to various constraints from the games used, the film maker’s originality is greatly confined. Furthermore, as the number of film-makers using the same game getting larger, the similarity among them is easily to be discovered. For instance, the motions of anger in films tend to be identical and that hardly reflects the author’s creativity.

In order to remedy the above situations, we need a platform in which users can construct their own art assets for our system reference implementation. Unreal Development Kit, so far, seems to fit our demands. However, it still has several limitations.

- **The Disadvantages of UDK**

  Based on the contents of chapter 13 in [1], we can summarize the flaws of UDK as follows. While Unreal Development Kit does has built-in tool called Matinee and that is originally made for producing film-style scenes. However, other than the drawbacks been stated above, it basically has to rely on user's controlling avatar to enter trigger zone, and then activates the video production process; thus, the workflow seems not very convenient.

  Currently UDK does not offer its source code and, thus, it would be nearly impossible for us to extend it as a full machinima platform. Let alone if we need to develop some adjunct systems in the future like the CamBot mentioned in section 2.2.2.

  Take into consideration all of the above, we eventually choose the game engine, Torque 3D, with source code as the reference implementation platform.

- **Torque 3D Game Engine**
Torque 3D [11] provides not only the source code but also many powerful editors. For example, the graphical user interface editor to build the tool interface in this research; terrain editor and river editor can help rapidly constructing 3D levels. The built-in capability for massively multiplayer networking and high performance real-time computing also fit our needs.

3.3 System Architecture

Similar to the ideas presented in section 2.4, our system framework bases on the concept of timeline on which events occur in a shot are arranged. The difference is that we divide the events into many categories instead of just describing character's motions.

As illustrated in figure 3, we use a shot, the minimum unit in cinema, as the scene-based system data structure. One can use graphical user interface to edit the six attributes of a shot including caption, camera, character, action, special effects and audio.

Taking the cross-platform into account, in addition to Torque script, we also provide XML script output for other game engines to import. As for the video output, one can use the Fraps software to capture during scripted basic units been played accordingly.

3.4 Structured System Attributes

In this subsection, we describe the six attributes given above in detail and also list the types they hold (Figure 4).

- **Caption**: The words appear under the screen during a shot, it can be dialogues between two characters or just a aside which represent a non-dialogue plot.
- **Camera**: Can be categorized as three types including default, fixed and dynamic camera.
  - **Default camera**: Its relative position is calculated based on the "Eye" point, which is added during creation of the model, in a character. We have total twelve types of default cameras including Eye, Face, Shoulder, Half Body, Thigh, Full Body, Over-the-shoulder Shot, Point of View, Low Angle, High Angle, Eye Level, and Bird's Eye View.
  - **Fixed camera**: If none of the twelve types of default cameras meets the demand of users, our system offer the option for one to specify his own cameras in fixed positions on the inside of 3D environment.
  - **Path camera**: Represents a dynamic camera that can move along an existing path, one can manipulate it on demand.
- **Actor**: The character in a cinema and it needs to be imported into game engine before start shooting. One can use a "Dummy Actor" to replace a real actor before its importing, and edit the movement and cameras beforehand. Eventually, the formal actors are placed back again.
- **Action**: The behavior of a character, including six types: Move, Aim, Facial Motion, Lip Motion, Body Motion and Mount.
  - **Move**: To control movement of the character.
  - **Aim**: To control a character to face another or a fixed direction.
  - **Facial Motion**: To control the happiness, anger, sadness, etc on a character's face.
  - **Lip Motion**: To control a character's lip animation during speaking.
  - **Body Motion**: To control a character's body animation.
  - **Mount**: To equip a character with an item or Un-equip an item.
- **Effect**: Special effects, including the Weather, Day and Night Variation, and Time Scale of
Aiming at the audio part, the music and voice usually are recorded afterward according to the atmosphere of the video. While still a few people prefer to finish the audio before start shooting, our system provide flexible options for this kind of operation which depends on the habit or requirement of a user.

4. System Implementation

In this section, we will describe the implementation in detail. As stated before, we use the Torque 3D source engine as our reference implementation platform.

4.1 The Production Workflow

As illustrated in figure 5, during the Pre-Production stage, the major work is to import art assets including terrain, character models, buildings etc. Once finished, one can start editing the scripts for the environment -- otherwise we suggest that, at least, terrain should be imported. This is because, inside game engine, all positions are in absolute coordinate. Therefore, if one does not use the actual terrain for shooting to arrange camera positions or related paths, then it is highly possible that, after terrain is replaced, all previous settings will need to be rebuilt, and is a waste of time!

Figure 5: Machinima production workflow

Inside the engine, one can edit the attributes of basic units by the user interface, then real-time preview the result and do fine tuning till the quality is satisfied. Once all units are ready, all we need to do is playing and recording using the Fraps software. One trick worth mention here is that it is not necessary to edit all shots at a time to make a video. Instead, one can record the video in pieces and then link them afterward. Ultimately, it is up to you to do post-production or re-editing.

4.2 Attribute Structure

We have presented the concept of attribute structure in section 3.4. This subsection describes the real data structure used to save them.

Figure 6: The data structure for attributes

We save values of the attributes in a ScriptObject, as shown in figure 6. The system will read in the values and performs corresponding actions. The main attributes are described in detail as follows:

- **Duration**: The time that a shot takes in seconds.
- **MoveDestinationList**: The list of all character movements includes five values: Trigger time, character ID, path name, start node, and end node.
- **SequenceList**: The list of body motions of all characters in a scenario. It consists of three values: Trigger time, character ID, and motion name.
- **AudioList**: The list of playing audios. It consists of two values: Trigger time and file name.
- **MountList**: The list of items mounted. It consists of four values: Trigger time, character ID, item name, and mount point.
- **UseCamera**: The types of camera modes include three values: Defaultcamera, CameraPos, and PathCamera.
- **CameraType**: Twelve types mentioned in section 2.2.
- **PathCamera**: The path name when using the path camera.

Please note that the SequenceList can contain...
many characters information by which the behaviors can be played in the same time.

4.3 System Functions

The system which is illustrated in figure 7 can be divided into two phases including the preprocessing stage and script event editing stage. During the preprocessing stage, we primarily arrange the necessary assets including character animations and previews, construction of the paths and nodes, and create the characters for actual shooting the scripts. In the next stage, we aim to arrange scenarios according to the script events. The functions include event editing, separate preview, script merge, event channel hiding, and single event hiding.

4.3.1 The Preprocessing Stage

- **Animation importing**: In order to speed up the importing workflow for character animations and increase the overall efficiency, we construct a user interface to aid user to select necessary action files and export them for characters.
- **Path creation**: The paths mainly used in two situations where a character or a dynamic camera needs a path to follow. The primary way of constructing the necessary paths is to create nodes in the 3D environment and two of them can form a path. Notably, we need to take the rotations of nodes into consideration because they also can affect the orientations of dynamic cameras.
- **Node creation**: Mainly used to build the positions for creating static cameras and characters. Therefore, the nodes can be categorized as the static camera nodes and the character creation nodes which record the starting positions for characters.
- **Character creation**: Create actual characters in the scene and save the necessary information of each by their creation order, e.g. actor1, actor2, etc. The benefit of this method lies in that if it were necessary to change an actor's model then all we have to do is changing the saved data of the character. Thus, we do not need to re-edit the scripts.

4.3.2 The Script Event Editing Stage

- **Event editing**: Use the time line user interface to arrange the events which are triggered in occurrence time order; therefore, the script flow and then the cinema can be produced.
- **Separate preview**: One can divide the script into several scenarios by selected time positions and preview each scenario. This function can help users to adjust minor scenarios.
- **Script merging**: Help a user to swiftly pick the scenarios been built before and rearrange them to form new scripts instead of creating from the beginning every time.
- **Event channel hiding**: If necessary, one can hide the events of the same series; e.g., if we choose to hide the body action channel of a character then the according animations will not be played during the preview.
- **Single event hiding**: In order to let users focus on previewing or correcting a specific event, one can hide the selected event; thus, it is not played.

4.4 Descriptions of the User Interfaces

In this section, we describe the main user interfaces used in the system.

As illustrated in figure 8, the user interface primarily divided into two windows. On the left, interface (a), it is a window for real-time previewing the 3D scene. On the right, window (b) consists of a time-line interface and auxiliary tool and character preview.

Figure 7: System functions.

Figure 8: (a) Real-time preview window. (b) The editing tool.
sub-windows. Each interface can be arranged according to a user's preference. The details of each interface will be described in the following subsections.

4.4.1 Real-time Preview Window

Figure 9: Real-time preview window.

In real-time preview window, by controlling the flying camera, one can create camera and character creation nodes as illustrated in figure 9. After the nodes been built, we can see their relative positions in 3D scene and then adjust the positions by related tool interface if necessary. Furthermore, we can preview the scenarios presented by each camera node. In addition, the built paths and related changes of event attributes can be directly real-time previewed. Eventually, in real production of the cinema, all we need to do is hiding those nodes, followed by playing the scripts and, in the meantime, recording the videos.

4.4.2 Character Preview Window

In this window, illustrated in figure 10, we can preview the character and its related animations. Furthermore, by this interface, one can add animations and modify their attributes; e.g., to check the cycle attribute's option.

Figure 10: Character preview window.

4.4.3 Auxiliary Tool Interface

Aiming to assist users to make machinima, the auxiliary tool interface which is shown in figure 11 currently composed of six functions include creation of the paths, creation of the nodes, event editing, script merging, creation of the character and animation importing. We will describe three of them in the following subsections.

4.4.3.1 Path Construction Function

To create a path, one can input a path name, followed by adding a new path using the path add button. You can select an already built path from the drop down menu and then add path nodes to it in real-time preview window. The node’s delay attribute denotes the duration, in seconds, when camera pause in this node; thus, improve the efficiency of camera manipulation. After all path nodes been added, you can preview the scenario in different camera speeds. The interface is shown in figure 12.

Figure 12: Path construction interface.

4.4.3.2 Node Construction Function

The procedure for creating a node is as follows. The user inputs a node’s name and select whether to create a camera node or a character node. Next, after we select an existing node from the drop down list, the dynamic camera can be transported to the node’s position for previewing, and therefore helps swiftly adjusting. Moreover, one can also select a character and transport it to a node’s position for helping the user to set a character’s position. The interface is shown in figure 13.
4.4.3 Event Editing Function

The event editing interface, illustrated in figure 14, is used to edit the events that are triggered in various scripts. It is mainly by means of selecting the event units in time-line interface to dynamically modify them. Moreover, the interface which is shown in figure 15 can change according to the event type that we select.

4.4.4 Time-line Interface

In the main editing interface, which is shown in figure 16, we provide a graphical time-line interface that is based on the Verve Tool Kit [12]. It helps the user to quickly arrange related event list by the event’s type in a hierarchical manner. Moreover, the vertical bars shown in each level represent trigger points which can be selected by mouse to edit their attributes in the auxiliary tool interface. Once the editing is finished, one can push the play button to preview the scripted motions of the characters and cameras. The above adjusting steps may be repeated several times until the scripts can be saved as a file that is useful for the following editing.

4.5 Case Study and Discuss

In this section, we present an experimental evaluation in the form of case study. The title of the film is “The Rates and Their Daughter” which is adapted from a Japanese folktale about the mouse parent want their daughter to marry the most powerful being in the world. Our system is performed on a PC running Windows 7 64bit with AMD Athlon II X3 450 processor, 4GB memory and an ATI Radeon HD 5770 graphics adaptor. The implementation is based on the Torque 3D game engine with the Verve add-on tool and the Yack Pack character dialog system [13].

As shown in Figure 17, we can see one of the processes with fixed camera mode during editing. The pink nodes denote character creation nodes which represent the initial positions for the actors. In addition, the red crosses denote the nodes used to construct a path.

The working hour statistics related to use our system can be divided as three parts: the planning phase, the design phase, and the shooting phase. In the planning phase, the tasks include component planning, dynamic scripts creation, etc. The tasks of the design phase include 3D models creation, texturing, etc. The tasks for the shot phase include art assets importing, event editing, various path creation, etc. As shown in Table 2, we can easily notice that, compared with the design phase, the working hours spent in the shooting phase are relatively much fewer. The pictures of the film are illustrated in figure 18.
Table 2: Working hour statistics of the “The Rates and Their Daughter”.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Working Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>63</td>
</tr>
<tr>
<td>Design</td>
<td>913</td>
</tr>
<tr>
<td>Shooting</td>
<td>213</td>
</tr>
<tr>
<td>Total</td>
<td>1189</td>
</tr>
</tbody>
</table>

Finally, by comparing the above working hours with the estimation of that of the rest films being shot, the relation between resource cost and the rate of reusability can be modeled as figure 19. It indicates that, once the assets been finished in the design phase, the cost can be effectively lowered by promoting the reusability.

Figure 19: The overall resource cost versus reusability.

5. Conclusion and Future Work

Compared with the traditional 3D animation, our system can reduce tremendous amount of time and cost because that, by using game engine, we do not need to wait for the long time of rendering and the quality requirement for the art assets can be lowered. Furthermore, the known disadvantages of the past machinima systems such as the limited in creativity can be alleviated by using our system.

Based on the awarding winning Torque Game Engine, our machinima production system allow the user to make cinemas that really meet the need instead of using the limited resources in games to create videos that are all similar in style.

The concrete contributions of our system are as follows.

- Use game engine technology to support art assets importing which is a common problem for the past systems.
- Make good use of the user-build art resources to overcome the flaws in shortage of the game’s art and, thus, that can improve the originality.
- Allow the user to create various camera paths to overcome the flaws in the camera manipulation.
- Use the arrangement of scripts to control multiple characters’ animations and behaviors that overcomes the problem in concurrently controlling many actors.
- Record related information during the production process to solve the problem of inconveniency in editing and shooting.

In addition to making cinemas such as commercials and shorts; our system can, in future, also used to make educational multimedia. Moreover, we plan to apply our results in the courses of integrating technology and arts. An interesting future research topic will be a fusion of existing devices and machinima.

Acknowledgements

This work was supported by the Mobile Content Business Model Research Project in 2009, the Department of Industrial Technology (DoIT) of the Ministry of Economic Affairs (MOEA) (Project Number 98-EC-17-A-29-S2-0113).

References