CAMERA CONTROL FOR SHOT SELECTION IN MACHINIMA GENERATED ANIMATION

[1,2]DELTA ARDY PRIMA, [3]TSUYOSHI USAGAWA, [1]I KETUT EDDY PURNAMA, [1]MOCHAMAD HARIADI,

[1]Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Indonesia

[2]Department of Informatics Engineering, Universitas Surabaya, Indonesia

[3]Graduate School of Science and Technology, Kumamoto University, Japan

E-mail: [1,2]delta@staff.ubaya.ac.id, [3]tuie@cs.kumamoto.ac.jp, [1]ketut@te.its.ac.id, [1]mochar@te.its.ac.id

ABSTRACT

The development of computer graphics–rendering technology is rapidly increasing—the next-gen game can present 3D images that are closer to reality and can be run in real time. This allows the emergence of a new technique called “machinima,” derived from the words machine and cinema, to make real-time animation movies in a virtual environment. Machinima filmmaking requires a cinematography rule to produce a good animation. A behavior tree is used to select the camera shot according to the cinematography rule using shot idiom such as close-up, back-shot, shoulder-shot, etc. The system allows the camera to decide the proper shot based on events and actions happening in the virtual scene.

Keywords: *Machinima, Cinematography, Behavior tree, Virtual camera, Game, Virtual world*

1. INTRODUCTION

Although current technology has reached a point where the level of realism in games is extremely high, the history of filmmaking has shown the importance of cinematography to better engage with the audience. Audiences have become accustomed to certain conventions in film, and they expect these conventions in any form of visual entertainment.

In a recent form of digital art, machinima, virtual cinematography is more important than before. Machinima refers to the innovation of leveraging video game technology and using pre-rendered 3D images to greatly ease the creation of computer animation [1]. Game engine for modeling human behavior research has been increasing as in [2], also filmmaking using machinima is a complex visual medium that combines virtual cinematography, avatars control, and editing to convey a story. The camera must film the right event at the right time to produce the right picture [1]. Cinematography and editing are both complex art forms in their own right.

However, manually placing the virtual camera can require a great deal of modeling and is a time-consuming effort that must be repeated for each scene [3]. Cinematography depends on storytelling over time, whereas in an interactive virtual world, the story is developed almost immediately. Because of this immediate reaction in accordance to scene elements, common phrases in cinematography, including framing, composition, motion, etc., can become too complex to be handled by automatic camera control in a virtual environment.

In composing the visual properties of a shot, a cinematographer may vary the size of a subject in the frame or the relative angle or height between the camera and subject. Shot sizes include extreme close-up, close-up, medium, long, and extreme long in which a subject’s size in the frame appears progressively smaller or more distant. A filmmaker can also use editing decisions such as shot duration and the frequency of cuts to artful effect [4].

Similar research has been attempted previously, although for different domains and goals. As in [5] present a successful attempt to give the game designer a more robust way to customize the portrayal game and give the user the ability to create complex camera transitions to provide a more engaging game experience. Work by [6] describe some of the established cinematographic techniques. Other works [7] implement a method for various camera movements (pans, tilts, rolls, and dollies).

Based on description above we need to study how to automatically place the virtual camera and implemented the cinematographic principles for shot selection in machinima generated animation in real time scenario and multiple scene.

Therefore, this paper explores the capabilities of a behavior tree to interpret one or more camera shot phrases in cinematography. This research makes possible different triggers (events, actions, or inputs) to achieve the proper shot and determine camera properties in virtual environments that built using game engine. Our system automatically shoots the character, action, or events occurring in real time. The system tracks a single actor, multiple actors, a dialogue scene, and a fighting scene to demonstrate real-time machinima movie making.

The paper is organized as follows. Section 2 discusses previous related work. Section 3 describes an overview of our approach. Section 4 presents the experimental results. Finally, in Section 5 we conclude the paper and perspective for future work.

2. RELATED WORKS

Similar research has been attempted previously to dealt with automatic camera placement and implemented the cinematographic principles in many different domains and goals.

Cinematography has many varieties of rules and principles to communicate the meaning of the film from the directors to the viewers.

Markowitz *et al*. [5] provide the system with smart events that store the behavioral response and parameter information inside the event. The system then automatically updated the camera placement using information given by the smart event.

Work by [6] provide the camera planning system with declarative camera control that allows the system to formalize, encode and implement generic shot idiom – a stereotypical way to capture the scene as a series or sequence of shot like medium shot, close-up, and over the shoulder shot. The input needed by the system are the character position and action and also which actor to be captured by the system.

As in [7], they developed the CINEMA system that consists of two major part, the database that contain objects information and the parser part that interpret user commands. The system used for teaching the student how to present simple animation and plan the real camera shot.

Other research [8] build video sequence for documentaries from images that using only wheeled type camera movements. Work by [9] uses automated reasoning using a cinematic knowledge to determine camera behavior.

3. METHODOLOGY

In this paper, each input is derived from changes in a scene. These changes provide information to the data level, which is then processed into coordinates that are used for camera placement.

3.1 Design System

The camera system has the ability to recognize changes to an event. These changes can be marked by a change in position, an actor's circumstances, and information regarding the position of the camera. In order to determine what shot to take, the camera system must have an autonomous decision maker; in this case, the system will adapt a behavior tree into a series of sequential shots as shown in Figure 1.

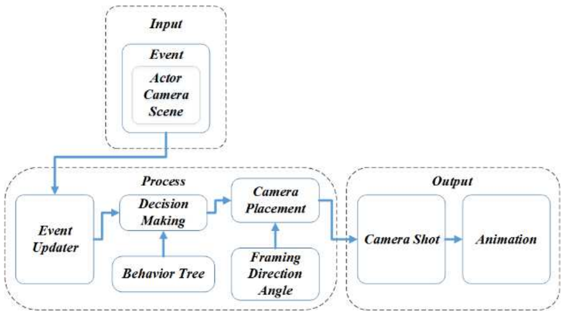


Figure 1: Camera placement system design with behavior tree

After receiving the input, but before the camera moves, the system will provide information regarding which direction the camera should rotate and how the camera will capture the viewpoint, including framing, direction, and angle, based on the results given by the behavior tree

3.2 Behavior Tree for Camera

Instead using decision tree that commonly used in the game to be incorporated and promote better reasoning [10], for metafleet movement, attack time and determine number of battleship in DEFCON game [11], we proposed to use behavior tree. Behavior tree allows for more complex behavior to be performed like to capture the shot and implement cinematography rule. The chart of behavior trees using the logic "AND / OR" in action, it's have a form of a statement node and various commands to implement a variety of complex actions [1]. A simple form of a behavior tree is shown in Figure 2.a, the blue chart on the left is the event updater. The event updater will check the various changes that occur in sequential scenes. The right side shows a wide variety of shots performed sequentially from left to right. Figure 2.b, shows the detailed behavior tree sequence for dialog events.

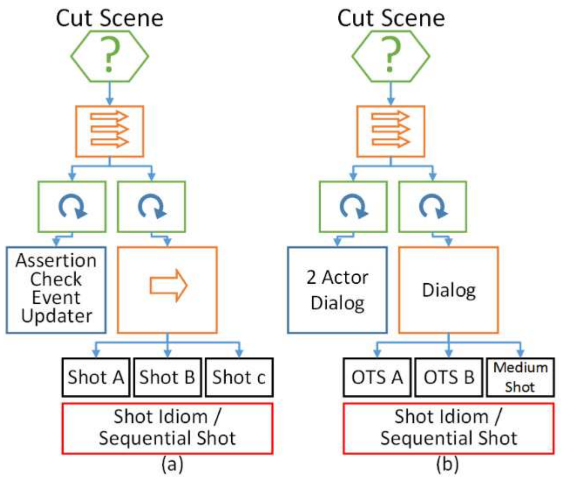

Cut Scene
?
Assertion Check Event Updater
Shot A
Shot B
Shot c
Shot Idiom / Sequential Shot
(a)
Cut Scene
?
2 Actor Dialog
Dialog
OTS A
OTS B
Medium Shot
Shot Idiom / Sequential Shot
(b)

Figure 2: (a) Simple behavior tree, (b) detailed behavior tree

An Event Module as shown in Figure 1 stores behavioral responses and parameter information so that any changes to the sequence or context of a scene do not interfere with the camera's ability to properly record it. The camera must be able to update how it shoots and records while moving. To achieve this, the Event Module must have a wide range of preprogrammed options arranged in sequential order. For more details on this process, see Figure 3.

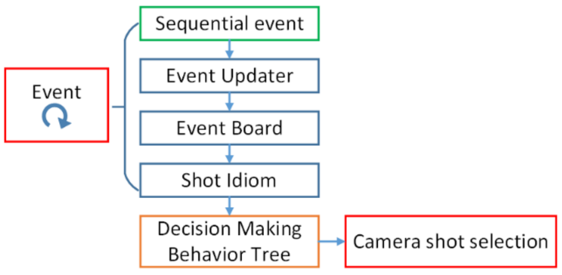


Figure 3: Event module in behavior tree system

The first part of the Event Module provides updates about the event, which it then transfers to the Event board, allowing it to make a shooting decision, thus determining the shot's idiom.

If a change occurs, such as the number of actors in a scene or the background setting, the event updater will transfer the updated event information to the Event board.

After the Event Updater sends information to the Event Board, the latter will decide the idiom of the shot. The Event Board is the core of the Event Module. The Event Board will determine which camera will be active for the next cut scene. If using more than one camera, the Event Updater will send input to the Event Updater about which camera was previously active and will assign cameras tasks on a priority basis contingent upon the specific shot idiom.

The resulting information will be converted into a series of sequential shots which will in turn be implemented into a behavior tree decision making process. The Event Module will continuously update the behavioral tree regarding any changes to the sequence of events.

3.3 Cinematography and Shot Idiom

Idiom shot is a conversion of a wide variety of data types resulting from cinematographic representation in a virtual world with a camera cinematography process that can be translated into a narrative form.

A film composed of a wide variety of scenes that sometime demands a change of shot in each sequence. So every idiom shot contains a sequence set up camera. A shot idiom is encoded by defining the necessary parameters within the scene, that is, the actors, time, and angle of the camera. With the idiom shot, the camera is expected to perform a series of different shots in a sequential order.

Different camera shots can create different storytelling styles, and even different interpretations of the story. Camera shots must be planned in a coherent style known as a sequence. In each shot, the camera can be placed in a steady, consistent point with a certain angle, or it can move in a particular path or be rotated around a certain axis following the motion of the shot. Although the options are theoretically infinite, cinematography consists of certain heuristic principles to determine camera placement, angle, and motion based on the context of the shot [12].

Where the camera is placed depends on the space occupied by the actor (or actors) in the framing shot. Framing and composition are essential to communicate the situation. The depth of a particular frame may provide different information for viewers. The framing shot also has a composition—the various arrangement of elements in the shot that creates balance [13].

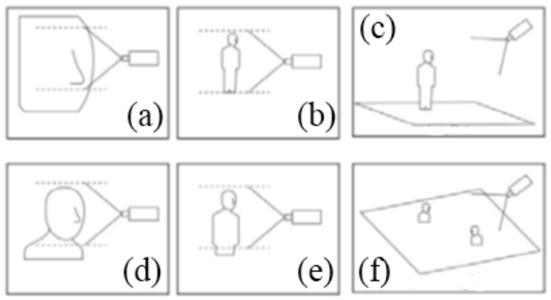


Figure 4: Shot idioms in cinematography principles, (a) extreme close-up, (b) full shot, (c) long shot, (d) close-up, (e) medium shot, and (f) extreme long shot.

Figure 4 shows different shot idioms according to cinematography principles.

- Extreme Close-up: Generally, refers to a shot showing only the eyes and mouth, but can also indicate an even tighter shot showing only one feature that is important to the current scene.
- Close-up: This shot shows the head down to the top of the throat.
- Medium Close-up: This shot shows the head and shoulder.

The angle at which the camera views the scene can also have an important impact on the scene's mood and meaning.

- High Angle: A shot that looks down on the scene. This shot tends to give the audience a feeling of dominance or omnipotence and reduces the importance of elements in the scene.
- Low Angle: Opposite of high angle, this is looking up at the scene. This shot tends to make the audience feel more diminutive and lends a sense of power and greater importance to the elements of the scene.
- Dutch Tilt: A shot with the camera rotating around the view axis. Although not commonly used, this shot can emphasize story elements such as disorientation, hallucination, or confusion.

Basic camera moves include three actions: pans, tilts, and tracks. In general, the most complex camera moves are simply combinations of these basic components. These combinations provide six degrees of freedom, allowing the camera to be positioned correctly, as shown in Figure 5.

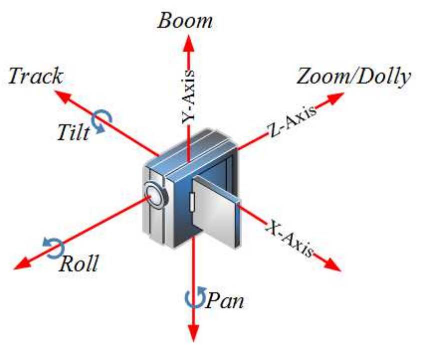


Figure 5: Basic camera movement.

The camera system used in this study has several properties that will be used to determine the type of shot used based on the parameters and inputs provided by the events that occurred in the virtual world. Some of the properties are:

- position (x, y, z): The camera position in the 3D virtual world with a vector xyz, this coordinate point is used to record the position of the actor.
- Rotation (Rx, Ry, Rz): For camera movement involving panning and tilting.
- Target: Target actor that the camera is "following."
- POV: Ideal point of view for the camera.
- FOV: Field of view for the camera.
- Aspect Ratio: Default aspect ratio for the camera.
- Distance: Distance from actor to the camera and the distance of each actor from one another. This allows for grouping actors at a distance.
- Height: Makes it possible to capture shots of a certain height.
- Event ID: Data about specific events that allows the camera to move based on a specific event. Event ID will be described with a cinematic representation item.

The properties of the camera will be used to perform a wide variety of responses to a running scene.

3.4 Shot Direction and Shot Angle

Shot direction is the difference between the camera rotation angle and the actor rotation angle from the Y axis. A direction of 0º means that the camera sees the actor from the front; conversely, a direction of 180º means the camera looks at the actors from behind (Hawkins, 2005). In this research, the direction of the shot is converted into a limited form (discrete) with a 15º angle difference (see Figure 6.a).

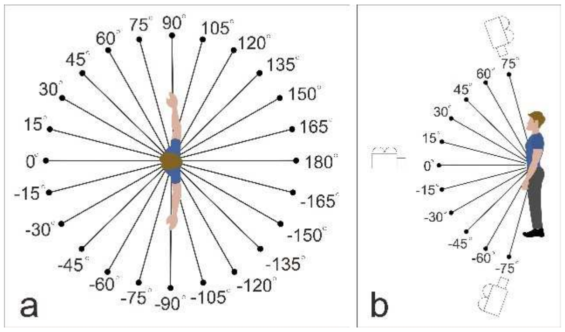


Figure 6: Shot direction (a) and shot angle (b).

The shot angle is the difference between the camera rotation angle and the actor rotation angle from the X axis. Just like the shot direction, the shot angle is limited by differences in 15º angles (see Figure 6.b)

3.5 Spherical Coordinate to Euler Conversion

In the process of framing the direction and shot angle, the camera moves to a certain position relative to the actor. To translate a location in the virtual world to Cartesian coordinates, we need to convert the spherical coordinates to a Euler coordinate system.

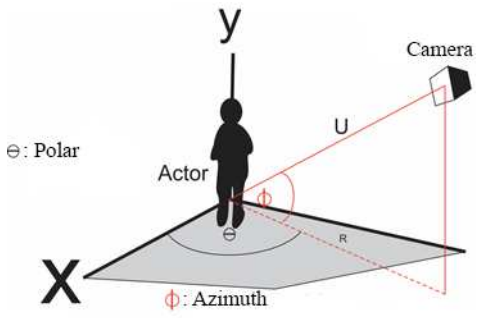


Figure 7: Camera and actor position in spherical coordinate.

In Figure 7, R is the radial distance to the center point of the world. In this case, the center point of the world in the actor. U is the relative distance of the camera to the center of the world. Then ϕ is the angle formed on the xz plane, known as the altitude of the camera position. θ is the azimuth position of the camera with a viewing angle on the xy plane. ϕ is in the interval $-\frac{\pi}{2} \leq \emptyset < \frac{\pi}{2}$ and θ is in the interval $0 \leq \theta \leq 2\pi$.

Using trigonometry, we can understand the relationship between the spherical coordinates and Cartesian coordinates (u_x, u_u, u_z) to the position of the camera in the virtual world using the formula:

$$u_x = R\cos\emptyset\cos\theta$$
$$u_y = R\sin\emptyset$$
$$u_z = R\cos\emptyset\sin\theta \qquad (1)$$

The relationship can also be reversed with the formula:

$$R = \sqrt{u_x^2 + u_y^2 + u_z^2}$$
$$\emptyset = \sin^{-1}\left(\frac{u_y}{R}\right), \theta = \tan^{-1}(u_x, u_y) \qquad (2)$$

3.6 Framing

Framing is closely related to R in terms of the spherical coordinates that was explained in the previous section, wherein R is the relative distance from the actor to the machinima camera.

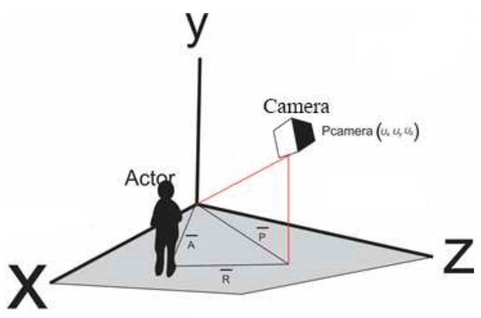


Figure 8: Camera distance with actor in virtual world.

In Figure 8, A is a distance relative to the actor from the center point of the world, P is a distance relative to the camera toward the center point of the world, and R is the distance from the actor to the camera in the virtual world; then R (vector) can be expressed by:

$$\bar{R} = \bar{A} + \bar{P} \qquad (3)$$

Framing is a process that determines the height of the camera when taking a shot. Thus, the camera angle is important when creating the shot associated with actor’s height (see Figure 9).

Journal of Theoretical and Applied Information Technology
15th November 2016. Vol.93. No.1
© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

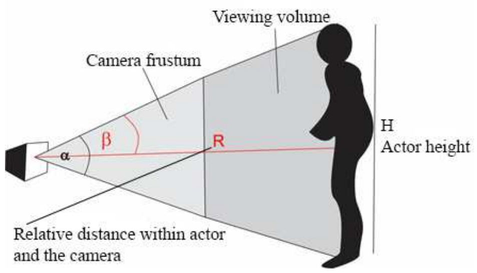


Figure 9: Relationship between camera distance and actor height.

Where β is the angle obtained from the maximum height of actor with R. Then the relationship between the height of the actor (H) and the angle α is:

$$R = \frac{h}{\tan\beta} \tag{4}$$

where h is given by:

$$h = C \times H \tag{5}$$

C is a constant value that compares the height of objects that need to be shown by the camera with the overall height of an object or actor (see Table 2.1).

Table 1: Shot type and C value.

Shot Type	C
Medium shot	1
Close-Up	1/5
Full shot	2
Close shot	1/7
Medium full shot	1/4

4. RESULTS

To test our system, we created a scenario and behavior tree for camera placement and shot selection in the virtual world. The environment was built using unreal development kit. Commercial game engine has been quiet successful to develop engaging and entertaining virtual environment [2]. The testing environment consisted of one camera and three actors. The animation test was run in real-time to test how the system was running and whether the system was able to determine the type of shot used, the angle, and how the scene changes; for example, can the system adapt if the number of actors is increased.

4.1 Model and Scenario

The event updater in the system was designed to recognize any changes that occur in a scene. Testing is done by running a real-time animation and behavior tree module that was previously created (see Figure 10 and Figure 11).

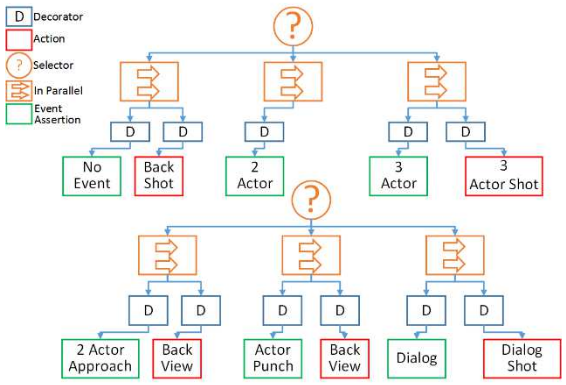


Figure 10: Behavior tree module.

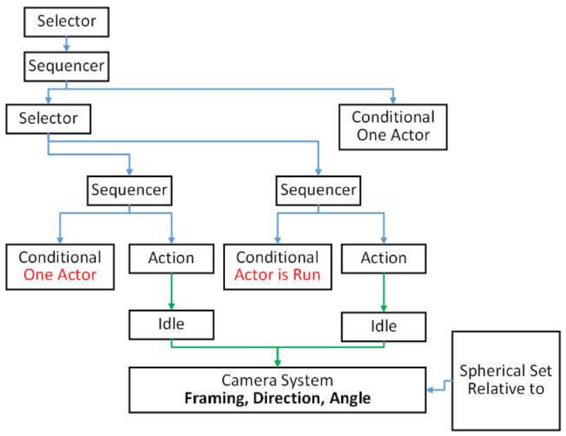


Figure 11: Detailed behavior tree module.

Figure 12 shows a module to count the number of actors in the scene, this module will determine the proper action will take such as if there are two actors and they are doing dialog scene.

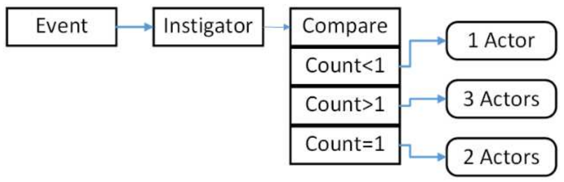


Figure 12: Spawn actor module.

Figure13, Figure 14, and Figure 15 respectively shows a module to handle events that occur in a scene, such as conversation, fighting, and general movement events.

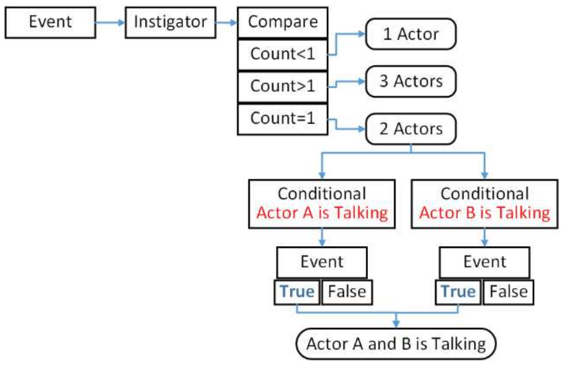


Figure 13: Dialog event module.

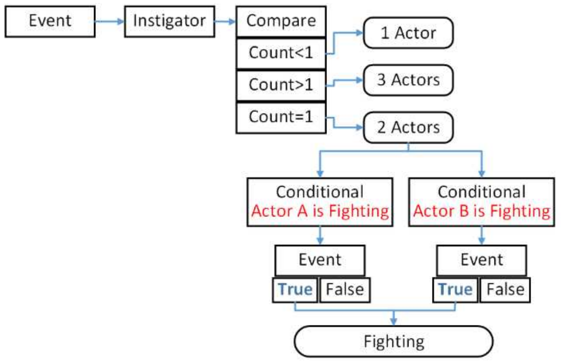


Figure 14: Fighting event module

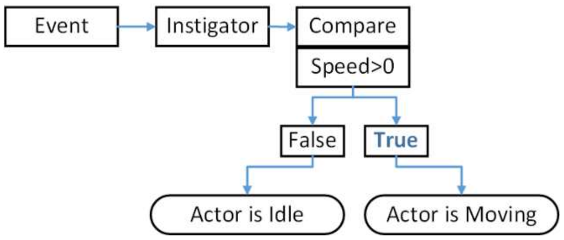
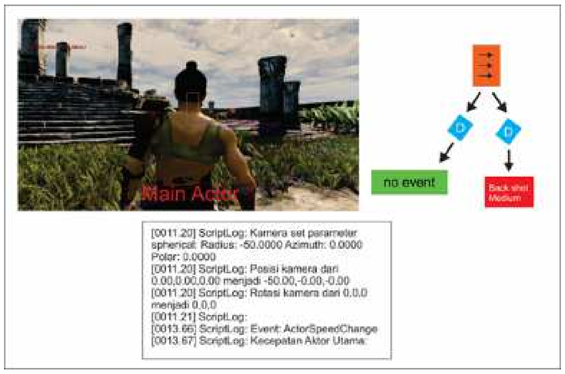


Figure 15: Movement event module

4.2 Event and Animation Test

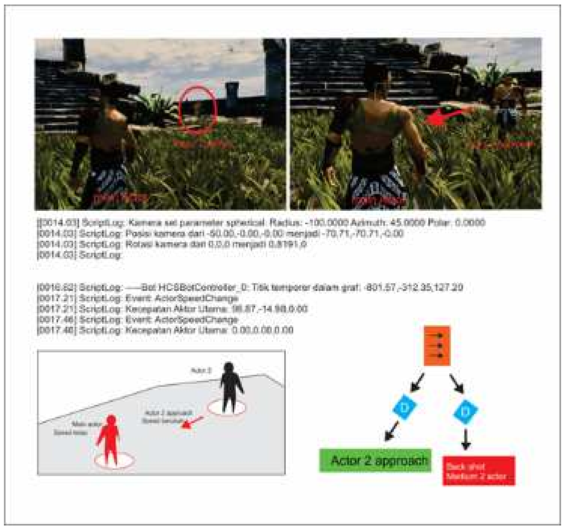
When the system starts, only one actor appears on the screen. In this case, spawn actor module detected that there in only one actor present at the scene, then selector detected that the actor is not doing either dialog or movement, so using the system medium back shot is automatically selected by the behavior tree module (see Figure 16).



```
[0011.20] ScriptLog: Kamera set parameter
spherical: Radius: -50.0000 Azimuth: 0.0000
Polar: 0.0000
[0011.20] ScriptLog: Posisi kamera dari
0.00,0.00,0.00 menjadi -50.00,-0.00,-0.00
[0011.20] ScriptLog: Rotasi kamera dari 0,0,0
menjadi 0,0,0
[0011.21] ScriptLog:
[0013.66] ScriptLog: Event: ActorSpeedChange
[0013.67] ScriptLog: Kecepatan Aktor Utama:
```

Figure 16: One Actor and no event condition

The next experiment involves the arrival of the second actor. In this case, spawn actor module detected the are two actors present at the scene, the decorator then updates the event that the actor is approaching one another and then the type of shot that can capture both actors in one screen by using behavior tree module is back view has been selected (see Figure 17).

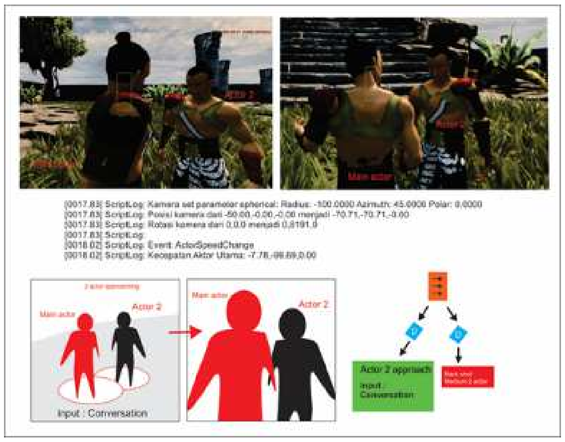


```
[[0014.03] ScriptLog: Kamera set parameter spherical: Radius: -100.0000 Azimuth: 45.0000 Polar: 0.0000
[0014.03] ScriptLog: Posisi kamera dari -50.00,-0.00,-0.00 menjadi -70.71,-70.71,-0.00
[0014.03] ScriptLog: Rotasi kamera dari 0,0,0 menjadi 0,8191,0
[0014.03] ScriptLog:

[0016.62] ScriptLog: ----Bot HCSBotController_0: Titik temporer dalam graf: -801.57,-312.35,127.20
[0017.21] ScriptLog: Event: ActorSpeedChange
[0017.21] ScriptLog: Kecepatan Aktor Utama: 98.87,-14.98,0.00
[0017.46] ScriptLog: Event: ActorSpeedChange
[0017.46] ScriptLog: Kecepatan Aktor Utama: 0.00,0.00,0.00
```

Figure 17: Two actors approach condition.

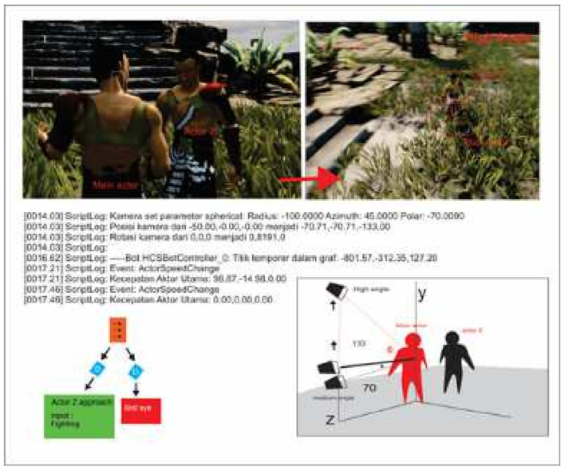
When spawn actor detect that two actors are present at the scene, and conditional status for dialog state is true then conversation event is triggered, the Over the shoulder shot is selected for this scenario (see Figure 18).



```
[0017.83] ScriptLog: Kamera set parameter spherical: Radius: -100.0000 Azimuth: 45.0000 Polar: 0.0000
[0017.83] ScriptLog: Posisi kamera dari -50.00,-0.00,-0.00 menjadi -70.71,-70.71,-0.00
[0017.83] ScriptLog: Rotasi kamera dari 0,0,0 menjadi 0,8191,0
[0017.83] ScriptLog:
[0018.02] ScriptLog: Event: ActorSpeedChange
[0018.02] ScriptLog: Kecepatan Aktor Utama: -7.78,-98.89,0.00
```

Figure 18: Two actors dialog condition.

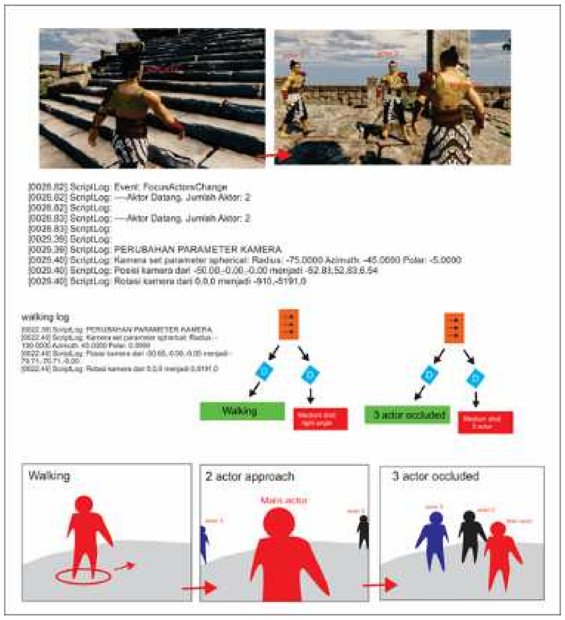
The next experiment involved a fighting event. When spawn actor detect that two actors are present at the scene, and conditional status for fighting state is true then the behavior tree module selected t the bird's-eye shot. The bird's-eye shot is selected because the viewers need a clear view of their opponent (see Figure 19).



```
[0014.03] ScriptLog: Kamera set parameter spherical: Radius: -100.0000 Azimuth: 45.0000 Polar: -70.0000
[0014.03] ScriptLog: Posisi kamera dari -50.00,-0.00,-0.00 menjadi -70.71,-70.71,-133.00
[0014.03] ScriptLog: Rotasi kamera dari 0,0,0 menjadi 0,8191,0
[0014.03] ScriptLog:
[0016.62] ScriptLog: -----Bot HCSBotController_0: Titik temporer dalam graf: -801.57,-312.35,127.20
[0017.21] ScriptLog: Event: ActorSpeedChange
[0017.21] ScriptLog: Kecepatan Aktor Utama: 98.87,-14.98,0.00
[0017.46] ScriptLog: Event: ActorSpeedChange
[0017.46] ScriptLog: Kecepatan Aktor Utama: 0.00,0.00,0.00
```

Figure 19: Two actors fighting condition.

Another test involved placement of the camera and shot selection when three actors appear on the screen simultaneously. The camera is automatically placed in the correct position to shoot all three actors within the scene (see Figure 20).



```
[0028.82] ScriptLog: Event: FocusActorsChange
[0028.82] ScriptLog: ----Aktor Datang, Jumlah Aktor: 2
[0028.82] ScriptLog:
[0028.83] ScriptLog: ----Aktor Datang, Jumlah Aktor: 2
[0028.83] ScriptLog:
[0029.39] ScriptLog:
[0029.39] ScriptLog: PERUBAHAN PARAMETER KAMERA
[0029.40] ScriptLog: Kamera set parameter spherical: Radius: -75.0000 Azimuth: -45.0000 Polar: -5.0000
[0029.40] ScriptLog: Posisi kamera dari -50.00,-0.00,-0.00 menjadi -52.83,52.83,6.54
[0029.40] ScriptLog: Rotasi kamera dari 0,0,0 menjadi -910,-8191,0
```

walking log

```
[0022.39] ScriptLog: PERUBAHAN PARAMETER KAMERA
[0022.40] ScriptLog: Kamera set parameter spherical: Radius: -
100.0000 Azimuth: 45.0000 Polar: 0.0000
[0022.40] ScriptLog: Posisi kamera dari -50.00,-0.00,-0.00 menjadi -
70.71,-70.71,-0.00
[0022.40] ScriptLog: Rotasi kamera dari 0,0,0 menjadi 0,8191,0
```

Figure 19: Three actors condition.

The experiment in our system is aimed to create machinima generated animation are based on the story of Gajah Mada from Kingdom of Majapahit who dream of uniting Nusantara. Camera control is done automatically within the virtual world in near real time condition when user move the characters or press a key for triggering the events such as walking, doing dialogue with another character or doing fighting with the enemy.

4.3 Performance Test

The performance evaluation is done by examining how long it takes the camera to move for framing, calculate the shot direction, and determine the camera angle. Table 1 shows the camera performance evaluation in milliseconds on a setup machine of Intel core i7@3.2Ghz, 16 GB of RAM, and NVidia GeForce 980 GTX graphic card.

The execution for performance test started when the level is loaded and the character is spawned in the virtual world. During the character movement and when the user presses the button to trigger an event the system automatically counts the running time since the user pressed the button until the camera take the place in the machinima world.

Table 1: Camera Placement Running Time.

No	Camera Step	Time(ms)
1	Camera orientation	0.09874
2	Camera direction	0.03783
3	Framing	0.01314
4	Angle	0.01050
5	Projection size	0.09346
6	Location targeting	0.09201

The next performance evaluation examines how long it takes for the camera to make a different shot selection based on the behavior module or a triggered event. Table 2 shows the camera performance evaluation in milliseconds.

Table 2: Camera Shot Running Tim.

Shot Type	Time(ms)
Idle back shot	0.04135
Walking medium shot	0.04311
Back shot spawn	0.04219
Back shot new actor approach	0.04025
Back shot fighting	0.04421
Walking uphill	0.04225
Medium uphill	0.04221
Back shot uphill	0.04331
Occlusion medium two actors	0.04522
Occlusion back shot three actors	0.04677

5. CONCLUSION

In this paper the we have presented a prototype system to automatically generate animation video and shot selection for machinima. The system takes an event that occurred in a virtual world and combines it with a cinematography technique to acquire machinima-generated animation. Our system extends the virtual camera computation approach to extract events from a given scenario, solve problems with camera placement, and determine virtual camera placement and shot selection to create an animation in machinima.

The system exploits the ability of a behavior tree to structure shot selection in a meaningful way. We used Unreal Development Kit game engine for creating the virtual world and scene.

The limitation of this work are the actor involved in the scene is limited up to three actors and the system was tested only for three scene or scenario which are the idle event, dialog event, and the fighting event.

We also planning to improve the camera control placement and shot selection for more than three actors and extend the scenario involving other moving objects and unpredictable events for future works.

REFRENCES:

[1] D. A. Prima, B. B. Ferial Java, E. Suryapto and M. Hariadi, "Secondary Camera Placement using Behavior Trees," in *2013 International Conference on QiR (Quality in Research)*, Yogyakarta, 2013.

[2] G. T. Hanold and M. D. Petty, "Enhancing a Commercial Game Engine to Support Research on Route Realism for Synthetic Human Characters," *International Journal of Computer Games Technology,* 2011.

[3] R. Ranon, L. Chittaro and F. Buttusi, "Automatic Camera Control Meets Emergency Simulation : An Application to Aviation Safety," *Computer & Graphics,* 2015.

[4] D. Arijon, Grammar of the Film Language, Hasting House Publishers, 1976.

[5] D. Markowitz, A. Shoulson, N. I. Badler and J. T. Kider.Jr, "Intelligent Camera Control Using Behavior Trees," in *4th International Conference, MIG 2011*, Edinburgh, 2011.

[6] D. B. Christianson, S. E. Anderson, L. W. He, D. H. Salesin, D. S. Weld and M. S. Cohen, "Declarative camera control for automatic cinematography," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[7] S. M. Drucker, T. A. Galyean and D. Zeltzer, "Cinema : a system for procedural camera Movements," in *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, New York, 1992.

[8] C. Callaway, E. Not, A. Novello, C. Rocchi, O. Stock and M. Zancanaro, "Automatic cinematography and multilingual NLG for generating video documentaries," *Artificial Intelligence,* p. 57–89, 2005.

[9] D. Friedman and Y. A. Feldman, "Automated cinematic reasoning about camera behavior," *Expert Systems with Applications,* p. 694–704, 2006.

[10] R. Haworth, S. S. Tagh Bostani and K. Sedig, "Visualizing Decision Trees in Games to Support Children's Analytic Reasoning: Any Negative Effects on Gameplay?," *International Journal of Computer Games Technology,* 2010 .

[11] R. Baumgarten, . S. Colton and M. Morris, "Combining AI Methods for Learning Bots in a Real-Time Strategy Game," *International Journal of Computer Games Technology,* 2009.

[12] B. Hawkins, Real-Time Cinematography for Games, Hingham,Massachusetts: Charles River Media, INC, 2005.

[13] R. Barsam and D. Monahan, Looking at Movies: An Introduction to Film, New York: W.W Norton & Company, 2010.