I.J. Intelligent Systems and Applications, **2018, 8, 1-10**
Published Online August 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2018.08.01

Multi-Character Fighting Simulation

Sukoco
Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences
Universitas Gadjah Mada, Yogyakarta, 55281, Indonesia
Department of Informatics, Universitas Surakarta, Surakarta, 57772, Indonesia
E-mail: pak_koco@yahoo.com

Retantyo Wardoyo and Agus Harjoko
Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences
Universitas Gadjah Mada, Yogyakarta, 55281, Indonesia
E-mail: rw@ugm.ac.id, aharjoko@ugm.ac.id

Mochamad Hariadi
Department of Electrical Engineering, Faculty of Industrial Technology,
Institut Teknologi Sepuluh Nopember (ITS), Surabaya, 60111, Indonesia
E-mail: mochar@ee.its.ac.id

Received: 17 April 2017; Accepted: 20 December 2017; Published: 08 August 2018

***Abstract*—**In the development of and research into multi-character fighting computer games, Non-Player Characters (NPCs) frequently seem less intelligent owing to them having a single focus. As such, multi-character fighting becomes one-on-one fighting; one character will encounter another character only once the previous opponent is defeated. This study develops a new model in multi-character fighting, in which each NPC can simultaneously fight against many characters. Following this model, each character becomes an agent that makes his own decisions. The first advantage of this model is the integration of multi-character behaviors in fights. Each character can seek out enemies/opponents, select one target opponent, avoid obstacles, approach the target opponent, change the target opponent, and then defeat the opponent or be defeated by the opponent; in other words, each character can thus fight against many opponents. All of the behaviors in the fight take place automatically. The second advantage of this model is that each character does not only focus on the opponent being targeted, but also on the other opponents surrounding him. Each character can move from one opponent to another, even when the target opponent is not yet defeated. The third advantage of this model is that each character can move to another fight cluster, thus ensuring that fights seem more dynamic. This research has experimented with the model using a 3D application that can run on personal computers or smart phones.

***Index Terms*—**Multi-character fighting, NPC, 3D simulation.

I. Introduction

There is high demand for creating fight scenes in the film, television, and game industries. Although fighting is an event of continuous human interactions, it is not easily simulated on computers [1]. One important element in video games is NPCs (non-player characters), characters that are controlled not by a player, but by the computer. Research into NPCs has focused primarily on the quality of their behavior [2]. Non-player characters improve the gameplay experience [3], and their visual appearance in a game has a direct effect on player performance [4]. While it is relatively easy to simulate simple, individual NPC behaviors, it is much more difficult to create meaningful interactions between them [5].

In game applications, NPCs in multi-character fight scenes frequently appear foolish, as despite being close to opponents they will do nothing. They will focus only on one opponent, even when other opponents are closer. Therefore, although fight scenes appear to involve multiple characters, they are essentially one-on-one; NPCs will only switch to another opponent once the one being fought is defeated.

Animating scenes with multi-character interactions can be a particularly complex process [6], especially multi-character fight scenes. So far, research into multi-character fights is still limited. Also, multi-character fight simulation studies have several limitations. In simulation by Shum et al. [7], it seems that characters only attack or avoid attacks automatically. This is limited to generating scenes where multiple characters continuously interact and require enemies to be defeated after interaction [8]. As such, NPCs will only switch their targets if the opponent being fought is defeated.

Shum et al. [7] simulates multi-character fights using motion capture data instead of an agent-based approach. In creating his simulation, Shum uses more settings from an interaction patch. This has weaknesses in simulating multi-character fights because it only generates one-on-one fights, and then combines these fights to form "multi-

character" fights. Characters continue to change their opponents only once the opponent being fought is defeated.

Agent system has many advantages in solving many problems [9]. The primary objective of this paper is to establish a model so each character can, as an agent, fight many opponents simultaneously. Therefore, each character can switch from one opponent to another, even when the opponent presently being fought has not yet been defeated. The advantage of this model is that multi-character fighting can take place automatically.

In this model, each character acts as an agent that can independently seek out and select opponents. Each character can fight many opponents and decide to choose one available opponent to be confronted, as each character focuses not only on one opponent but also on those surrounding him. The model functions both with low and high character populations. In implementing this model, a 3D simulation has been developed to simulate multi-character fighting.

The paper is organized as follows. Section I describes the problem statement, the research objective, and the organization of the paper. Section II provides related work, includes previous studies of interaction simulation and fight simulation. Section III explains the proposed model. Section IV describes simulation result and discussion. Section V presents the conclusion of this research.

II. Related Works

A. Interaction Simulation

Much research has been conducted in interaction simulation. In previous studies, many have emphasized the simulation human agents' movement. Zhao et al. present a simulation of pedestrians' backtrack and the waiting behavior [10]. Heliovaara et al. present a model for agents' behavior in counterflow situations [11]. Pouke et al. calibrate fluctuations of pedestrian traffic in a random model [12]. Li et al. animate large crowds using existing examples of groups motions' by applying an enhanced copy and paste technique on characters [13].

Hofinger et al. create models of human factors, including physical, cognitive, motivational and social variables, in evacuation [14]. Wong et al. present an algorithm to compute the optimal route for each local region. The system is to reduce congestion and maximize the number of evacuees arriving at exits in each time span. They also simulate crowd movements during route optimization [15].

Weiss et al. present a crowd simulation method that runs at interactive rates for hundreds of thousands of agents. The simulation animates a sparse and dense group of agents at interactive rates. It optimizes evacuation routes based on crowd simulations [16].

Interaction among individuals further develops into the interaction of agents with another object, as when Safonova and Hodgins simulate a man walking and taking a ball from the ground. They use A^* search to get an optimal solution in a graph that satisfies the user's specification [17]. Jablonski et al. present pedestrians (agents) that move, watch stores, enter stores, and reappear from stores. Each agent has their own personal interests and needs, which affects its objectives and interactions with the surroundings. Genetic algorithms are used to animate the dynamic behavior of the environment and the knowledge spreading [18]. Mankovecky simulates pedestrians and obstacles in virtual cities. His model is based on the Social Forces Model and improved the model using Monte Carlo simulation [19]. Henry et al. simulate a crowd of humans passing through complex obstacles. They represent the crowd with a deformable mesh, and allow the user, via multitouch input, to identify high level movements and formations that are important for context delivery [20]. Fata et al. simulate pilgrims' movements circling the Kaaba (*Tawaf*). Each agent is developed with some parameters such as: age, gender and intention outlook, in order to simulate the *Tawaf* crowd animation [21]. Vaillant et al. extend a quadratic program-based task-space character control approach to multiple characters interacting with each other and with objects [22]. Xhao et al. propose to use the interaction bisector surface (IBS) between the body and the object as a feature of the interaction [23].

Hyun et al. use rewriting rules and grammar parsing to synthesize the three-dimensional animation of multiple characters. The simulation demonstrates animating basketball games from drawings on a tactic board [24]. Kusuma et al. [25] purpose a new crowd simulation model for traditional markets. The crowd model includes simplified movement and unplanned purchasing models.

Bosse et al. simulate a real-life incident that happened on May 4, 2010, in Amsterdam [26]. Durupınar et al. add a psychological element to create mobs' collective misbehavior [27]. Park et al. propose a decision support system called SimCrowdControl has been developed using the technique of agent-based modeling and simulation (ABMS). The simulation shows the polices block the hostile rioters to protect core downtown areas[28].

B. Fight Simulation

There are several studies in fight simulation. Study of characters' interactions through fighting have been conducted. In general, fight simulations develop beginning with one-on-one fights, then develop into one-against-many fights and ultimately many-against-many fights.

Zordan and Hodgins simulated motion that responds to attacks on the upper body of a human using motion capture data [29] and implemented their research in the form of a boxing animation between two people. They succeeded in simulating the reactions of people who are hit and beaten in boxing, but were unable to realistically show the intensive interactions between two competing characters. Their research was also implemented using table tennis and fencing.

Lee and Lee developed interaction between agents for a boxing simulation using data obtained from motion

capture. Their developed model successfully selected the motion with minimal runtime cost, and their method was based on reinforcement learning allowing autonomous agents to learn certain behaviors through trial and error [30]. However, this simulation was still limited to the interaction of two characters boxing.

Kovar and Gleicher simulated kicking, based on motion capture data. The developed model was meant to find and select appropriate motion for motion capture sequences [31], and it had not shown a fight yet.

Komura conducted research on a character's reactions to a blow that hit it. Although animations generated from motion capture data with a momentum-based inverse kinematic were capable of displaying various reactions that occur after a character receives a punch [32], this is not a continuously animated fight.

Shum et al. simulate the interaction of two boxing characters, based on motion capture data [8]. Shum et al. developed this further in 2008 by simulating many characters with a model called interaction patch, which was used with min-max search techniques to generate a series of motions innon-player characters. The patch interaction took a microlook at agents' interactions with other agents. Interaction between agents was obtained by creating a patch table in the form of data obtained from motion capture. Interaction between agents was done with less computation [7]. This interaction patch, however, has limitations in simulating many characters with continuous interaction [33].

Arikan et al. simulated push effects on the upper part of the human body [34]. Their method could find visually plausible transitions that did not necessarily correspond to similar motions in terms of configuration. This method could start with a limited set of recorded motions, which could be modified to serve different pushes on the upper body.

Ishihara et al. evaluate the performance of Monte-Carlo Tree Search (MCTS) in a fighting game artificial intelligence and proposes an improvement for the algorithm. They improve the Monte-Carlo tree search in fighting game artificial intelligence and enhance it with Roulette Selection and a rule base. They proposed improvements, especially Roulette Selection, effectively enhance the MCTS based method [35].

III. Multi-Character Fighting Model

This study aims to establish a new model of simultaneous fighting involving multiple characters. In fulfilling the objective of this study, a multi-character fighting model has been built.

A. Model and State

In this model, each character is an NPC that can seek out and select opponents. Each character avoids any obstacles that block his way. Each character can switch from one opponent to another without waiting for the current opponent to be defeated.

The model considers every agent as the same type of character, without any variation in height, weight, strength, initial health, and damage points. Therefore, each character can make equal decisions during a fight. Fig. 1 shows this model.

This model consists of 11 states, as follows:

1) *Idle*: The state in which this model starts. In this state, a character does not move from his place.
2) *Search Opponent*: State in which the character seeks an available opponent that is a predetermined distance from him. The opponent is another character that has a different tag and is a certain distance from him. The list covers all opponents near the character.
3) *Choose Opponent*: State in which the character selects one opponent randomly from the available list.
4) *Search Enemy*: State in which the character looks for an enemy on the battlefield. An enemy is a character that has a different tag than the character. Enemy searching is performed over the entire battlefield, no matter the distance.
5) *Choose a Candidate Opponent*: State in which the character selects one enemy out of all enemies to be a potential opponent. The candidate opponent chosen is the nearest enemy.

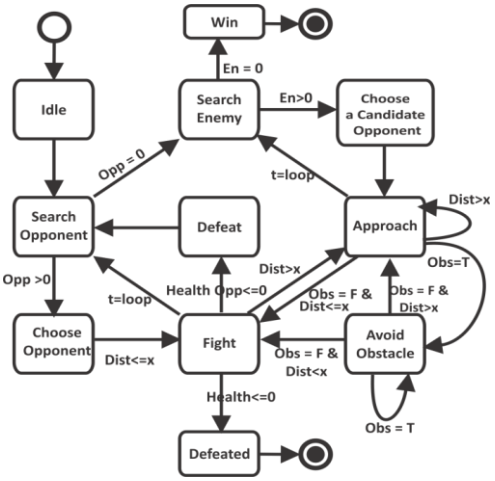


Fig.1. Model of multi-character fight

6) *Approach*: State in which the character approaches the potential opponent selected.
7) *Avoid Obstacle*: State in which the character avoids an obstacle that is preventing him from reaching his opponent. In this state, the character moves to the right or left.
8) *Fight*: State in which the character is fighting against an opponent. In this state, the character tries to defeat the opponent.
9) *Defeat*: State in which the character successfully defeats the opponent.
10) *Defeated*: State in which the opponent defeats/kills the character. The defeated character is considered dead/lost.
11) *Win*: State in which there are no more enemies on the battlefield.

The multi-character fighting model is initiated from the Idle state. Characters check whether or not there are any opponents (Search Opponent state). A character recognizes that nearby opponents using a circle that intersects with other character's colliders. The use of the intersection between the circle and colliders identify the tag of other characters. Fig. 2 illustrates the pseudocode of the main proposed algorithm.

When in Search Opponent state, if there are some potential opponents (sumOpponent>0), the character selects an opponent to be encountered (Choose Opponent state). If only one potential opponent is available, he automatically becomes the opponent and enters the Fight state.

If there is more than one opponent, the opponent is selected randomly. In Fight state, in every looping period, the character checks the opponents near him and chooses one to be encountered.

If a character checks for an opponent but finds none (sumOpponent=0), he will search for enemies over the whole battlefield (Search Enemy state). Enemy searching is performed by searching for a character that is tagged differently. If there is no enemy found (En=0), then the character's group wins (Win state).

In Search Enemy state, when he finds an enemy, that enemy becomes the character's target opponent. When a character finds more than one enemy, he will select the one closest to his position to become the target opponent (Choose a Candidate Opponent state). After finding a target opponent, the character moves closer to the target opponent (Approach state). In Approach state, the character searches for an enemy in every looping period.

The character approaches the target opponents, and after reaching a certain distance (Dist<x), he enters Fight state. In this model, x is two length units.

```
Opponent[] myOpponents = OverlapSphere
  (center, radius, tagX);
//radius = Maximum distance to involve the enemy
  in the fight
sumOpponent = myOpponents.Length ;
a = random(0, sumOpponent);
if (!isFight) then
  if (sumOpponent =0)
    FindClosestEnemyAsCandidateOpp(out
    targetOpponent);
    LookTarget(targetOpponent);
    if (targetOpponent)
      if  frontObstacle then avoidObstacle
      else goStraight;
    else Win;
  else
    if (sumOpponent=1)
        targetOpponent = myOpponents [a];
        LookTarget(targetOpponent);
        Play (FightAnimation);
        isFight = true;
    else if (sumOpponent> 1)
        targetOpponent = myOpponents[a];
        LookTarget(targetOpponent);
        Play (FightAnimation);
        isFight = true;
    else
        If frontObstacle then avoidObstacle
        else goStraight;
    LookTarget(targetOpponent);
```

Fig.2. Main pseudocode of multi-character fighting

If there is an obstacle (object/human) between the character and the target opponent (Obs=True), then he should enter Avoid Obstacle state following the obstacle avoidance algorithm described in Subsection III.B.

In avoiding an obstacle, if the distance between the character and his enemy is greater than two length units, and there is no obstacle (Obs=F & Dist>x), then the character will approach the opponent (Approach state).

If the character's distance from the opponent is less than or equal to two (Dist≤x), and there is no obstacle (Obs=False), he will enter Fight state and thus approach the opponent to attack.

In Fight state, when more than one opponent is available, an opponent selection is performed randomly among available opponents. Every looping period, each character updates his list of opponents surrounding him. The attack consists of two kinds of motion, kicks and punches, which are performed randomly.

A character's attack reduces the health of the opponent. If the health of a character reaches zero (Health ≤ 0), then he enters Defeated state and is considered defeated. In this simulation, the defeated character is simulated as disappearing from the battlefield. Details about the health system are provided in Subsection III.C.

If a character successfully defeats an opponent, he enters Search Opponent state again. If he finds an opponent, then he will fight again, until all of the opponents are defeated. If there is no more enemies remain (En=0), the character's group is considered victorious (Win state).

B. Obstacle Avoidance System

Often, a character is impeded by an obstacle when moving towards an opponent. Such obstacles may be the character's allies or other objects. Often a character's allies intuitively block other characters from reaching opponents. If that happens, a character must avoid the obstacle. Fig. 3 shows the obstacle avoidance pseudocode.

```
If (ObstacleAtFront()) then
  if  (isRight) then
    if (RightEmpty()) then goRight()
    else if (RearRightEmpty()) then goRearRight()
    else if (LeftEmpty()) then
        isRight = false ;
        goLeft();
    else if (rearLeftEmpty()) then
        isRight = false;
        goRearLeft();
  else
    if (LeftEmpty()) then goLeft()
    else if (rearLeftEmpty()) then goRearLeft ()
    else if (RightEmpty()) then
        isRight = true;
        goRight ();
    else if (RearRightEmpty ()) then
        isRight = true;
        goRearRight();
else
  goStraight();
end
```

Fig.3. Obstacle avoidance pseudocode

The obstacle avoidance is performed through the value isRight, a boolean variable with a randomly determined

value. IsRight is used to determine a character's tendency to move to the right or left.

If a character (white circle) moves forward and finds an obstacle, and if isRight is True, that character will move to the right. If there is an obstacle to the right, then the character will move to the rear right. If there is an obstacle there, the character will move to the left. If there is an obstacle to the left, the character will move to the rear left (see Fig. 4 (a–e)).

Conversely, if isRight is False, the character checks the left first. If there is no obstacle to the left, the character then moves to the left. If there is an obstacle to the left, the character moves to the rear left. If there is an obstacle to the rear left, the character moves to the right. If there is an obstacleto the right, the character will go to the rear right (see Fig. 4 (f–j)).

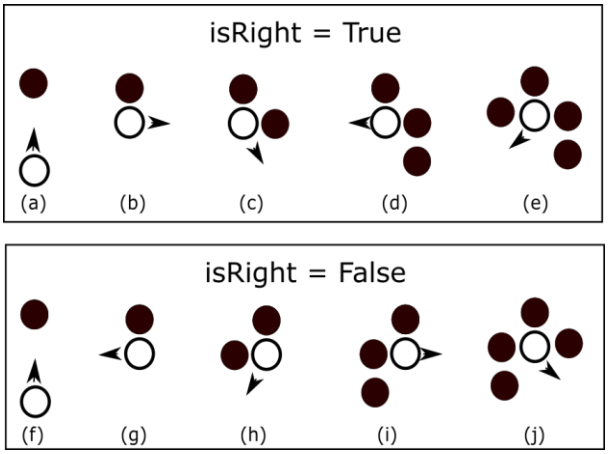


Fig.4. Avoiding obstacle system

C. Damage and Health System

In game development, there must be disequilibrium, winners and losers. In fighting games, a character's health is reduced when he is attacked by an opponent. Fig. 5 illustrates the damage and health pseudocode used.

```
if (health - damage > 0)
    health = health - damage;
    healthBar=(health/initHealth)*scale;
else
    Destroy;
```

Fig.5. Damage and health pseudocode

Damage in this model uses a Damage Per Second (DPS) system. DPS is the value of damage performed by a character over one second. A Damage Point (DP) is the power of a single attack. Usually, every type of character has different damage points, but in this model, all characters have the same damage points. Hit Speed (HS) is the amount of time used for a single attack. In this model, all characters have the same Hit Speed. DPS is calculated by:

$$DPS = DP / HS \quad (1)$$

The life of a character is shown in the form of a health bar. In this model, all characters have the same initial health. If a character is attacked, then his health is reduced by an amount corresponding to the opponent's DPS value; when a character attacks his opponent, his health is unchanged (it does not decrease). In this model, the type of attack (a punch or a kick) does not affect the character's damage points.

IV. Simulation Result and Discussion

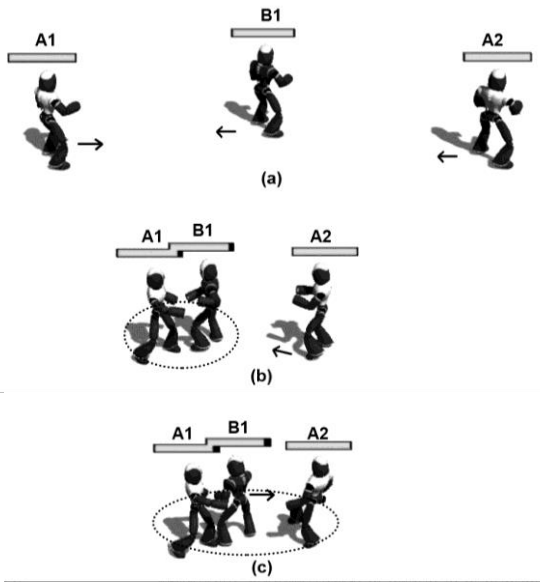
Personal computers and smart phones have been used to implement this model. This was rendered using the Unity game engine.

In this simulation, a fight takes place between two groups, Group A and Group B. Group A wears white clothes while group B wears black clothes.

Every character has the same behavior. The character searches for opponents, selects one target opponent, and fights with the target opponent. The character also searches for enemies, chooses a potential opponent, moves to approach the opponent, and avoids obstacles.

When searching for opponents, the distance searched is less than or equal to two length units ($x \leq 2$). If the distance is less than or equal to two, the character enters the Fight state. In the Approach state, if the distance to the opponent is less than or equal to two, the opponent will also enter the Fight state.

Every character has the same initial health, 1000. Each character has a damage point of 30 and requires 0.8 seconds per attack. As such, each character's DPS value is 30 / 0.8 = 37.5; in other words, each character reduces his opponent's health by 37.5 points every second of attacking. If a character attacks for 5 seconds, then he will reduce the opponent's health by 5 x 37.5 = 187.5 points.



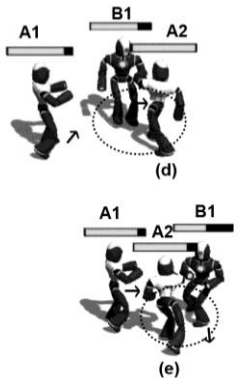
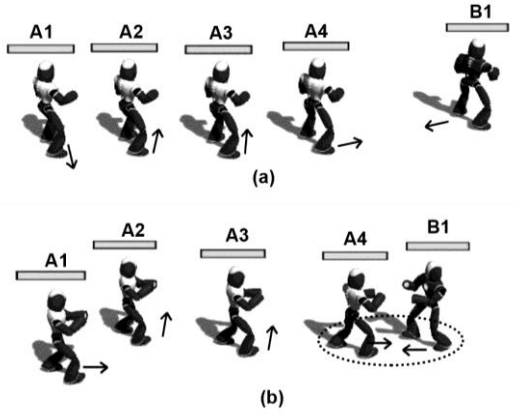


Fig.6. Three characters fight

In a personal computer implementation (see Fig. 6), a simulation of three characters fighting in one-on-two fighting is shown. Fig. 6(a) shows an Idle state, in which each character can select his opponent. Each character will select the closest enemy. A1 and A2 have only one enemy who can potentially become their opponent (B1). B1 has two enemies, A1 and A2. As A1 is nearer than A2, then B1 chooses A1 as his opponent. B1 moves to approach A1. If the distance between A1 and B1 is less than two units, A1 and B1 attack each other as opponents in a Fight state (see Fig. 6(b)). When A1 and B1 are in a Fight state, A2 goes forward to approach B1.

In Fig. 6(c), A2 approaches B1. B1 thus has two opponents on his list, A1 and A2. In the Fight state, at a certain time B1 randomly chooses a target to attack. In Fig. 6(d), B1 changes his target opponent from A1 to A2. In Fig. 6(e) A2 and B1 are in Fight state, A1 goes forward to approach B1.

Fig. 7(a–e) shows a simulation of the obstacle avoiding system. The scene takes the form of a fight in which four characters face off against one character. The simulation is initiated (see Fig. 7(a)) when A1, A2, A3 and A4 approach B1. A4 goes directly to B1 because he has no obstacles ahead of him. A3 avoids A4 by moving to the left as he approaches B1. A2 also avoids A3 by moving to the left. A1 avoids A2 by moving to the right.



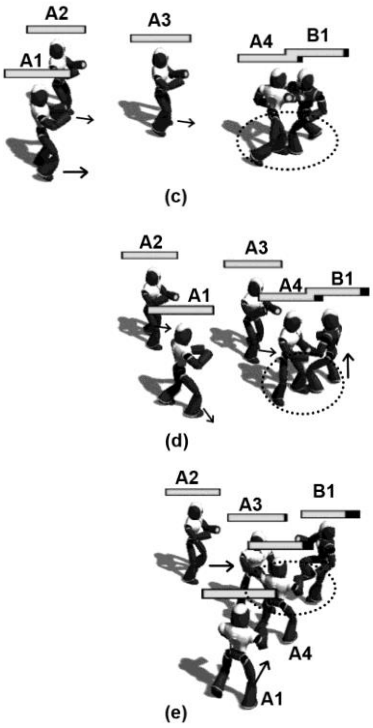
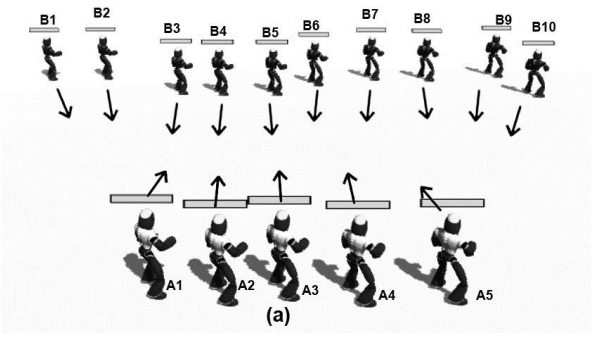


Fig.7. Avoiding obstacle simulation

After A2 moves to left, A1 has no obstacle ahead of him. A1 may go forward (see Fig. 7(b)). In Fig. 7(c) A1, A2 and A3 move forward. In Fig. 7(d), A1 moves to the right and A2 moves forward to B1.

Fig. 7(a-e) shows that the model's obstacle avoidance has worked. If an obstacle is in front of the character, the character automatically makes a decision to move left or right. If the character faces no obstacles, he moves straight.

Fig. 8 shows a multi-character fight involving 15 characters. This fight consists of two groups, Group A (five characters) and Group B (ten characters). In Fig. 8(a), all characters are in an Idle state. When the simulation begins, all characters search for the closest enemy. Each then approaches the potential opponent (see Fig. 8(b) and (c)). If the distance between a character and his candidate opponent is less than two distance units, they are in a Fight state (see Fig. 8(d)).



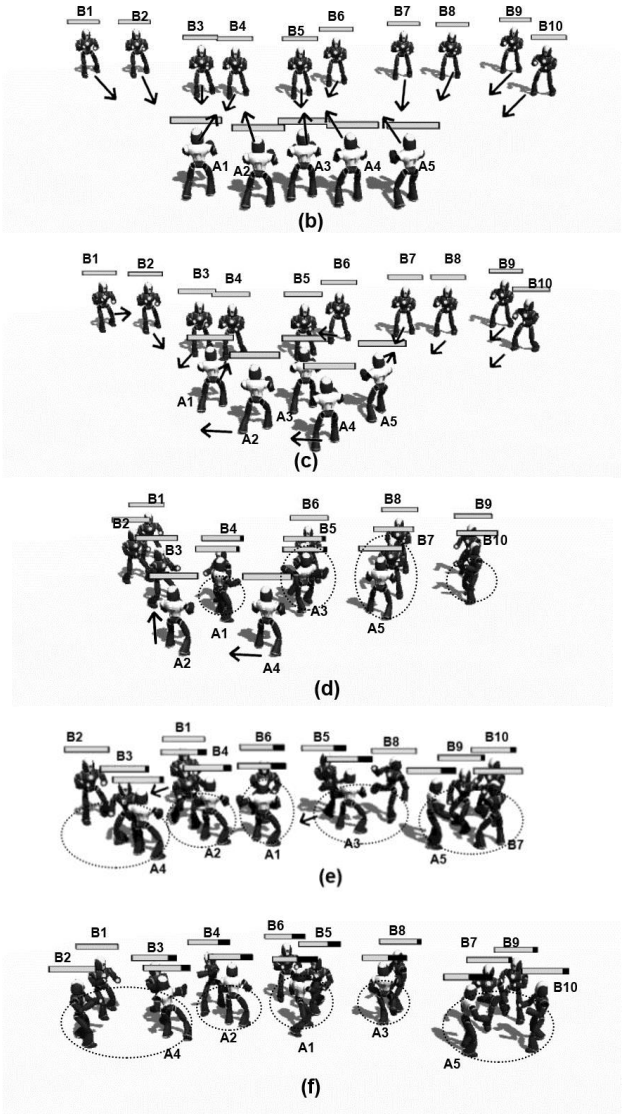


Fig.8. Changes enemy in fighting

In Fig. 8(e), character A1 has one opponent, A2 has two opponents, A3 has two opponents, A4 has two opponents, and A5 has three opponents. In the Fight state, each character may change his target opponent. B5 changes his target opponent from A3 to A1. B1 changes his target opponent from A2 to A4. This also means that B1 and B5 change their fight cluster. A1 changes his target from B6 to B5. A4 changes his target from B5 to B8. A5 changes his target from B9 to B7. Finally, in Fig 8(f), character A1 has two opponents, A2 has one opponent, A3 has one opponent, A4 has three opponents, and A5 has three opponents.

Fig. 8 indicates that in this model, a character searches for enemies, selects a potential opponent, and approaches this potential opponent. Characters may sometimes have one opponent, and sometimes have several opponents. Eventually, each character will enter the Fight state to defeat his target opponents or be defeated by his target opponent. In a fight cluster, each character can move from one opponent to another, without waiting for the current opponent's defeat. Each character can move from one fight cluster to nearby fight cluster.

In the Fight state, each character can change his target opponent randomly in his fight cluster, even when his target opponent is not defeated. Each opponent can then fight all opponents simultaneously. Sometimes a character moves from one fight cluster to another, thus making the fight scene more dynamic.

With this model, then, multiple NPCs can thus fight each other. In Fig. 9, a fight occurs between two groups, with each group consisting of many characters. Fig. 9(a) shows an Idle state position, a character does not move from his place. After Idle state, all character check whether or not there are any opponents. If a character has an enemy nearby, the enemy will become an opponent. Each character may thus encounter one or more characters simultaneously (see Fig. 9(b) and 9(c)). If a character encounters many potential opponents, he selects randomly a target opponent to be encountered. If an opponent is defeated, the character will search for another enemy as a new target opponent (see Fig. 9(d)). If a character is defeated, it will be dead and dissapeared.

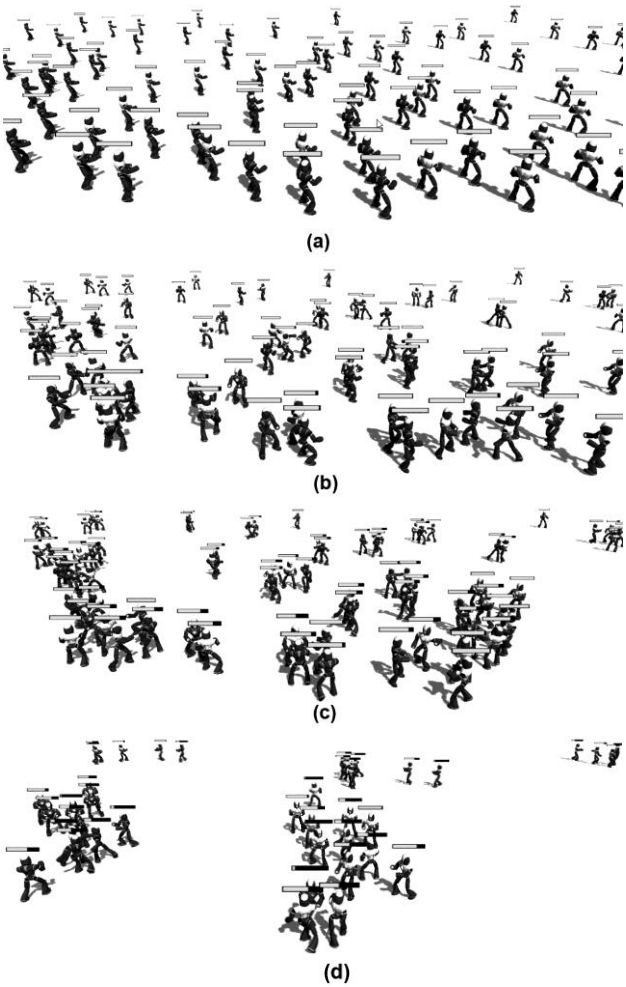


Fig.9. Crowd fighting

The implementation of this model utilized a smart phone. The smart phone has dual core procesor 1.3 GHz and RAM 512 MB. Fig. 10 shows the implementation. There are two groups fighting each other; with each

group consisting of ten characters. As with the experiment using a personal computer, the implementation of this model on a smart phone allowed fights to be run simultaneously.

Fig.10. Twenty characters fight in smart phone

Five game players have tested the implementation of this model, and they have the notion that it would be positively applied in a game simulation. They hope that this model can make fight scenes in games more realistic.

V. Conclusion

This research has simulated fights between multiple-characters. Each character in this research is an NPC (non-player character). This model can simulate fights that can be run by the system and can be run simultaneously. Each character can search for enemies or opponents and select target opponents. Each character can avoid obstacles, approach target opponents, change target opponents, defeat opponents or be defeated by opponents; as such, each character can fight against many opponents.

Each character may encounter many characters at a time. Each character focuses not only on one opponent, but also those surrounding him. Each character can move from one opponent to another in a fight cluster, without waiting for the current opponent's defeat. Each character can also move from one fight cluster to another. The movement of characters in one cluster or between clusters will make fight scenes more dynamic. This research has implemented this model as a 3D simulation that can be run on a personal computer or a smart phone.

In the future study and game development, each character may have different power, skills, and thoughts to create variations in each character's behavior during the fighting. In game development, hit points and damage per second can be varied for each different type of character to provide variety and avoid a monotone.

References

[1] H. Shum, and T. Komura, Generating Realistic Fighting Scenes by Game Tree, *Conference: SCA '06: Proceedings of the 2006 ACM SIGGRAPH/ Eurographics symposium on Computer animation*, At Vienna, Austria, January 2006, ISBN: 3-905673-34-7

[2] A. Rankin, G. Acton, and M. Katchabaw, 2010, A Scalable Approach to Believable Non Player Characters In Modern Video Games, *Conference: GameOn 2010*, at Leicester, United Kingdom

[3] O. Szymanezyk, P. Dickinson, T. Duckett, From Individual Characters to Large Crowds: Augmenting the Believability of Open-World Games through Exploring Social Emotion in Pedestrian Groups, *Proceedings of Digital Games Research Association DiGRA 2011 Conference: Think Design Play*, 2011

[4] E. Carrigan, E. Kokkinara, F. Gheorghe, M. Houlier, S. Donikian, R. McDonnell, Crowd Appearance Affects Player Performance In Game Combat Scenarios, *ACM., MiG'16, Proceedings of the 9th International Conference on Motion in Games*, Burlingame, California — October 10 - 12, 2016, Pages 187-192, doi: 10.1145/2994258 .2994273

[5] M. Černý, C. Brom, R. Barták, M. Antoš, Spice it up! Enriching Open World NPC Simulation Using Constraint Satisfaction, *Proceedings of the Tenth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2014)*, AAAI Press, 2014, 16-22, ISBN:1577356810 978-1-57735-681-3

[6] J. Won, K. Lee, C. O'Sullivan, J. K. Hodgins, J. Lee, Generating and Ranking Diverse Multi-Character Interactions, *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2014*, Volume 33 Issue 6, November 2014, Article No. 219, doi: 10.1145/2661229.2661271

[7] H. P. H. Shum, T. Komura, M. Shiraishi, S. Yamazaki, Interaction Patches for Multi-Character Animation. *ACM Transactions on Graphics*, Vol. 27, No. 5, Publication date: December 8, 2008, Article No 114, doi: 10.1145/ 1457515.1409067

[8] H. P. H., Shum, T., Komura, and S. Yamazaki, Simulating Competitive Interactions Using Singly Captured Motions, *Proceedings of ACM Virtual Reality Software Technology 2007*, p. 65–72, doi: 10.1145/1315184.1315194

[9] M. S. Iraji, Fuzzy Agent Oriented Software Effort Estimate with COCOMO, *I.J. Intelligent Systems and Applications*, 2015, 08, 18-29, doi: 10.5815/ijisa.2015.08.03

[10] X. Zhao, Y. You, Y. Zhang, Evacuation Simulation of Counter-current Behavior Using Agent Model for Pedestrian Dynamics, *I.J. Education and Management Engineering 2011*, 4, 72-79, doi: 10.5815/ ijeme.2011.04.12

[11] S. Heliövaara, T. Korhonen, S. Hostikka, H. Ehtamo, Counterflow Model For Agent-Based Simulation Of Crowd Dynamics, *Elsevier: Building and Environment*, vol. 48 (2012), pp. 89-10

[12] M. Pouke, J. Goncalves, D. Ferreira, V. Kostakos, Practical simulation of virtual crowds using points of interest, *Computers, Environment and Urban Systems* 57 (2016) 118–129, 2016, Elsevier Ltd, doi: 10.1016/j.compenvurbsys.2016.02.004

[13] Y. Li, M. Christie, O. Siret, R. Kulpa and J. Pettré, Cloning Crowd Motions, *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)*, Pages 201-210, Switzerland, ISBN: 978-3-905674-37-8

[14] G. Hofinger, R. Zinke, L. Künzer, Human factors in evacuation simulation, planning, and guidance, *Elsevier Transportation Research Procedia 2* (2014) 603 – 611, doi: 10.1016/j.trpro.2014.09.101

[15] S. Wong, Y.S. Wang, P.K. Tang, and T.Y. Tsai, Optimized evacuation route based on crowd simulation, *Computational Visual Media*, Vol. 3, No. 3, September 2017, 243–261, doi: 10.1007/s41095-017-0081-9

[16] T.Weiss, A. Litteneker, C. Jiang, and D. Terzopoulos, Position-Based Multi-Agent Dynamics For Real-Time Crowd Simulation, MiG'17, November8–10, 2017, Barcelona, Spain, ACM, DOI:10.1145/3136457.3136462
[17] A. Safonova, and J.K. Hodgins, Construction And Optimal Search Of Interpolated Motion Graphs, *ACM Transactions on Graphics 3*, 2007
[18] K. Jablonski, V. Argyriou, D. Greenhill, Crowd simulation for dynamic environments based on information spreading and agents'personal interests, *Elsevier: Transportation Research Procedia 2 (2014)* 412 – 417, doi: 10.1016/ j.trpro.2014.09.046
[19] R. Mankovecky, Dynamic Simulation of Virtual Agents and Obstacles in Virtual Cities, *Proceedings of CESCG 2016: The 20th Central European Seminar on Computer Graphics*, CESCG 2016, April 24-27, 2016, Smolenice, Slovakia, ISBN: 3950253386, 9783950253382
[20] J. Henry, H.P.H. Shum, and T. Komura, Interactive Formation Control in Complex Environments, *IEEE Transactions On Visualization And Computer Graphics*, Vol.20, No.2, February 2014, p 211-222, doi: 10.1109/TVCG.2013.116
[21] A.Z.A. Fata, M.S.M. Rahim, S. Kari, Autonomous Tawaf Crowd Simulation, *Borneo Science* 36 (2): September 2015.
[22] J. Vaillant, K. Bouyarmane, and A. Kheddar, Multi-Character Physical and Behavioral Interactions Controller, *IEEE Trans Vis Comput Graph. 2017* Jun; 23(6):1650-1662. doi: 10.1109/TVCG.2016.2542067
[23] X. Xhao, M.G. Choi, T. Komura, Character-Object Interaction Retrieval using the Interaction Bisector Surface, *EUROGRAPHICS 2017*, The Eurographics Association and John Wiley & Sons Ltd., Volume 36 (2017), Number 2, doi: 10.1111/cgf.13112.
[24] K. Hyun, K. Lee, and J. Lee, Motion Grammars for Character Animation, *EUROGRAPHICS 2016*, Volume 35(2016), Number 2, Pages 103-113, doi: 10.1111/cgf.12815
[25] P. D. Kusuma, Azhari, R. Pulungan, Agent-Based Crowd Simulation of Daily Goods Traditional Markets, *I.J. Intelligent Systems and Applications*, 2016, 10, 1-10, Published Online October 2016 in MECS (http://www.mecs-press.org/), doi: 10.5815/ijisa.2016.10.01
[26] T. Bosse, M. Hoogendoorn, M. C. A. Klein, J. Treur, C. N. Vanderwal, A. van Wissen, Modelling Collective Decision Making in Groups And Crowds: Integrating Social Contagion And Interacting Emotions, Beliefs And Intentions, *Springer, Auton Agent Multi-Agent Syst(2013)* 27:52–84, doi: 10.1007/s10458-012-9201-1
[27] F. Durupınar, U. Gudukbay, A. Aman, N.I. Badle, Psychological Parameters for Crowd Simulation: From Audiences to Mobs, *IEEE Trans Vis Comput Graph.* 2016 Sep;22(9):2145-59. doi: 10.1109/ TVCG. 2015.2501801
[28] A.J. Park, S. Buckley, H.H. Tsang, V. Spicer, A Decision Support System for Crowd Control Using Agent-Based Modeling and Simulation, *2015 IEEE 15th International Conference on Data Mining Workshops*, doi: 10.1109/ICDMW.2015.249
[29] V. B. Zordan, J. K. Hodgins, Motion Capture-Driven Simulations That Hit And React, *Proceedings of ACM SIG-GRAPH Symposium on Computer Animation*, July 21-22, 2002, p. 89-96, doi: 10.1145/545261.545276
[30] J. Lee, and K. H. Lee, Precomputing Avatar Behavior From Human Motion Data, *Proc of 2004 ACM SIGGRAPH/ Eurographics Symp on Computer Animation*, 2004, 79–87, doi: 10.1145/1028523.1028535
[31] Kovar , L. dan Gleicher, M., 2004, Automated Extraction and Parameterization of Motions in Large Data Sets , *Transactions on Graphics, 23, 3 (SIGGRAPH 2004)*
[32] T. Komura, E.S.L. Ho, and R.W.H. Lau, *Animating Reactive Motion Using Momentum-Based Inverse Kinematics, Computer Animation And Virtual Worlds* 2005;16, John Wiley & Sons, Ltd., 2005, p 213–223, doi: 10.1002/cav.101
[33] H.P.H., Shum, *Simulating Interactions Among Multiple Characters*, Dissertation, University of Edinburgh, 2010
[34] O. Arikan, DA. Forsyth, JF. O'Brien, JF, Pushing People Around, *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 29–31 July 2005, Los Angeles, p 59-67, doi: 10.1145/1073368.1073376
[35] M. Ishihara, T. Miyazaki, C.Y. Chu, T. Harada, R. Thawonmas, Applying and Improving Monte-Carlo Tree Search in a Fighting Game AI, *Proceeding of the 13th International Conference on Advances in computer entertainment Technology*, ACM, Article No 27, Osaka, Japan, November 9, 2016, ISBN 978-1-4503-4773-0, doi: 10.1145/3001773.3001797

Authors' Profiles

Sukoco. Currently is pursuing his Doctoral Program in Department of Computer Science and Electronics FMIPA, Universitas Gadjah Mada, Yogyakarta, Indonesia.

He is a lecturer at Informatics Departement of Universitas Surakarta, Indonesia. He had his undergraduate degree (SSi) in Electronics and Instrumentation, FMIPA, Universitas Gadjah Mada, Yogyakarta, Indonesia in 1992, and Master (MKom) in Informatics from Universitas Dian Nuswantoro, Semarang, Indonesia in 2010; also he had his Master (MSi) in Environment Science from Universitas Sebelas Maret, Surakarta, Indonesia in 2009.

Mr. Sukoco's research areas of interest are game, animation and simulation.

Retantyo Wardoyo. He had his undergraduate (Drs) in Mathematics from Universitas Gadjah Mada, Yogyakarta, Indonesia in 1982, and Master (MSc.) in Computer Science from the University of Manchester, UK in 1990. He had his Doctoral degree (PhD) in Computation from University of Manchester Institute of Science and Technology, UK in 1996.

Dr. Wardoyo's research area of interests are database systems, operating systems, management information systems, fuzzy logics, and software engineering.

Agus Harjoko. He had his undergraduate degree (Drs) in Electronics and Instrumentation, FMIPA, Universitas Gadjah Mada, Yogyakarta, Indonesia in 1986, and Master (MSc) in Computer Science, University of New Brunswick, Canada, in 1990. He had his Doctor (PhD) from Faculty of Computer Science, University of New Brunswick, Canada, in 1996.

Dr. Harjoko's research area of interests are digital image and video processing, machine vision, and pattern recognition.

Mochamad Hariadi. He had his undergraduate degree (ST) in Electrical Engineering from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 1995, and Master (MSc) in School of Information Sciences, Dept. of Intelligent Sciences, Tohoku University, Japan, in 2003. He had his Doctoral degree (PhD) from School of Information Sciences, Dept. of Computer and Mathematical Sciences, Tohoku University, Japan, in 2006.

Dr. Hariadi's research area of interests are computational intelligence, computer vision, video and image processing, game technology, big data and internet of think.

How to cite this paper: Sukoco, Retantyo Wardoyo, Agus Harjoko, Mochamad Hariadi, "Multi-Character Fighting Simulation", International Journal of Intelligent Systems and Applications(IJISA), Vol.10, No.8, pp.1-10, 2018. DOI: 10.5815/ijisa.2018.08.01

