Monte-Carlo Tree Search Implementation of Fighting Game AIs Having Personas

Ryota Ishii
Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
is0245ve@ed.ritsumei.ac.jp

Suguru Ito
Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
is0202iv@ed.ritsumei.ac.jp

Makoto Ishihara
Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
is0153hx@ed.ritsumei.ac.jp

Tomohiro Harada
College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
harada@is.ritsumei.ac.jp

Ruck Thawonmas
College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
ruck@is.ritsumei.ac.jp

***Abstract*—In this paper, we propose a method for implementing a game AI with a persona using Monte-Carlo Tree Search (MCTS). Video games are now a powerful entertainment media not just for players but spectators as well. Since each spectator has personal preferences, customized spectator-specific gameplay is arguably a promising option to increase the entertainment value of video games streaming. In this paper, we focus on personas, which represent playstyles in the game, in particular fighting games. In order to create an AI player (character) with a given persona, we use a recently developed variant of MCTS called Puppet-Master MCTS, which controls all characters in the game, and introduce a new evaluation function, which makes each character take their actions according to the given persona, and roulette selection-based simulation to this MCTS. The results of a conducted experiment using FightingICE, a fighting game platform used in a game AI competition at CIG since 2014, show that the proposed method can make both characters successfully behave according to given personas, which were identified by participants – spectators – in the experiment.**

***Index Terms*—Monte-Carlo tree search, persona, play style, fighting game AI, FightingICE**

I. Introduction

In recent years, more than 100 million spectators watch gameplay videos (GPV) every month using streaming platforms such as Twitch and YouTube. This phenomenon suggests video games are a powerful entertainment media not just for players but spectators as well. Spectators can be divided into three groups. “Let’s Play” (LP) is one of them where spectators enjoy watching GPVs streamed by live streamers [1]. Such spectators often look for GPVs that entertain themselves by judging GPVs according to not only how well the players play them but also how entertaining those GPVs are. As a result, it is desirable that game streamers must be able to generate customized spectator-specific GPVs that match their spectators’ preferences. Due to a wide variety of spectators’ preferences, it is, however, challenging for streamers to provide such GPVs. Recently, Thawonmas and Harada proposed a novel concept of called procedural play generation (PPG) [2]. Their goal is to generate GPVs automatically and recommend those GPVs to spectators according to their preferences. To realize this concept, one needs artificial intelligence (AI) methods for automatically generating GPVs with various contents and recommender systems for recommending GPVs to spectators according to their preferences. In this paper, we cope with the AI part.

In particular, we propose a method that can automatically generate various GPVs using Monte-Carlo Tree Search (MCTS) [3], [4]. We target fighting games, one of the game genres whose GPVs are often streamed by streamers and adopted as competitions in eSports. In addition, we focus on playstyles in fighting games; such styles are called personas [5]. In order to generate GPVs where each AI player (character) has a specific persona, we adopt a recently proposed MCTS called Puppet-Master MCTS (PM-MCTS) [6] that controls all characters in the game as a base method and introduce a new evaluation function that is based on the distance between both characters and their actions. We also introduce roulette selection to the simulation part in PM-MCTS to improve the simulation accuracy. We verify whether our proposed method can generate GPVs where the characters have personas by a subjective experiment using FightingICE, a fighting game platform used in a game AI competition at CIG since 2014 [7].

II. Related Work

A. Persona

According to Tychsen and Canossa [5], persona in the game context is the playstyle of a player. They analyzed playstyles of various game players, and those playstyles were used for testing game designs. Following this definition of persona, Holmgård *et al.* [8] proposed a method for implementing AIs that behave according to specified personas using MCTS. They classified five personas in "MiniDungeons 2", which is a turn-based rogue-like game. They also introduced an evaluation function for each persona to MCTS to make the AI behave according to the given persona. From their experimental results, they showed that their proposed AIs could play "MiniDungeons 2" according to the specified personas in terms of the AIs' action frequcies and differences in decision making. However, they did not quantitatively verify whether spectators could subjectively recognize and identify the AIs' personas.

B. Puppet-Master MCTS

PM-MCTS is a variant of MCTS, proposed by Ito *et al.* [6], that controls all characters in the game using only a single game tree. An open loop approach [9] is adopted in PM-MCTS. Figure 1 shows an overview of PM-MCTS applied to a two-player fighting game, such as FightingICE. In this tree, according to the open loop approach, each node represents an action choice for either of the character (circle: P1; square: P2) that can execute a new action. PM-MCTS builds such a tree starting from an initial state, defined by information such as the Hit-Point (HP), energy, coordinates, and action of each character and the game remaining time. Each edge represents the ongoing execution of two actions: one just started from the parent node and the other started earlier by the other character. Note that the branching factor at a given node in PM-MCTS is the number of actions of the respective character, not the combination of all possible actions from all characters. In addition, it is more straight-forward and efficient to build the opponent models for PM-MCTS than for multiple MCTs, one for each character.

PM-MCTS comprises five steps: selection, expansion, simulation, backpropagation, and decision making, the first four of which are the usual steps in MCTS algorithms. These five steps are described individually in the following subsections.

1) Selection: Nodes are selected from the root node until a leaf node is reached according to the given selection criterion. In our work, we use Upper Confidence Bounds ($UCB1$) value [10], which is widely used in this step of MCTS, defined by the following formula:

$$UCB1_i^{pl} = \overline{X}_i^{pl} + C\sqrt{\frac{2\ln N}{N_i}}, \tag{1}$$

where N_i is the number of times node (action) i was visited, N is the sum of N_i for node i and its sibling nodes, and C is a constant. $\overline{X}_i^{pl}$ defined in formula (2) is the average evaluation value of node i from the perspective of character pl, the one who can start executing an action at this node. It is worth reminding that every node of the tree contains the $UCB1$ values from the perspective of both characters, as well as a counter on how many times the node has been visited. When the AI selects a child node, it uses the $UCB1$ value of the character who can start the next action at its parent node.

$$\overline{X}_i^{pl} = \frac{1}{N_i}\sum_{j=1}^{N_i} Eval_j^{pl}, \tag{2}$$

where $Eval_j^{pl}$ is the reward value gained in the jth simulation from the perspective of character pl.

In this work, the AI selects the nodes with the highest $UCB1$ value from the root node until a leaf node is reached.

2) Expansion: After a leaf node is reached in the $Selection$ step, if the number of times the leaf node has been explored exceeds a threshold N_{max} and the depth of the tree is lower than a threshold D_{max}, all possible child nodes are created at once from the leaf node. If the root node is the only node in the tree, PM-MCTS creates all of its child nodes, ignoring above conditions. Each newly created child node represents the game state when either of the characters can start a new action after the character at the parent node has finished its action. Note that if both characters can start their action at a leaf node of interest, PM-MCTS creates child nodes for P1 first, and when this situation happens again it creates child nodes for P2, and this alternation is continued.

3) Simulation: A simulation is carried out for T_{sim} seconds, sequentially using all actions of both characters in the path from the root node to the current leaf node. If T_{sim} has not passed yet after those actions have been executed, a rollout will be carried out until T_{sim} runs out using randomly selected actions; in other words, T_{sim} limits the rollout depth in this case. Each character's $Eval_j$ is then calculated using formula (2).

4) Backpropagation: Each character's $Eval_j$ obtained in the $Simulation$ step is backpropagated from the leaf node to the root node. Each character's $UCB1$ value at each node along the path is updated as well.

5) Decision Making: PM-MCTS repeats the above four steps until the character who will be executing an action at the root node becomes available to do so ,i.e., the character's previous action has finished. The AI then chooses the character's action according to a given recommendation policy. In this work, it selects the edge (action) connected to the direct child node that has the highest $\overline{X}_i^{pl}$ from the root node as the next action. That child node will be used as the next root node and its sibling nodes will be pruned. This step is explicitly described here and shown in Fig. 1 to emphasize the reuse of tree structures, as often done in the open-loop approach.

III. Personas in Fighting Games

In this section, we give the definition of personas in fighting games. In this work, we define two fighting-game personas, i.e., *RushDown* and *Zoning*, and individually assign them a set of actions. Both action sets do not overlap and are fixed during gameplay. The details of them are described in the following subsections.

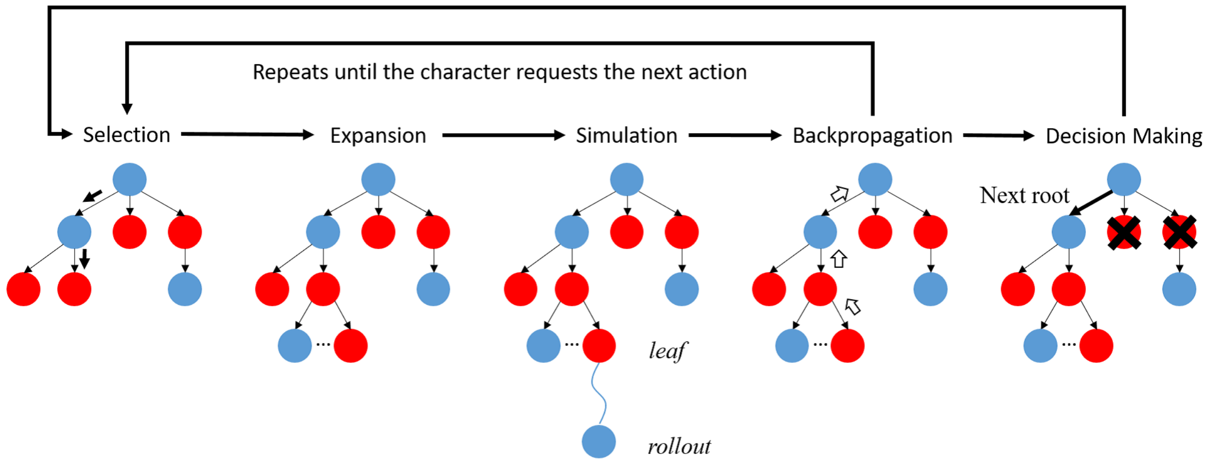

Fig. 1: An overview of PM-MCTS for a two-player game

A. *RushDown*

RushDown is a playstyle in fighting games where the player prefers to fight in close ranges. The behavioral tendency of *RushDown* is that the player often uses moving actions such as step forward to approach the opponent and close-range attacks such as punch or kick. In FightingICE, described in Section V, the actions shown in Table I are defined as actions belonging to *RushDown*.

B. *Zoning*

Zoning is a playstyle in fighting games where the player prefers to fight while keeping a certain distance with the opponent. The behavioral tendency of *Zoning* is that the player often uses moving actions such as back step to move away from the opponent, and long-range attacks such as fire-ball shoots. In FightingICE, the actions shown in Table II are defined as actions belonging to *Zoning*.

IV. Proposed Method

In this section, we describe the proposed method for automatically generating GPVs where each character behaves according to the given persona. PM-MCTS is used as a base method, but we introduce a new evaluation function and roulette selection-based simulation. The details of our proposed method are given in the following subsections.

A. *Evaluation function*

The proposed evaluation function for character pl in the jth simulation is defined as follows:

$$Eval_j^{pl} = ePersona_j^{pl} \times eHP_j^{pl}, \tag{3}$$

where $ePersona_j^{pl}$ is the evaluation value regarding how much the character's persona could be realized at a node of interest and is defined as

$$ePersona_j^{pl} = \frac{act_{Persona_j}^{pl} + dist_{Persona_j}^{pl}}{3} \tag{4}$$

TABLE I: All actions belonging to *RushDown*

Skill name	Skill content	Damage
STAND_A	Simple punch	5
STAND_B	Simple kick	10
CROUCH_A	Crouch punch	5
CROUCH_B	Crouch kick	10
STAND_D_DB_BA	Jumping punch	10
STAND_D_DB_BB	Sliding kick	20
FORWARD_WALK	Walk forward	0
DASH	Dash forward	0

TABLE II: All actions belonging to *Zoning*

Skill name	Skill content	Damage
STAND_D_DF_FA	Shoot projectile forward	10
STAND_D_DF_FB	Shoot strong projectile forward	40
THROW_A	Throw the opponent	10
THROW_B	Strongly throw the opponent	20
BACK_STEP	Step back	0
BACK_JUMP	Jump backward	0
FOR_JUMP	Jump forward	0

In the above formula, $act_{Persona_j}^{pl}$ is the term that evaluates whether character pl conducts an action belonging to the given persona, $Persona \in \{RushDown, Zoning\}$, in this simulation and is defined as

$$act_{Persona_j}^{pl} = \begin{cases} 1 & (belongs\ to\ Persona) \\ -1 & (otherwise) \end{cases} \tag{5}$$

If the character conducts an action belonging to $Persona$, it will obtain a positive evaluation; otherwise, a negative one.

The term $dist_{Persona_j}^{pl}$ considers the distance between both characters. In our work, according to our experience and preliminary experiments, we define $dist_{Persona_j}^{pl}$ when the persona of character pl is *RushDown* (formula (6)) and *Zoning* (formula (7)), respectively, as follows:

$$dist^{pl}_{RushDown_j} = \begin{cases} 2 & (distance < 130) \\ 2 - (\frac{distance}{width} \times 4) & (otherwise) \end{cases} \tag{6}$$

$$dist^{pl}_{zoning_j} = \begin{cases} -2 & (distance < 240) \\ 2 & (240 \leq distance < 450) \\ 2 - (\frac{distance}{width} \times 4) & (otherwise) \end{cases} \tag{7}$$

In formulas (6) and (7), $distance$ represents the distance between the characters, and $width$ is the number of pixels indicating the width of the game screen. When a character is within the distance range suitable for realization of the given persona, the highest evaluation value is obtained. In our work, we define the distance that a short-range attack hits the opponent as a suitable distance for *RushDown* while the distance that a long-range attack hits the opponent and that a character can avoid the opponent's close-range attacks as a suitable distance for *Zoning*.

In addition, in formula (3), eHP^{pl}_j is the term that evaluates how much the HP of the opponent of character pl has decreased and is defined as follows:

$$eHP^{pl}_j = oppHP_{root} - oppHP_{rollout} \tag{6}$$

where $oppHP_{root}$ and $oppHP_{rollout}$ stand for the HP of the opponent at the root node and after the rollout, respectively. If character pl gives a high amount of damage to the opponent, eHP will have a high value. The main role of this term is to keep the believability of character pl: if there was only $ePersona$ in our evaluation function, for example, a character might conduct unnatural actions such as repeating punch or dash only to realize *RushDown*, its persona.

The proposed evaluation function makes the character select actions by considering not only how to realize its persona but also always how to defeat the opponent, which is the main purpose of players in fighting games.

B. Roulette selection applied to rollout

The original PM-MCTS uses randomly selected actions for both characters in the rollout. However, if this was adopted in our work, actions not appropriate for each character's persona might be conducted. This may cause inaccurate evaluation of all nodes in the current path if the random-based rollout is still used.

In order to solve this issue, we introduce roulette selection, where the actions belonging to the persona of a character of interest have higher weights, to the rollout. The algorithm of roulette selection is shown in Algorithm 1. As the variables in this algorithm, $actionList$ is a list containing all actions, $actFit$ is an array containing all actions' fitness values, $dart$ is a threshold, and $totalFit$ is a sum of all actions' fitness values in $actFit$. The function initialize() initializes the fitness values of all actions passed to it as the argument $actionList$. In our work, the fitness value of each action is set to 3 if it belongs to the action set defined in the persona of a character of interest; otherwise, 1, not 0 in order to allow actions not belonging to this persona but with a strong damage value to be also chosen.

This mechanism makes a character of interest use actions that belong to its persona in a rollout more and hence increases the accuracy of the simulation.

Algorithm 1 Algorithm of roulette selection

```
// initializes all actions' fitness values
$actFit \leftarrow$ initialize($actionList$)
$dart \leftarrow$ random(0,1)
$wheel \leftarrow actFit[0]/totalFit$
$count \leftarrow 0$
while $dart > wheel$ do
    $count \leftarrow count + 1$
    $wheel \leftarrow wheel + actFit[count]/totalFit$
end while
//Returns the action selected by roulette selection
return $ActionList.get(count)$
```

Algorithm 2 Algorithm of the proposed method

```
$state \leftarrow$ getNowState() //Gets the current game state
$root \leftarrow$ Initialize($state$) //Initializes the root node
while $!isGameEnd$ do
    $activePlayer \leftarrow root.activePlayer$
    //Runs PM-MCTS until $activePlayer$ can execute its next action
    $bestAct \leftarrow$ TreeSearch($root$, $state$, $activePlayer$)
    runAction($bestAct$, $activePlayer$)
    $root \leftarrow$ nextRoot($root$, $bestAct$) //Changes the root node
    $state \leftarrow$ getNowState() //Gets the current game state
end while
```

C. Algorithm of the proposed method

Algorithm 2 shows the algorithm of the proposed method. As the variables in this algorithm, $state$ represents the current game state, $root$ is the root node, $activePlayer$ is the character who will be executing the next action in the game, and $bestAct$ is the selected action by PM-MCTS. The function getNowState() is for obtaining the current game state, initialize() is for initializing a node passed to it as the argument, TreeSearch() is for running PM-MCTS, runAction() is for conducting an action passed to it as the argument, and nextRoot() is for changing the root node.

V. Experiment

In this section, we describe the conducted experiment to verify the performance of the proposed method (P-AI).

A. FightingICE

FightingICE (Fig. 2) is a real-time 2D fighting game platform used in a game AI competition (FTGAIC)[1] at CIG since

[1]http://www.ice.ci.ritsumei.ac.jp/~ftgaic/
https://github.com/TeamFightingICE/FightingICE

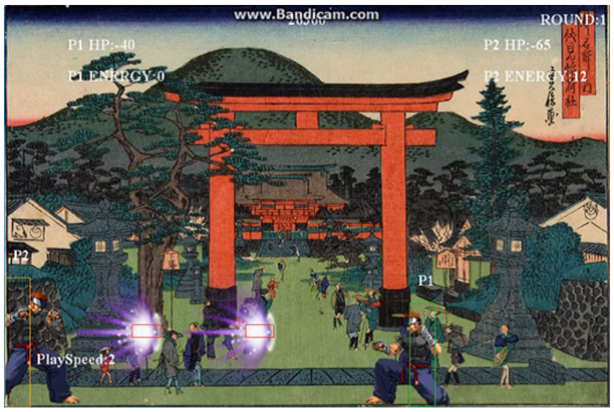

Fig. 2: Screenshot of FightingICE

2014 [7] and for research [6, 11-23]. Because FightingICE has been originally developed from scratch without using a ROM emulator and publicly made available, there are no legal issues to be concerned. In FightingICE, one game consists of three 60-second rounds, and one frame lasts 1/60 seconds. Each character has to decide and input an action in one frame. The HP for both characters is initially set to HP_{max} and it decreases when the corresponding character is hit. When the play is conducted for 60 seconds or either of the two characters' HP becomes 0, the game will proceed to the next round unless the current round is the 3rd one, after which each character's HP will be reset to HP_{max}. The character with the larger remaining HP at the end of a round is the round's winner. In our experiment, the value of HP_{max} is set to 400 according to the rule of Standard Track of FTGAIC.

One of the limitations, from our work's perspective, of FightingICE used in FTGAIC is that each character must be individually controlled. To remove this limitation, we modified some functions in FightingICE so that PPM-MCTS can control both characters. Another limitation is that the characters can only obtain each time a game state delayed by 15 frames (0.25 s), taking into account the delay of human perception. However, since our work is focused on the generation of GPVs, not on the development of a character for fighting against another AI opponent or a human opponent in competitions, we also removed the delay from the FightingICE platform in our experiment.

B. AIs and parameters in use

In the experiment, we compared the proposed AI (P-AI) and another MCTS-based AI (M-AI) that controls only one character [18] using the same evaluation function and roulette selection described in Section IV. The parameters used in both AIs are shown in Table III. These parameters were set empirically through pre-experiments.

C. Details

Our experiment consists of 35 participants (average age: 22.8 ± 2.6).

TABLE III: Parameters used in the experiment

Notation	Meaning	Value
C	Balancing parameter	0.025
N_{max}	Threshold of the number of visits	10
D_{max}	Threshold of the tree depth	10
T_{sim}	Simulation-time budget	60 frames
$width$	Width of the game screen	960 pixels

TABLE IV: Persona of each character in a fight

P1	P2
RushDown	RushDown
Zoning	Zoning
RushDown	Zoning
Zoning	RushDown

1) Generation of GPVs: A 60-second GPV for each combination of two personas shown in Table IV was generated by each AI, leading to eight GPVs[2] in total. For generation of GPVs, the P-AI controlled both characters. However, since an M-AI can control only one character, two M-AIs (2M-AIs), each assigned a persona accordingly, were used to generate a GPV; at the simulation step, the opponent's actions are selected by roulette selection using its persona.

2) Procedure: The procedure of our experiment is as follows:

1) Explain the concept of each of the two personas, *RushDown* and *Zoning*, and show sample GPVs to a participant.
2) Ask each participant to watch one of the generated GPVs.
3) Ask each participant to choose the persona for P1 and that for P2, among "RushDown", "Zoning", and "Other".
4) Repeat Steps 2) and 3) for all generated GPVs.

Note that at Step 2), a GPV was displayed in random order.

VI. Results and Discussions

In this section, we show the experimental results and our discussions in terms of whether the participants were able to identify each character's persona in GPVs. The summarization tables of the correctly answering participants and incorrectly answering participants for both characters (a), P1 (b), and P2 (c) in each GPV are shown in Tables V–IX, in a 2×2 contingency table style often used in the McNemar's test conducted below. In these tables, P, M, T, F, and S are P-AI, 2M-AIs, the number of participants who correctly answered, the number of people who incorrectly answered, and the sum of numbers in the corresponding row or column, respectively. Note that in each table (a), we counted the number of participants who correctly answered the personas of both characters as T; otherwise, F. We describe the result of each persona combination in the following subsections.

A. RushDown vs RushDown

Table V shows the summarization of the numbers of correct participants and incorrect participants for the *RushDown* vs

[2]http://www.ice.ci.ritsumei.ac.jp/~ruck/personaGPVs-cig2018.htm

RushDown GPVs generated by P-AI and 2M-AIs. In Tables Va and Vb, we can see that the participants could correctly answer the personas of both P1&P2 and P1 alone in the GPV generated by P-AI (33/35 or 94.3% in both cases) more than those in the one generated by 2M-AIs (P1&P2: 17/35 or 48.6%, P1: 18/35 or 51.4%). From the results of McNemar's tests, there are significant differences at 1% between P-AI and 2M-AIs in both cases. However, from Table Vc, both AIs obtain a high accuracy. From our observation, P2, controlled by either P-AI or 2M-AIs, aggressively shortened the distance between the characters even than P1 in the GPV generated by P-AI. Due to this behavior, P2's persona was judged as *RushDown* by most of the participants.

B. Zoning vs Zoning

Table VI shows the summarization of the numbers of correct participants and incorrect participants for the *Zoning* vs *Zoning* GPVs generated by P-AI and 2M-AIs. We can see that the participants could correctly answer the personas in the GPV generated by P-AI (P1&P2: 16/35 or 45.7%, P1: 31/35 or 88.6%, and P2: 20/35 or 57.1%) more than those in the one generated by 2M-AIs (P1&P2: 1/35 or 2.9%, P1: 4/35 or 11.4%, and P2: 12/35 or 34.3%). From the results of McNemar's tests, there are significant differences at 1% between P-AI and 2M-AIs for P1&P2 and P1, and at 10% for P2. However, there are fewer people who correctly answered both characters' personas than those who did not. This is due to the specification of attacks belonging to *Zoning*. All of the attacks in *Zoning* given in Table II need energy to conduct them, and this makes zoning characters select moving actions such as a jump forward. In addition, when both characters approach each other due to such moving actions, they often conduct actions that give damage to their opponent according to the term eHP in formula (6). Due to these behaviors, both characters' personas were judged as *RushDown* by a number of participants, especially in the GPV generated by 2M-AIs.

C. RushDown vs Zoning

Table VIII shows the numbers of correct participants and incorrect participants for the *RushDown* vs *Zoning* GPVs generated by P-AI and 2M-AIs. From these tables, we can see that the participants could correctly answer the personas in the GPV generated by P-AI more than those in the one generated by 2M-AIs for all cases, especially P1&P2 and P2. From the results of McNemar's tests, there are significant differences at 1% between P-AI and 2M-AIs for the three cases. This is because each M-AI individually decides actions according to its own evaluation function, without considering the opponent's next action, which is different from P-AI. This often caused mismatched fighting such as approaching each other, which looks like *RushDown* vs *RushDown*. Due to these behaviors, many participants answered wrong personas in the GPV generated by 2M-AIs.

D. Zoning vs RushDown

Table IX shows the summarization of the numbers of correct participants and incorrect participants for the *Zoning* vs *RushDown* GPVs generated by P-AI and 2M-AIs. In Tables Xa and Xb, we can see that the participants could correctly answer the personas of P1&P2 and P1 in the GPV generated by P-AI (P1&P2: 22/35 or 62.9%, P1: 31/35 or 88.6%) more than those in the one generated by 2M-AIs (P1&P2: 1/35 or 2.9%, P1: 2/35 or 5.7%). From the results of McNemar's tests, there is a significant difference at 1% between P-AI and 2M-AIs for each of the aforementioned two cases. However, from Table Xc, there are fewer correct participants for P2's persona in the GPV generated by P-AI than that by 2M-AIs, with a significant difference at 10%. From our observation, in the GPV generated by P-AI, P2 sometimes used projectile attacks belonging to *Zoning*. This is because such attacks, if used, can give more damage than simple close-range attacks belonging to *RushDown*, causing the term eHP for P-AI to obtain a much higher evaluation value than $ePersona$ and hence making rushdown characters behave like *Zoning* rather than *RushDown*.

TABLE V: The numbers of correct participants and incorrect participants for the *RushDown* vs *RushDown* GPVs generated by P-AI and 2M-AIs

(a) P1 & P2

M P	T	F	S
T	17	16	33
F	0	2	2
S	17	18	35

(b) P1

M P	T	F	S
T	18	15	33
F	0	2	2
S	18	17	35

(c) P2

M P	T	F	S
T	33	1	34
F	1	0	1
S	34	1	35

TABLE VI: The numbers of correct participants and incorrect participants for the *Zoning* vs *Zoning* GPVs generated by P-AI and 2M-AIs

(a) P1 & P2

M P	T	F	S
T	1	15	16
F	0	19	19
S	1	34	35

(b) P1

M P	T	F	S
T	4	27	31
F	0	4	4
S	4	31	35

(c) P2

M P	T	F	S
T	7	13	20
F	5	10	15
S	12	23	35

E. Summary of the results

In summary, we can conclude that the participants could better identify both characters' personas in the GPVs generated by P-AI than those in the GPVs generated by 2M-AIs. As

TABLE VIII: The numbers of correct participants and incorrect participants for the *RushDown* vs *Zoning* GPVs generated by P-AI and 2M-AIs

(a) P1 & P2

P \ M	T	F	S
T	4	28	32
F	1	2	3
S	5	30	35

(b) P1

P \ M	T	F	S
T	20	12	32
F	1	2	3
S	21	14	35

(c) P2

P \ M	T	F	S
T	8	25	33
F	1	1	2
S	9	26	35

TABLE IX: The numbers of correct participants and incorrect participants for the *Zoning* vs *RushDown* GPVs generated by P-AI and 2M-AIs

(a) P1 & P2

P \ M	T	F	S
T	1	21	22
F	0	13	13
S	1	34	35

(b) P1

P \ M	T	F	S
T	2	29	31
F	0	4	4
S	2	33	35

(c) P2

P \ M	T	F	S
T	21	3	24
F	10	1	11
S	31	4	35

mentioned earlier, P-AI controls both characters based on the information on their future actions. Due to this mechanism, P-AI can select the next action for each character that well expresses its persona.

VII. Conclusions and Future Work

Video games are now an attractive entertainment media not just for players but also spectators. To provide customized spectator-specific GPVs that match various kinds of spectators' preferences, AIs that can automatically generate GPVs with a variety of contents are needed. In this paper, we focused on personas, which represent playstyles, in fighting games. In order to generate GVPs where each character plays according to a given persona, we adopted a recently developed variant of MCTS called Puppet-Master MCTS, which controls all characters in the game, and introduced a new evaluation function and roulette selection-based simulation to this MCTS.

The results of the conducted experiment confirmed that the proposed AI can make both characters successfully behave according to their personas, which were identified by the participants or spectators in the experiment. However, the proposed AI still has an issue when controlling characters having the persona of *Zoning*. For future work, we plan to develop new evaluation functions and mechanisms to realize given personas more accurately. In addition, we plan to introduce more personas besides *RushDown* and *Zoning* for generating a higher variety of GPVs. It might also be interesting to consider believability [23] and human-play emulation [24] aspects or to analyze generated gameplay with action metrics recently proposed by Zook and Riedl [25]. We also plan to verify whether GPVs generated by our proposed AI can entertain spectators through extensive user studies.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments. They would also like to thank their lab members, in particular, the FightingICE team members for their fruitful discussions. This research was partially supported by Strategic Research Foundation Grant-aided Project for Private Universities (S1511026), Japan.

References

[1] T. Smith, M. Obrist and P. Wright, "Changes the (Video) Game," in *Proc. 11th European Conference on Interactive TV and Video (Live-Streaming)*, ACM, pp. 131-138, 2013.

[2] R. Thawonmas and T. Harada, "AI for Game Spectators: Rise of PPG," in *Proc. AAAI 2017 Workshop on What's next for AI in games*, San Francisco, USA, pp. 1032-1033, 2017.

[3] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Proc. International Conference on Computers and Games*, pp. 72-83, 2006.

[4] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1-43, 2012.

[5] A. Tychsen and A. Canossa, "Defining personas in games using metrics," in *Proc. 2008 Conference on Future Play: Research, Play, Share*, ACM, pp. 73-80, 2008.

[6] S. Ito, M. Ishihara, M. Tamassia, T. Harada, R. Thawonmas and F. Zambetta, "Procedural Play Generation According to Play Arcs Using Monte-Carlo Tree Search," in *Proc. 18th International Conference on Intelligent Games and Simulation*, pp. 67-71, 2017.

[7] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee and R. Thawonmas, "Fighting Game Artificial Intelligence Competition Platform," in *Proc. IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, pp. 320-323, 2013.

[8] C. Holmgård, A. Liapis, J. Togelius, and G. Yannakakis, "Monte-Carlo Tree Search for Persona Based Player Modeling," in *Proc. AIIDE workshop on Player Modeling*, 7 pages, 2015.

[9] D.P. Liebana, J. Dieskau, M. Hunermund, S. Mostaghim, S. Lucas, "Open Loop Search for General Video Game Playing," in *Proc. the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO' 15)*, pp. 337-344, 2015.

[10] P. Auer, N. Cesa-Bianchi, P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235-256, 2002.

[11] X. Neufeld, S. Mostaghim, and D. Perez-Liebana, "HTN fighter: Planning in a highly-dynamic game," in *Proc. 2017 Computer Science and Electronic Engineering (CEEC 2017)*, Colchester, pp. 189-194, Sep. 2017.

[12] S. Demediuk, M. Tamassia, Wi. Raffe, F. Zambetta, X. Li and F.F. Mueller, "Monte Carlo Tree Search Based Algorithms for Dynamic Difficulty Adjustment," in *Proc. 2017 IEEE Conference on Computational Intelligence and Games (CIG 2017)*, 2017.

[13] S. Yoon and K.-J. Kim, "Deep Q Networks for Visual Fighting Game AI," in *Proc. 2017 IEEE Conference on Computational Intelligence and Games (CIG 2017)*, 2017.

[14] M.-J. Kim and K.-J. Kim, "Opponent Modeling based on Action Table for MCTS-based Fighting Game AI," in *Proc. 2017 IEEE Conference on Computational Intelligence and Games (CIG 2017)*, 2017.

[15] D.T.T Nguyen, V. Quang and K. Ikeda, “Optimized Non-visual Information for Deep Neural Network in Fighting Game,” in *Proc. 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pp. 676-680, 2017.

[16] M. Ishihara, T. Miyazaki, Y. Chu, T. Harada, R. Thawonmas, “Applying and Improving Monte-Carlo Tree Search in a Fighting Game AI,” in *Proc. 13th International Conference on Advances in Computer Entertainment Technology*, ACM, no. 27, 2016.

[17] T. Kristo, N.U. Maulidevi, “Deduction of fighting game countermeasures using Neuroevolution of Augmenting Topologies,” in *Proc. 2016 International Conference on Data and Software Engineering (ICoDSE)*, 2016. DOI:10.1109/ICODSE.2016.7936127

[18] S. Yoshida, M. Ishihara, T. Miyazaki, Y. Nakagawa, T. Harada and R. Thawonmas, “Application of Monte-Carlo Tree Search in a Fighting Game AI,” in *Proc. IEEE 5th Global Conference on Consumer Electronics (GCCE)*, pp. 623-624, 2016.

[19] K. Majchrzak, J. Quadflieg, and G. Rudolph, “Advanced Dynamic Scripting for Fighting Game AI,” in *Proc. Entertainment Computing (ICEC 2015)*, pp. 86-99, 2015.

[20] K. Asayama, K. Moriyama, K. Fukui, and M. Numao, “Prediction as Faster Perception in a Real-time Fighting Video Game,” in *Proc. 2015 IEEE Conference on Computational Intelligence and Games (CIG 2015)*, pp. 517-522, 2015.

[21] N. Sato, S. Temsiririkkul, S. Sone. and K. Ikeda, “Adaptive Fighting Game Computer Player by Switching Multiple Rule-based Controllers,” in *Proc. 3rd International Conference on Applied Computing and Information Technology (ACIT 2015)*, pp. 52-59, 2015.

[22] H. Park and K.J. Kim, “Learning to Play Fighting Game using Massive Play Data,” in *Proc. 2014 IEEE Conference on Computational Intelligence and Games (CIG 2014)*, pp. 458-459, 2014.

[23] M. Ishihara, S. Ito, R. Ishii, T. Harada and R. Thawonmas, “Monte-Carlo Tree Search for Implementation of Dynamic Difficulty Adjustment Fighting Game AIs Having Believable Behaviors,” in *Proc. 2018 IEEE Conference on Computational Intelligence and Games (CIG 2018)*, 2018.

[24] S. Devlin, A. Anspoka, N. Sephton, P.I. Cowling, “Combining Gameplay Data with Monte Carlo Tree Search to Emulate Human Play,” in *Proc. Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2016)*, pp. 16-22, 2016.

[25] A. Zook and M.O. Riedl, “Learning How Design Choices Impact Gameplay Behavior,” *IEEE Transactions on Games*, Mar, 2018. (Early Access)