applied sciences

MDPI

Article

Improved Belgian AI Algorithm for Dynamic Management in Action Role-Playing Games

Qingwei Mi and Tianhan Gao *

Software College, Northeastern University, Shenyang 110169, China
* Correspondence: gaoth@mail.neu.edu.cn

Abstract: Artificial intelligence in games is one of the most challenging tasks in academia and industry. In action role-playing games, how to manage combat effectively is a key issue related to game development and the player's experience. The Belgian artificial intelligence (BAI) algorithm is a classic but limited method that is widely used for combat management between the player and enemies. To address the poor adaptability of BAI, this paper proposes an improved Belgian artificial intelligence (IBAI) algorithm with dynamic difficulty adjustment (DDA) and implements two systems separately based on BAI and IBAI in Unreal Engine 4. Advantages on 12 parameters—10.086 mean total score greater, and 0.079 standard deviation smaller—demonstrate that the system based on IBAI has higher adaptability and a better player experience by comparing the two systems in different situations and inviting players to participate in gameplay experiences and questionnaires. The robust dynamic management mechanism of IBAI can help game designers and developers achieve the combat system of action role-playing games more efficiently, thus, shortening the development cycle and improving the player retention rate.

Keywords: action role-playing games; game artificial intelligence; dynamic difficulty adjustment; combat management

check for updates

Citation: Mi, Q.; Gao, T. Improved Belgian AI Algorithm for Dynamic Management in Action Role-Playing Games. *Appl. Sci.* **2022**, *12*, 11860. https://doi.org/10.3390/app122211860

Academic Editors: Davide Gadia, Dario Maggiorini and Laura Anna Ripamonti

Received: 13 September 2022
Accepted: 18 November 2022
Published: 21 November 2022

1. Introduction

Artificial intelligence is developing by leaps and bounds and has cut a figure in academia and industry. In general research, AI is highly technical and professional to lay the foundation of every walk of life [1–3]. In the field of games, artificial intelligence R and D is challenging work that requires game designers and developers to have creative thinking as well as solid insight [4,5]. Game AI aims mainly to maximize the fun of the player while enhancing the game challenge and supporting an extraordinary gaming experience [6–8]. The player experience refers to the total effect of feelings, thoughts, emotions, and behaviors exerted on the player during the gameplay, including immersive experience, game participation experience, demand satisfaction experience, etc. [9,10]. The player experience is not an attribute of the game, but the state in which the player interacts with the game, which directly determines the sales volume, praise, player retention rate, and other key indicators [11–13]. Dynamic difficulty adjustment (DDA) is a typical technology that can maintain the player's interest to ensure their gaming experience by modifying game parameters, behaviors, and scenes in real-time according to the player's skill level automatically [14–17].

Role-playing games are one of the oldest and is also the most popular game genre [18]. In recent years, research focusing on the design of role-playing games has become a hotspot, including procedural generation [19–21], pathfinding systems [22], narrative systems [23], weapon systems [24], drop systems [25], etc. However, research on DDA and combat management system is relatively rare. Action role-playing games are a representative branch of role-playing games [26], where the game challenge is mainly derived from the combat interaction between the enemy and the player [27]. The management of combat will affect the development progress, cost, and difficulty of the game directly, thus, affecting the player experience [28].

The kung-fu circle is a classic combat management algorithm, where the enemy is able to attack the player [29]. However, kung-fu circle is inappropriate for fast-paced combat, and it does not take into account the relationship between enemy types and attack types for multiple enemies.

"Kingdoms of Amalur: Reckoning" is an excellent action role-playing game. The combat management system of this game was designed and implemented based on the Belgian artificial intelligence (BAI) algorithm [29]. BAI uses the same rules-based content for each decision, which is more complete and adaptive than Kung-Fu Circle. From the top level, BAI is based on rectangular grids, and only one enemy is allowed in each grid. There are glaring differences in the interaction area between enemies and the player in different grids, where a few attack types can be managed. Meanwhile, the decision mechanism of BAI may cause problems when the number or position of the enemy changes. In addition, the combat diversity is also insufficient, which leads to a decline in game balance.

In order to address the problems above, this paper proposes an improved Belgian artificial intelligence (IBAI) algorithm, whose main contributions are as follows.

1. The melee-ranged buffer-pursuit (MRBP) ring model with DDA introduced, the grid priority (GP), and real-time induction and distribution (RID) algorithm, as well as attack weight threshold (AWT) and global cooldown (GCD) are suggested to improve the scope and imperfections of BAI.
2. Two combat management systems based on BAI and IBAI are implemented in Unreal Engine 4. By testing the adaptability of the two systems in different situations and inviting players to participate in gameplay experiences and questionnaires, the results demonstrate that IBAI owns higher adaptability and player experience than BAI.
3. IBAI is able to help game designers and developers achieve combat management systems for action role-playing games with high work efficiency.

This work combines quantitative and qualitative research methodologies. The remainder of the paper is organized as follows: In Section 2, the authors adopt literature research and case studies to find and sketch the flaws of BAI. Section 3 elaborates on the proposed algorithm (IBAI). Questionnaire and statistical methods are used, and the adaptability as well as player experience of BAI and IBAI are analyzed by comparing and gameplay testing in Section 4. Finally, the contributions, implications, limitations, and future work of the paper are concluded in Section 5.

2. Belgian Artificial Intelligence Algorithm

BAI is designed based on rectangular grids around the player. It divides the game scene into player-centric and uniformly aligned rectangular grids. The eight grids around the player are utilized to hold the attacking enemies; each grid can hold only one enemy.

In action role-playing games, BAI sets four variables for the player, namely maximum grid capacity (MGC), current grid capacity (CGC), maximum attack capacity (MAC), and current attack capacity (CAC). MGC limits the number of enemies that can attack the player at the same time, while MAC determines the attack types. The initial CGC and CAC of the player are equal to the MGC and MAC, respectively. BAI sets grid weight (GW) and attack priority (AP) for each enemy, and attack weight (AW) for each attack type. GW represents the grid capacity value that the enemy occupies in the grid, while AW represents the attack capacity value of the enemy attack type. Each enemy has a normal attack, and its AW is zero.

BAI presents a specialized AI called a stage manager, which provides maintenance for enemy positioning in combat. The sum of GWs of enemies attacking the player simultaneously must be less than or equal to MGC, and the sum of AWs must be less than or equal to MAC. That is, the CGC and CAC of the player are not allowed to drop below zero. The execution process of BAI is shown as Algorithm 1.

Algorithm 1 BAI

```
Require:
the player's $MGC$, $CGC$, $MAC$, $CAC$, the enemy's $GW$, $AP$, each attack type's $AW$
Ensure:
behavior logic, process logic
$n_t \leftarrow$ the number of enemies that have gained the attack right
$n_f \leftarrow$ the number of enemies waiting outside the grids
$n_a \leftarrow$ the number of attack types of the enemy
$x \leftarrow$ the distance between the enemy and the player
$n_0 \leftarrow n_t$
$flag \leftarrow 0$
if an enemy registers a request to attack the player to the stage manager then
    CompareGrid: if the enemy's $GW \leq$ the player's $CGC$ then
        stage manager agrees to the request
        stage manager assigns the enemy the closest available grid position
        the enemy gains the attack right to the player
        the player's $CGC$ = the player's $CGC$—the enemy's $GW$
        $n_t \leftarrow n_t + 1$
        if $flag = 1$ then
            $n_f \leftarrow n_f - 1$
    else
        stage manager rejects the request
        the enemy waits outside the grids
        if $flag = 0$ then
            $n_f \leftarrow n_f + 1$
        else
            goto ContinueLoop
if an enemy that has gained the attack right dies then
    the player's $CGC$ = the player's $CGC$ + the enemy's $GW$
    $n_t \leftarrow n_t - 1$
if $n_t > n_0$ then
    the player's $CAC \leftarrow$ the player's $MAC$
    Sort all enemies that have gained the attack right from 0 to $n_t$–1 according to the priority
  of $AP$ in descending order and $x$ in ascending order
    for $i \leftarrow 0$ to $n_t - 1$ do
        Sort all attack types of the enemy from 0 to $n_a$–1 according to the priority of $AW$ in
  descending order and the game preset order
        for $j \leftarrow 0$ to $n_a - 1$ do
            the enemy registers a request to attack with the attack type to the stage manager
            if this attack type's $AW \leq$ the player's $CAC$ then
                stage manager agrees to the request
                the player's $CAC$ = the player's $CAC$—this attack type's $AW$
                break
            else
                stage manager rejects the request
        end for
    end for
    if $flag = 1$ then
        goto ContinueLoop
elseif $n_t < n_0$ then
    the player's $CAC \leftarrow$ the player's $MAC$
    Sort all enemies waiting outside the grids from 0 to $n_f - 1$ according to the previous
  registration order
    for $k \leftarrow 0$ to $n_f - 1$ do
        $flag \leftarrow 1$
        goto CompareGrid
        ContinueLoop: $flag \leftarrow 0$
    end for
return behavior logic, process logic
```

A combat example (shown in Figure 1) is provided to illustrate the interaction process between variables and the stage manager in BAI. In this example, MGC and MAC of the player are both 10, and all enemies use melee attacks only.

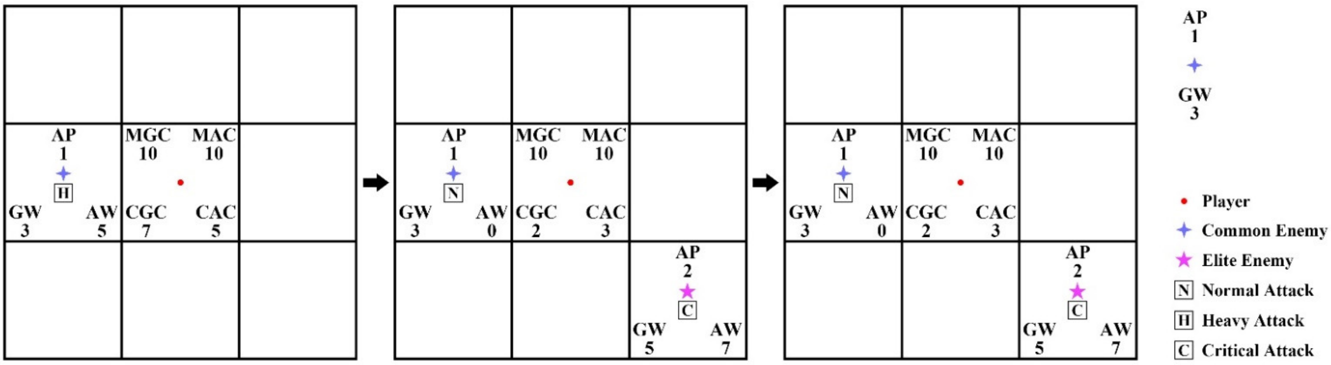

Figure 1. Combat example.

The player enters the attack decision area of a common enemy, which has a normal attack with an AW of zero and a heavy attack with an AW of five. The enemy will prioritize the heavy attack to attack the player as assigned by the stage manager. Later, an elite enemy joins the combat. It has two attack types, normal attack and critical attack, with an AW of zero and seven, respectively. In this case, the elite enemy will attack with a critical attack while the common enemy will attack with a normal attack after the reprocessing of the stage manager. Even if the player later enters the attack decision area of other enemies, the requests of the enemies will be rejected. Therefore, the enemies cannot enter the eight grids to attack the player and must wait outside.

BAI has a sounder mechanism and better adaptability compared with kung-fu circle, but it still has much room for improvement. According to the detailed analysis with combined research methodologies, the flaws of BAI can be concluded as below:

1. BAI holds the enemy in the form of eight rectangular grids, each of which can hold only one enemy and cannot be adjusted adaptively according to the specific game mechanism. Additionally, the interaction area between enemies and the player in the four grids at a 45-degree angle to the player is significantly smaller than the ones at a 90-degree angle. Moreover, BAI can only manage melee attacks rather than ranged attacks.
2. When multiple enemies register requests to attack the player simultaneously, the stage manager will reject them even if the player's CGC is large enough and the sum of these enemies' GW is greater than the player's CGC. Additionally, when the number or position of enemies that have gained the attack right changes, the position where enemies that have gained the attack right before the change may not be the closest attack position after the change, or the enemy that is not assigned has already been in the grids after the change, but its position conflicts with the position where enemies that have gained the attack right before the change. The enemy may possibly not gain the attack right.
3. In case the enemy is authorized by the stage manager to attack the player, it will use an attack type to attack constantly until the number of enemies that have gained the attack right changes. Furthermore, it is impossible to prevent the same type of enemies from attacking the player with the same attack type simultaneously, which will reduce combat diversity and game balance.

3. Improved Belgian Artificial Intelligence Algorithm

Based on the definition of variables and the stage manager in BAI, IBAI is proposed to include three components, melee-ranged buffer-pursuit (MRBP) ring model with DDA introduced, grid priority (GP), and real-time induction and distribution (RID) algorithm, as well as attack weight threshold (AWT) and global cooldown (GCD).

3.1. MRBP Ring Model

In action role-playing games, enemies usually own some common behaviors and attack rules. It means that enemies using melee attacks and ranged attacks can share the common behavior set of ones with the same type. This design pattern adds behavior reuse to some extent [30], thus, the MRBP ring model addresses the problems of BAI.

Compared with BAI, which uses rectangular grids to hold enemies, the MRBP ring model is a player-centric four-ring one (shown in Figure 2). The rectangular plane coordinate system is created with the current position of the player, where the directions of the x-axis and the y-axis are determined according to the game world, which does not change with the orientation of the player. Four circles with the radius of a, b, c, and d ($a < b < c < d$) are created with the player at the center, melee circle, ranged circle, buffer circle, and pursuit circle, from small to large. The four ring areas, from the inside out, are the melee area, ranged area, buffer area, and pursuit area. The outer area refers to the area outside the pursuit circle. The authors rotate the y-axis clockwise by 22.5 degrees, 67.5 degrees, 112.5 degrees, and 157.5 degrees, respectively, to form four straight lines along with four-line segments. The melee area, ranged area, and buffer area are then equally divided into eight sectors.

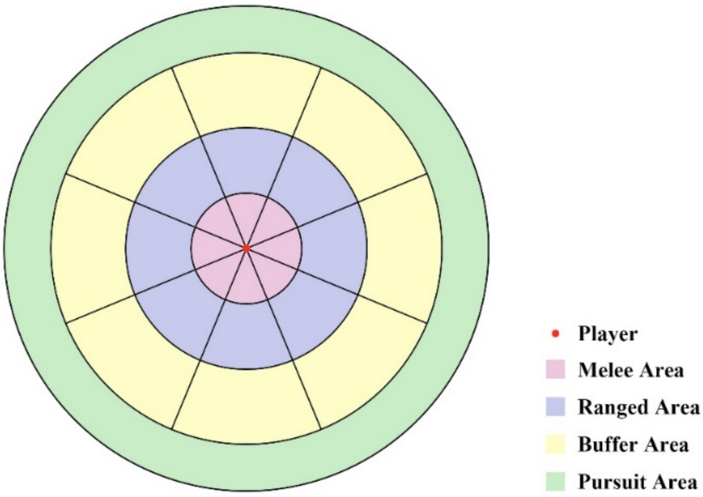

Figure 2. MRBP Ring Model.

In the model, the melee area and the ranged area are utilized to hold enemies using melee attacks or ranged attacks that have gained the attack right, respectively. The buffer area is used to hold enemies that have not gained the attack right, while the pursuit area is for the ones that are pursuing the player. The radii a, b, and c of the melee circle, ranged circle, and buffer circle are fixed variables, and d of the pursuit circle is dynamic. The number of enemies that can be held in each sector is proportional to the ring area in which it is located. Since enemies in the pursuit area are moving constantly, it is unnecessary to limit the number of enemies it can hold. The number of enemies that can be held in each sector of the melee area is set to one as the interaction area between enemies using melee attacks and the player becomes smaller. Meanwhile, the rounding up method is adopted to optimize and avoid the occurrence of zero or decimals in the ratio. Hence, the final ratio of the number of enemies that can be held in each sector of the melee area, ranged area, and buffer area is shown in (1).

$$1 : \frac{b^2 - a^2}{a^2} : \frac{c^2 - b^2}{a^2} \tag{1}$$

As for the DDA design, the greater the value of MGC and MAC, the more difficult the combat the player meets in the game. The execution logic of the DDA scheme is determined as follows on the basis of this feature and the sound management mechanism of the MRBP ring model.

1. Decrease MGC or MAC when the player is dying.
2. Increase MGC or MAC when the player has not been damaged for a long time.
3. Reset MGC and MAC to the player's initial value in the current combat after the player completes it.

The MRBP ring model is able to manage both melee attacks and ranged attacks. The radius of each circle in the model can be set adaptively according to the specific game mechanism, so as to adjust the number of enemies that can be held in each sector area. The interaction area between enemies and the player in each sector of the same ring area is consistent. On the DDA mechanism, the difficulty of the current combat can be adjusted effectively without changing the overall preset difficulty of the game by updating the player's MGC and MAC in real-time, so as to achieve the expected effect of DDA.

3.2. GP and RID Algorithm

IBAI sets the GP variable for each enemy and strengthens the control of the stage manager through the RID algorithm.

3.2.1. GP

When multiple enemies register requests to attack the player with the stage manager simultaneously, the higher the GP, the more preferred the enemy assignment. The enemy which is closer to the player will be assigned first if GPs of the enemies are equal.

The initial GP of each enemy is zero. When the number of enemies or the ring areas in which they are located in the current combat change except for exiting the combat, the GPs of enemies are set as shown in Table 1 according to the relationship between the player position and the casting range of the attack type with the smallest casting range of the enemy. GP will be reset to zero after the stage manager processes all requests.

Table 1. GP of enemies that have registered requests to attack after the change.

Player Position	Melee Attacks	Ranged Attacks
Within the range	3	2
Beyond the range	1	1

3.2.2. RID Algorithm

The RID algorithm employs the grid capacity overload mechanism (GCOM) for each enemy, which is disabled by default. When an enemy registers a request to attack the player, the stage manager will process the request according to the rule, which is that the sum of the GWs of enemies attacking the player at the same time must be less than or equal to the MGC of the player while GCOM is disabled. If the player's CGC is less than zero, the stage manager will keep the behaviors of enemies that have gained the right to attack the player unchanged and reject the enemy's request. The restriction will be lifted until the CGC is greater than or equal to zero. If GCOM is allowed, the stage manager will allow the sum of GWs of enemies attacking the player at the same time to exceed MGC.

IBAI sets an independent ring width *l* of the pursuit area for each enemy in the current combat, with the initial value of l_0 equaling *d*–*c*. Define *t* with the initial value of zero and the maximum value of *p* as the time elapsed since the enemy started pursuing the player. *L′* and *t′* are temporary variables to store the values of *l* and *t*, respectively, during state switching. The stage manager records the current position of the enemy in each frame, calculates the distance *s* between the enemy and the player, and detects *t*. The real-time ring width *l* of the pursuit area for each enemy is calculated as (2) according to the *s* values *s*1 of the current frame and *s*0 of the previous frame.

$$l = \begin{cases} l' + \frac{1}{p}\left(t - t'\right) * l_0,\ s_1 < s_0 \\ l',\ s_1 = s_0 \\ l' - \frac{1}{p}\left(t - t'\right) * l_0,\ s_1 > s_0 \end{cases} \tag{2}$$

Equation (2) demonstrates that the enemy is approaching if $s_1 < s_0$ while it is pursuing the player, and the pursuit area needs to be enlarged to adapt to the trend. Likewise, the pursuit area needs to be reduced if $s_1 > s_0$ for the enemy is moving away from the player. When $s_1 = s_0$, *l* will remain unchanged as the trend cannot be determined. The current values of *l* and *t* should be assigned to *l′* and *t′*, respectively, first while switching the three states above. If the enemy still does not enter the buffer circle as *t* = *p*, it will stop pursuing immediately and exit the current combat. Meanwhile, *l* and *t* will be reset to their initial values.

When the number of enemies or the ring areas where they are located in the current combat change except for exiting the combat, CGC and CAC will be reset to MGC and MAC, respectively. At the same time, all enemies in the combat before the change will stop their behaviors immediately, except for pursuing the player or exiting the current combat. The execution process of the RID algorithm is shown as Algorithm 2, while its execution logic is shown in Table 2 according to whether the enemies in each area after the change were in the current combat before the change.

Algorithm 2 IBAI-RID Algorithm-Change Process

```
Require:
the enemy's state before the change
Ensure:
behavior logic
if the enemy is not in the current combat before the change then
    if the player is in the attack decision area of the enemy or snaps the enemy's aggro then
        the enemy joins the current combat
        execute the behavior logic according to the ring areas in which the enemy is located currently
else
    if the position of the enemy changes except for the change caused by exiting the combat or the enemy dies then
        execute the behavior logic according to the ring areas in which the enemy is located before and after the change
return behavior logic
```

Table 2. Execution logic of RID algorithm.

Position Before the Change	Melee Area	Ranged Area	Buffer Area	Pursuit Area	Outer Area
Not in combat	Case A [1]	Case A	Case B [2]	Case C [3]	Case D [4]
Melee Area (in combat)	Case A	Case A	Case B	Case C	Case E [5]
Ranged Area (in combat)	Case A	Case A	Case B	Case C	Case E
Buffer Area (in combat)	Case A	Case A	Case B	Case C	Case E
Pursuit Area (in combat)	Case A	Case A	Case B	Case F [6]	Case E

[1] The enemy registers a request to attack the player and GCOM will be allowed. If $t \neq 0$, l and t will be reset to their initial values. [2] The enemy registers a request to attack the player. If $t \neq 0$, l and t will be reset to their initial values [3] The enemy registers a request to attack the player. [4] The enemy will not join the current combat for it cannot snap the player's aggro. [5] The enemy stops pursuing the player immediately and exits the current combat. If $t \neq 0$, l and t will be reset to their initial values. [6] The enemy continues pursuing the player.

As enemies complete executing the corresponding behavior logic, the stage manager will process the requests according to the GP and judge in terms of the state of GCOM. The logic of the stage manager processing requests to attack the player is shown in Algorithm 3.

Algorithm 3 IBAI-RID Algorithm-Attack Requests Process

Require:
the player's *MGC*, *CGC*, the enemy's *GW*, *GP*, *GCOM*
Ensure:
process logic

```
$flag_a, flag_b \leftarrow 0$
if the enemy's $GCOM$ is disabled then
    if the enemy's $GW \leq$ the player's $CGC$ then
        if any sector is not full in the corresponding ring area then
            $flag_a \leftarrow 1$
    if $flag_a$ = 1 then
        AgreeRequest: stage manager agrees to the request
        stage manager assigns the enemy the available position closest to its current position in the sector area which is not full in the corresponding ring area
        the enemy gains the attack right to the player
        the player's $CGC \leftarrow$ the player's $CGC$—the enemy's $GW$
        if $flag_b$ = 1 then
            goto SetMechanism
    else
        RejectRequest: stage manager rejects the request
        if any sector is not full in the Buffer area then
            stage manager assigns the enemy the available position closest to its current position in the sector area which is not full in the Buffer Area
            the enemy waits in place after arriving
        else
            the enemy exits the current combat
        if $flag_b$ = 1 then
            goto SetMechanism
else
        $flag_b \leftarrow 1$
        if any sector is not full in the corresponding ring area then
            goto AgreeRequest
    else
            goto RejectRequest
    SetMechanism: set the enemy's $GCOM$ disabled
return process logic
```

When the stage manager processes all requests to attack the player, enemies that have gained the attack right will execute the attack type selection process which is consistent with BAI. The stage manager will then process the requests according to GP and judge on the basis of the state of the sectors in the buffer area. The logic of the stage manager processing requests to pursue the player is shown in Algorithm 4.

Algorithm 4 IBAI-RID Algorithm-Pursue Requests Process

```
Require:
the enemy's $GP$, the sectors' state in the Buffer Area
Ensure:
process logic
if any sector is not full in the Buffer Area then
    the stage manager agrees to the request
    the enemy starts pursuing the player
else
    the enemy exit the current combat
return process logic
```

After all the above processes, the RID algorithm ends. All enemies in the current combat after the change will execute their behaviors according to the complete logic.

The GP sets the order for the stage manager to process requests when multiple enemies register requests to attack the player simultaneously. The RID algorithm enhances the control of the stage manager. Its execution process covers all situations when the number of enemies or the ring areas where they are located in the current combat change, except for the change caused by exiting the combat.

3.3. *AWT and GCD*

IBAI sets the AWT variable for each enemy and the GCD variable for each attack type of the enemy except normal attack.

3.3.1. AWT

Each enemy can only attack with an AW less than or equal to AWT. When an attack type is in cooldown, the enemy can attack with another attack type that meets the requirement of AWT. The purpose of setting AWT is to prevent the monopoly of the attack right of the same attack type.

The initial AWT of each enemy is zero, and the AWT will be set to the AW value of the attack type after the enemy is authorized by the stage manager. Additionally, AWT will be reset to zero if the enemy loses the attack right to the player.

3.3.2. GCD

Define each attack type of the enemy except normal attacks as a skill. When a skill is in GCD, all the same types of enemies cannot use this skill to attack. The vast majority of action role-playing games have more types of enemies and attack types [31]. The GCD aims to reduce the same type of enemies that attack the player with the same attack type simultaneously.

The casting time of the skill T refers to the time taken by the skill to be cast from the beginning to the end. The power of the skill P is calculated based on the weighted sum of the attributes. T, P, and the number of enemies in global combat are the essential factors affecting the length of GCD, which is calculated by (3).

$$GCD = n \ast T + P + m = n{\ast}(T_1{+}T_2){+}w_0 \ast (B + w_1{\ast}D + w_2 \ast A{+}w_3{\ast}R + w_4{\ast}E){+}m \tag{3}$$

In Equation (3), n is the number of enemies that can cast this skill in the current combat, and m is a constant. The GCD of the skill can be calculated according to the casting time when n enemies cast the same skill alternately, the power of the skill, and m. Among the variables, T is composed of the time before the skill roll T_1 and the time after the skill roll T_2, which refer to the time from emission to attack completion and the time from attack completion to the end of the skill action, respectively. B, D, A, R, and E are the scores of the five attributes of the enemy's level, damage, damage area, casting range, and effect. All scores are proportional to the strength of the corresponding attributes. As the effect score of the skill increases, the control, buff or debuff, overlay, interaction with the combat scene, effect triggered by special conditions, etc., will increase E accordingly. E is 0 if the skill has no effect. As for weights, w_0 is the global weight, and w_1, w_2, w_3, and w_4 are the local weights of D, A, R, and E, respectively.

When calculating T for enemies using melee attacks, the time from attack completion to the skill taking effect is extremely short and can be ignored. However, it cannot be ignored for enemies using ranged attacks, and its effect will be reflected by R so that it will not be calculated cumulatively. Moreover, P can provide a reference for setting AW of skills. That is, P and AW have a positive correlation.

In IBAI, AWT and GCD ensure combat diversity and game balance, thus enhancing the player experience.

4. Results and Discussion

In order to verify the improved effect of IBAI, two combat dynamic management systems were implemented in Unreal Engine 4 based on BAI and IBAI, respectively. The specific process and results of the design and implementation, testing, and comparison of the adaptability of the two systems are elaborated on in this section.

4.1. System Design and Implementation

Each system contains one player, multiple enemies, a stage manager, and other required assets, which are tested by a series of combats between the player and enemies at different test levels. The specific implementation process according to the relationships between the core classes in the system is as follows:

1. Create the player class and set related variables, functions, events, and interfaces.
2. Create the enemy class and complete the creation of EnemyMelee, EnemyRanged, and other classes based on the inheritance relationships. The system has six types of enemies, including three that use melee attacks named EnemyMelee1, EnemyMelee2, and EnemyMelee3, as well as the rest named EnemyRanged1, EnemyRanged2, and EnemyRanged3 that attack from a range. Among them, Skill1 of EnemyMelee2 is the same as EnemyMelee3, Skill1 of EnemyRanged2, and EnemyRanged3 are equivalent as well. The related variables, functions, events, and interfaces are then set.
3. Create the stage manager class and set its and all other related variables, functions, events, and interfaces.
4. Create meshes, models, weapons, animations, skill effects, and sound effects for the player and enemies.
5. Build test levels.

The settings of essential variables in the player class and each enemy class are shown in Table 3. In order to ensure the clarity of the screenshots during gameplay, the test levels for BAI and IBAI are designed with different styles of terrain. Meanwhile, the player′s head-up display information is hidden in the screenshots. The display of the player′s current health value will not affect the process or results of the tests. The screenshots of the test levels for BAI and IBAI are shown in Figure 3a,b, respectively.

Table 3. Essential variable values in the player class and each enemy class.

Variables	**Player**	**Enemy Melee1**	**Enemy Melee2**	**Enemy Melee3**	**Enemy Ranged1**	**Enemy Ranged2**	**Enemy Ranged3**
Pawn color	Grey	Red	Orange	Green	Cyan	Blue	Purple
MGC	10	-	-	-	-	-	-
MAC	10	-	-	-	-	-	-
GW	-	1	2	3	1	2	3
AP	-	1	3	5	2	4	6
Maximum health point	20	3	4	5	3	4	5
Number of skills	0	0	1	2	0	1	2
Normal attack′s AW	0	0	0	0	0	0	0
Normal attack′s damage performed	1	1	1	1	1	1	1
Normal attack′s cooldown	2.00	2.00	2.00	2.00	2.00	2.00	2.00
Skill1′s AW	-	-	2	2	-	2	2
Skill1′s damage performed	-	-	2	2	-	3	3
Skill1′s cooldown	-	-	2.00	2.00	-	2.00	2.00
Skill1′s T	-	-	1.10	1.10	-	1.33	1.33
Skill1′s P	-	-	0.50	0.50	-	0.90	0.90
Skill1′s m	-	-	0.10	0.10	-	0.10	0.10
Skill2′s AW	-	-	-	4	-	-	4
Skill2′s damage performed	-	-	-	3	-	-	1–12
Skill2′s cooldown	-	-	-	2.00	-	-	2.00
Skill2′s T	-	-	-	1.13	-	-	1.34
Skill2′s P	-	-	-	0.75	-	-	1.15
Skill2′s m	-	-	-	0.10	-	-	0.10

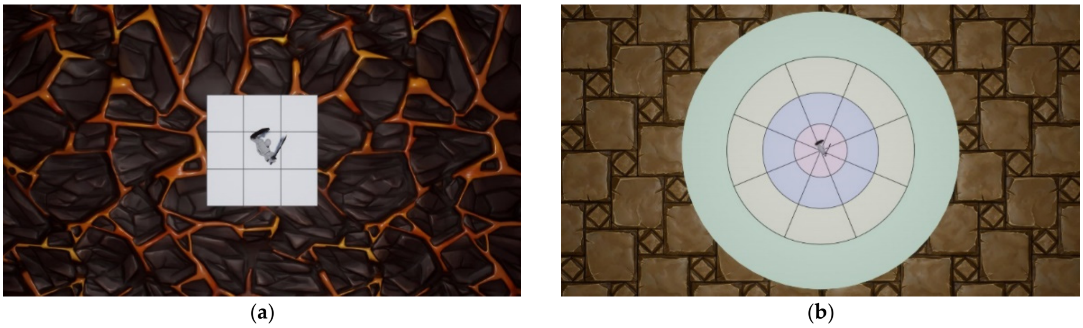

Figure 3. Test levels. (**a**) BAI; (**b**) IBAI.

4.2. System Tests

The adaptability tests of the two systems include three static tests, three dynamic tests, and one test for DDA effectiveness. Each test has five stages with different skill levels for the player.

To facilitate the division of the five skill levels, the authors invited 55 players to participate in internal tests on a small scale. The test requires the players to compete with enemies at multiple fixed global difficulties without DDA and sort them according to the total number of enemies killed within the specified time. The players were equally divided into five groups based on the subsequent results, and the authors selected the median player in each group to ensure that the selected players were representative of each skill level. The five selected players represent different skill levels to participate in the five test processes. Each process is repeated ten times.

4.2.1. Static Tests

The static tests are conducted according to the enemy types, the number of enemies, as well as the interaction area between enemies and the player. The type and number of enemies used in each static test are shown in Table 4.

Table 4. Type and number of enemies used in each static test.

Static Tests	**Algorithms**	**Enemy Melee1**	**Enemy Melee2**	**Enemy Melee3**	**Enemy Ranged1**	**Enemy Ranged2**	**Enemy Ranged3**
Enemy type management	BAI IBAI	1 1	1 1	1 0	0 1	0 1	0 1
Enemy number management	BAI IBAI	8 5	1 2	1 2	0 5	0 2	0 2
Interaction area	BAI IBAI	2 2	0 0	0 0	0 0	0 0	0 0

- Enemy type management tests

It can be found that the eight rectangular grids of BAI can only hold enemies using melee attacks rather than those using ranged attacks (shown in Figure 4a), which means BAI can only manage melee attacks. In contrast, Figure 4b shows that the MRBP ring model in IBAI can accommodate both the enemies that use melee attacks and ranged attacks. IBAI is able to process the management of melee attacks and ranged attacks simultaneously.

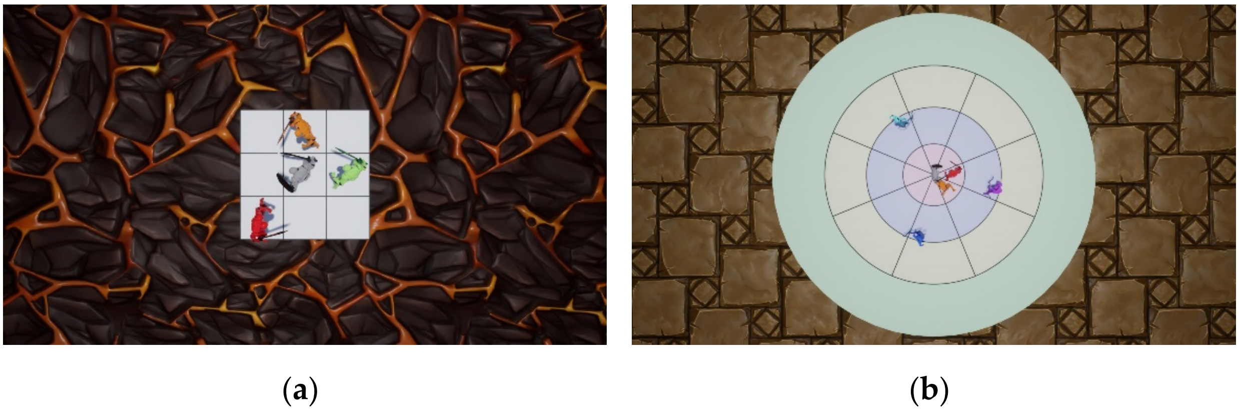

Figure 4. Tests for enemy types that can be managed. (**a**) BAI; (**b**) IBAI.

- Enemy number management tests

As shown in Figure 5a,b, BAI can simply hold one enemy in each grid, which cannot be changed. In the test of IBAI, *a*:*b*:*c* = 1:2:3. Each sector of the melee area, ranged area, and buffer area in the MRBP ring model can hold 1, 3, and 5 enemies, respectively. The radius of each circle in the MRBP ring model can be set adaptively according to the specific game mechanism, thereby adjusting the number of enemies that each sector can hold.

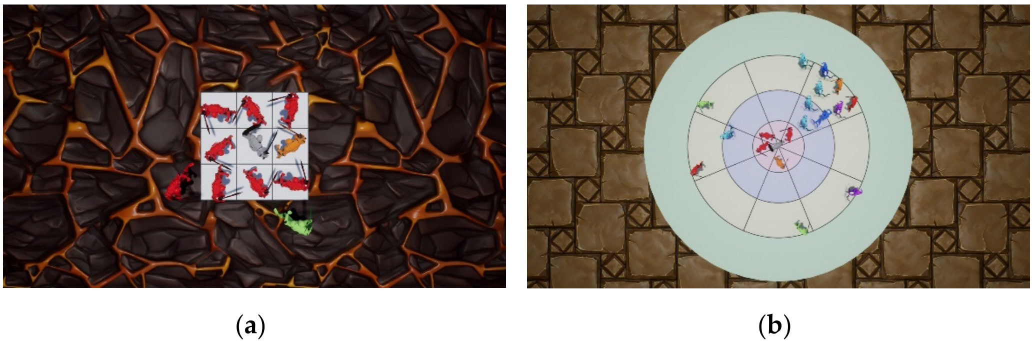
(**a**) (**b**)

Figure 5. Tests for the number of enemies that can be held. (**a**) BAI; (**b**) IBAI.

- Interaction area tests

During the tests of the interaction area, the player attacks the enemy above at a certain angle. The results demonstrate that this enemy gets damaged, but the one in the upper right does not. That is, the interaction area between enemies and the player in the grids at a 45-degree angle to the player is smaller than the ones at a 90-degree angle. Both of the two enemies get damaged at the IBAI level, which verifies the consistency of the interaction area in each sector of the same ring area in the MRBP ring model. The test process of the two algorithms is shown in Figure 6a,b, respectively.

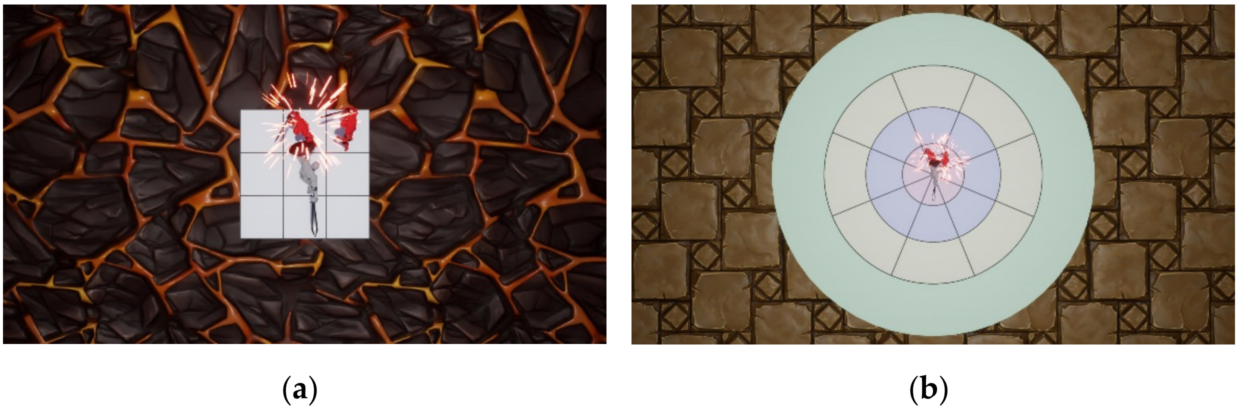
(**a**) (**b**)

Figure 6. Tests for the interaction area between enemies and the player. (**a**) BAI; (**b**) IBAI.

4.2.2. Dynamic Tests

The dynamic tests aimed at testing the attack mechanism of enemies, the allocation logic of the stage manager, as well as the execution process of the RID algorithm. The combat diversity, game balance, stage manager's request process logic, and the strengthened effect of control and scope of application for the stage manager of the RID algorithm are compared and verified with the management through BAI and IBAI. Table 5 shows the type and number of enemies used in each dynamic test.

Table 5. Type and number of enemies used in each dynamic test.

Dynamic Tests	Stages	Enemy Melee1	Enemy Melee2	Enemy Melee3	Enemy Ranged1	Enemy Ranged2	Enemy Ranged3
Attack mechanism	First	0	2	2	0	0	0
	Second	1	1	0	0	0	0
	Third	1	0	0	0	0	0
Allocation logic	First	1	0	3	0	0	0
	Second	0	1	0	0	0	0
	Third	2	0	0	0	0	0
RID algorithm	First	1	0	0	0	0	0
	Second	1	0	1	1	1	1
	Third	0	1	0	0	0	0
	Fourth	1	0	0	0	0	0
	Fifth	0	0	0	1	0	0
	Sixth	0	0	0	1	0	0

- Attack mechanism tests

In the BAI level and IBAI level, the player enters the attack decision area of enemies from the first stage to the third stage successively.

As the BAI level is shown in Figure 7a, the four enemies in the eight grids belong to the first stage; the two above and on the right of the player outside the grids belong to the second stage, and the one on the left outside belongs to the third stage. Two EnemyMelee3 use Skill2 to attack the player. The EnemyMelee2 below uses Skill1, while the one in the upper left uses a normal attack. Thereafter, the player kills the EnemyMelee3 on the bottom left, and then the two enemies in the second stage enter the grids (shown in Figure 7b). EnemyMelee1, EnemyMelee2, and EnemyMelee3 attack with normal attacks, Skill1, and Skill2, respectively.

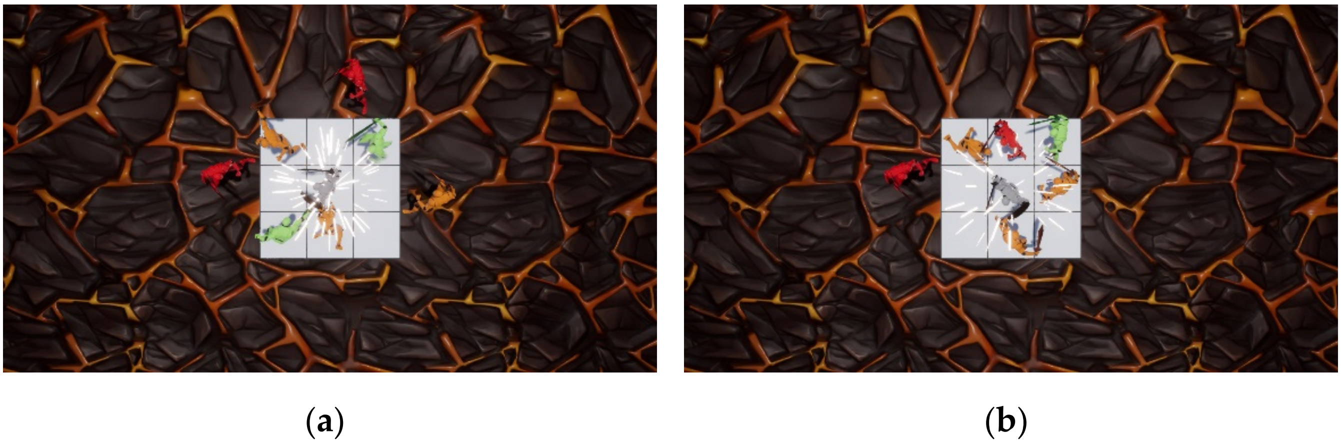

Figure 7. Attack mechanism tests of BAI. (**a**) The first combat scene; (**b**) The second combat scene.

Figure 8a shows the test process of IBAI. The enemies in the first stage are the four in the melee area, among whom EnemyMelee3 attacks with Skill2, Skill1, and a normal attack, and EnemyMelee2 attacks below and on the top left of the player with Skill1 and a normal attack. After EnemyMelee3 on the bottom left is killed by the player, the two enemies in the buffer area in the second stage enter the melee area while the one in the third stage does not (shown in Figure 8b). At this moment, EnemyMelee3, EnemyMelee2, and EnemyMelee1 use Skill2, Skill1, and normal attack alternately, Skill1 and normal attack alternately, and normal attack simply to attack the player.

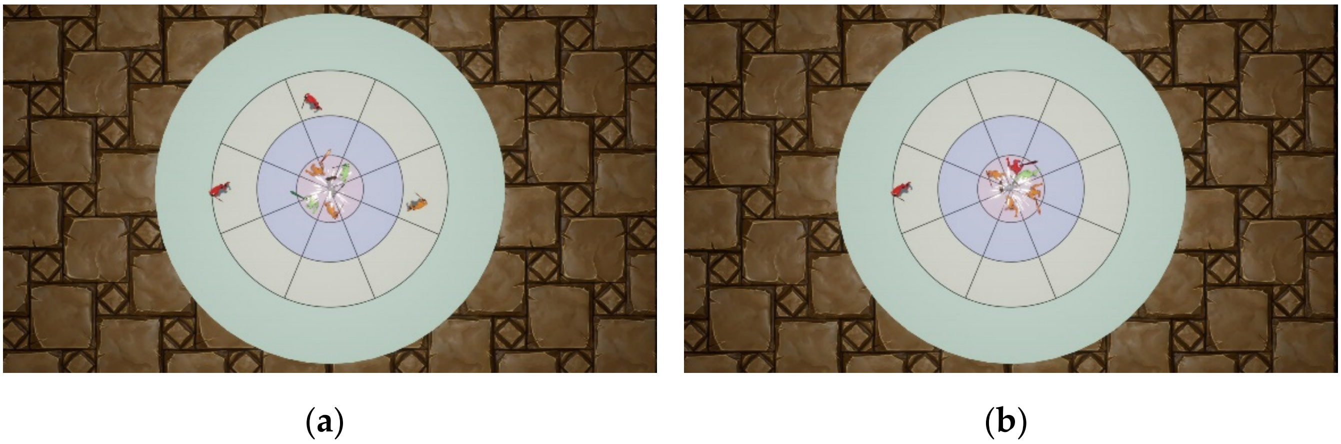

Figure 8. Attack mechanism tests of IBAI. (**a**) The first combat scene; (**b**) The second combat scene.

The results of attack mechanism tests demonstrate that enemies will constantly use the same attack type until the number of enemies that have gained the attack right changes when authorized by the stage manager with BAI. Additionally, the same type of enemy may attack the player with the same attack type at the same time. Compared with BAI, IBAI can provide higher combat diversity and maintain the game's balance. Enemies can use another attack type that meets the requirement of AWT if an attack type is in cooldown. When a skill is in GCD, all the same type of enemies cannot attack with it. The possibility of the same type of enemies attacking the player with the same attack type simultaneously is, therefore, reduced.

- Allocation logic tests

In common with the attack mechanism tests, the player's behaviors in allocation logic tests are still divided into three stages. The player enters the attack decision area of enemies in the first two stages successively.

In the BAI level, the enemies inside and outside the grids belong to the first and second stages, respectively (shown in Figure 9a). The player moves to the left, and enemies in the first stage move to the position assigned by the stage manager. While enemies in the second stage wait in place, whose positions conflict with that of the EnemyMelee1 on the top left of the player. EnemyMelee1 has already been in the grids, but it has not gained the attack right. The test process is shown in Figure 9b. As shown in Figure 9c, the player enters the attack decision area of the two enemies in the third stage after EnemyMelee1 and EnemyMelee2 are killed, and the two enemies wait outside the grids (shown in Figure 9d).

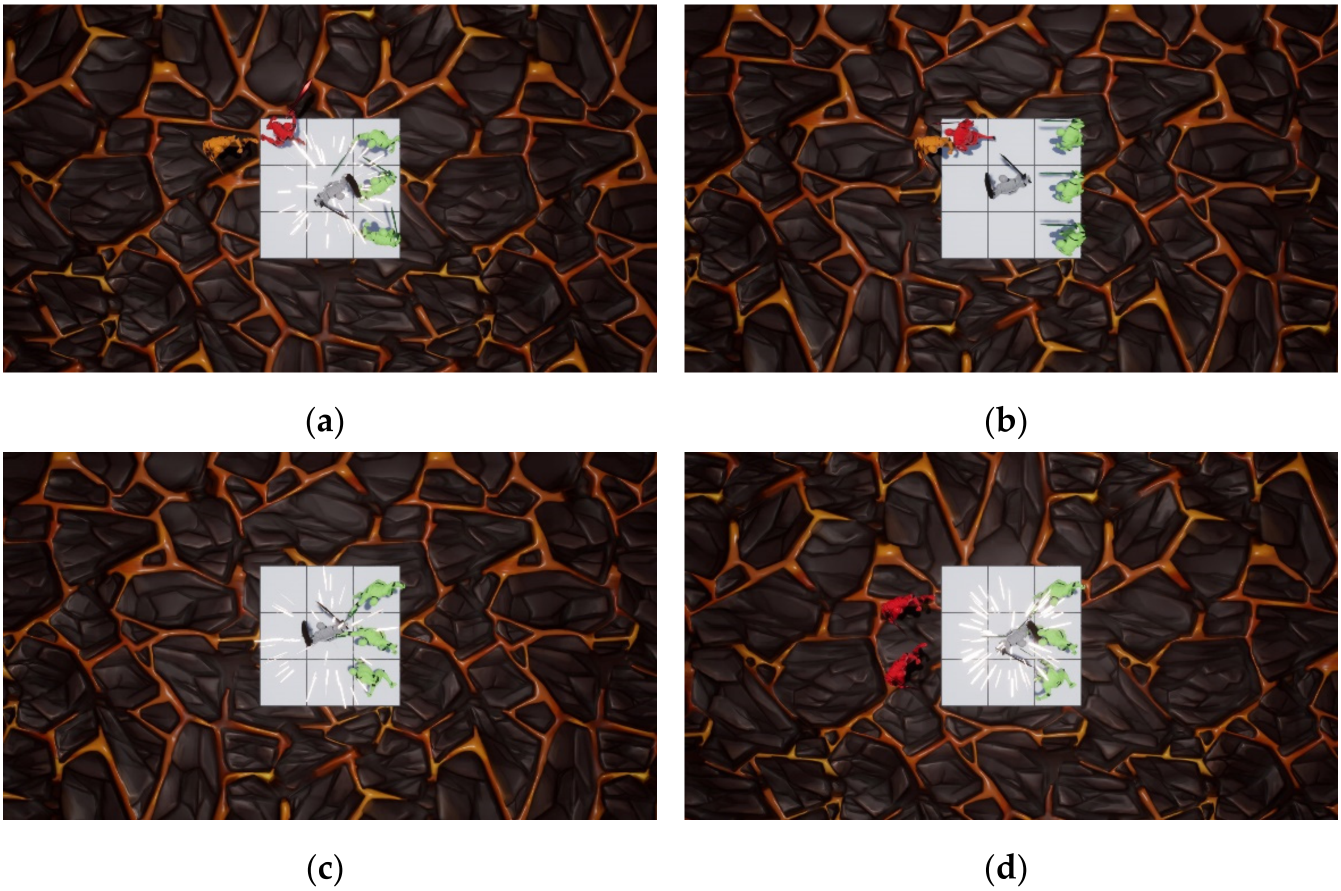

Figure 9. Allocation logic tests of BAI. (**a**) The first combat scene; (**b**) The second combat scene; (**c**) The third combat scene; (**d**) The fourth combat scene.

In the test process of IBAI shown in Figure 10a, the enemies in the first stage are in the melee area, and the ones in the second stage are in the buffer area. When the player moves to the left, all enemies in the pursuit circle move to follow the player (shown in Figure 10b). Thereafter, the player kills EnemyMelee1 and EnemyMelee2, as shown in Figure 10c, and enters the attack decision area of two EnemyMelee1s. The two enemies belong to the third stage, one of which enters the melee area and the other waits in the buffer area. The last step of the test process is shown in Figure 10d.

It can be seen from the results that when the number or position of enemies that have gained the attack right changes with the management through BAI, the position where some enemies had gained the attack right before the change may not be the closest attack position after the change. The situation of the enemy that is not assigned is already in the grids after the change, but its position conflicts with the enemies that have gained the attack right before the change. It happens that the enemy does not gain the attack right as well. Meanwhile, the attacks may fail if multiple enemies register requests to attack simultaneously through the overall test. As for IBAI, the problems of location allocation can be avoided effectively by executing the RID algorithm. The enemy with a higher GP could be assigned a higher priority if multiple enemies register requests at the same time. This can avoid the situation where the enemy cannot attack the player even if the player's CGC is large enough. IBAI is sounder and more logical than BAI.

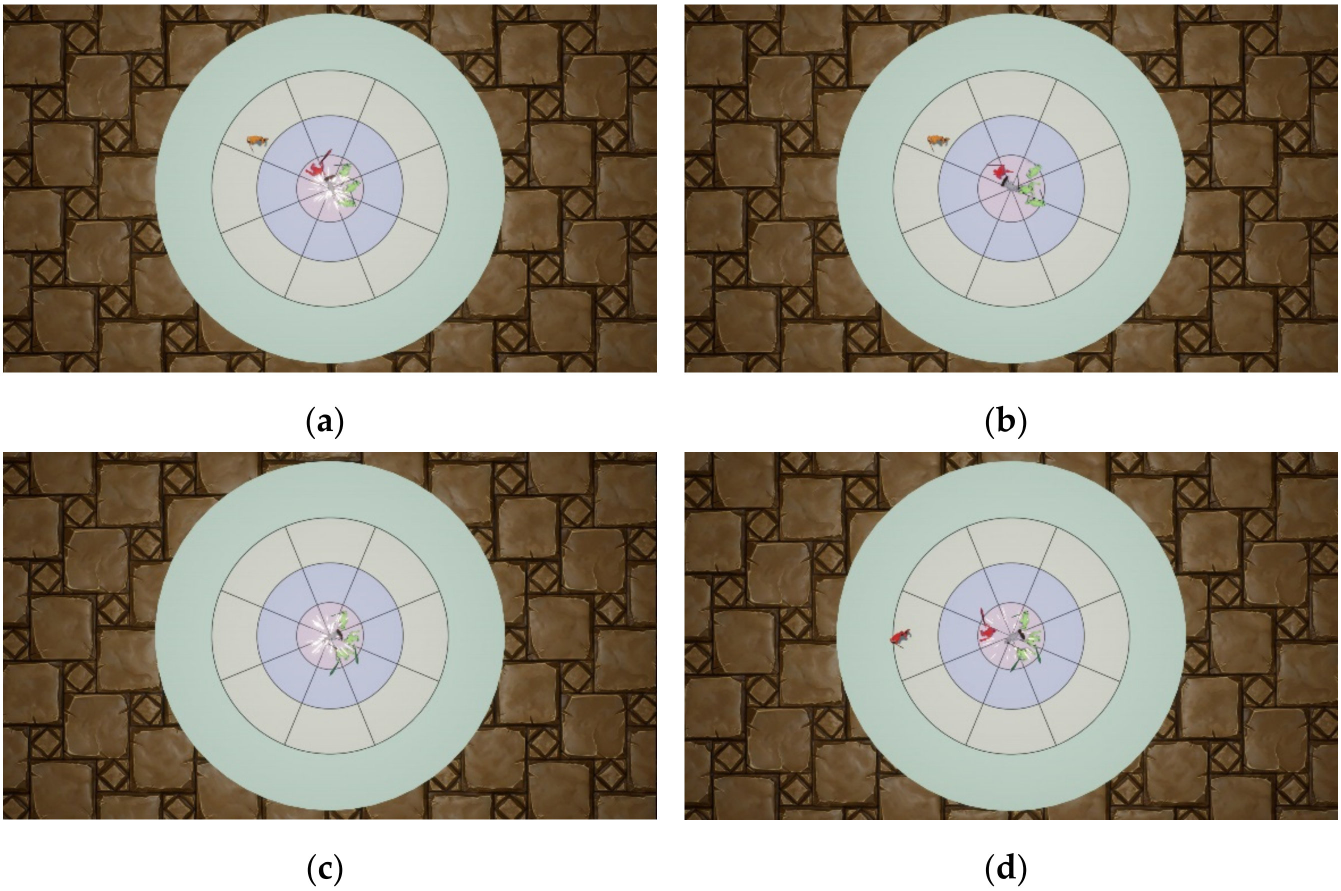

Figure 10. Allocation logic tests of IBAI. (**a**) The first combat scene; (**b**) The second combat scene; (**c**) The third combat scene; (**d**) The fourth combat scene.

- RID algorithm tests

The RID algorithm tests are only performed at the IBAI level. The whole process is divided into six stages and includes all enemy types preset by the system.

During the test, the player enters the attack decision area of the enemy in the first stage. As the player moves below, at a slower speed than the enemy, the enemy will pursue the player. As the player adjusts their speed to be greater than the enemy's speed and maintains the previous movement state, the enemy continues pursuing the player. The system displays the independent MRBP ring model of the enemy in real-time. The model statuses before and after speed adjustments are shown in Figure 11a,b, respectively. As a result, the player kills the enemy and enters the attack decision area of enemies from the second to the fourth stage successively. Among these enemies, the ones in the melee area and ranged area belong to the second stage, while the ones in the third and fourth stages are in the buffer area and pursuit area, respectively. At this moment, the player snaps the EnemyRanged1's aggro in the outer area, and the latter waits in place (shown as Figure 11c). The player then moves to the upper left, and the enemies in the pursuit circle move following the diagram in Figure 11d. When the player stops moving, the enemies move to the corresponding positions, which are shown in Figure 11e. Finally, the player kills all enemies in the current combat and enters the attack decision area of enemies in the last two stages successively. The two EnemyMelee1 move to pursue the player as the player moves to the right. The test process is shown in Figure 11f. Of the two enemies, the one in the pursuit area pursues the player, and the other in the outer area stops pursuing and exits the combat.

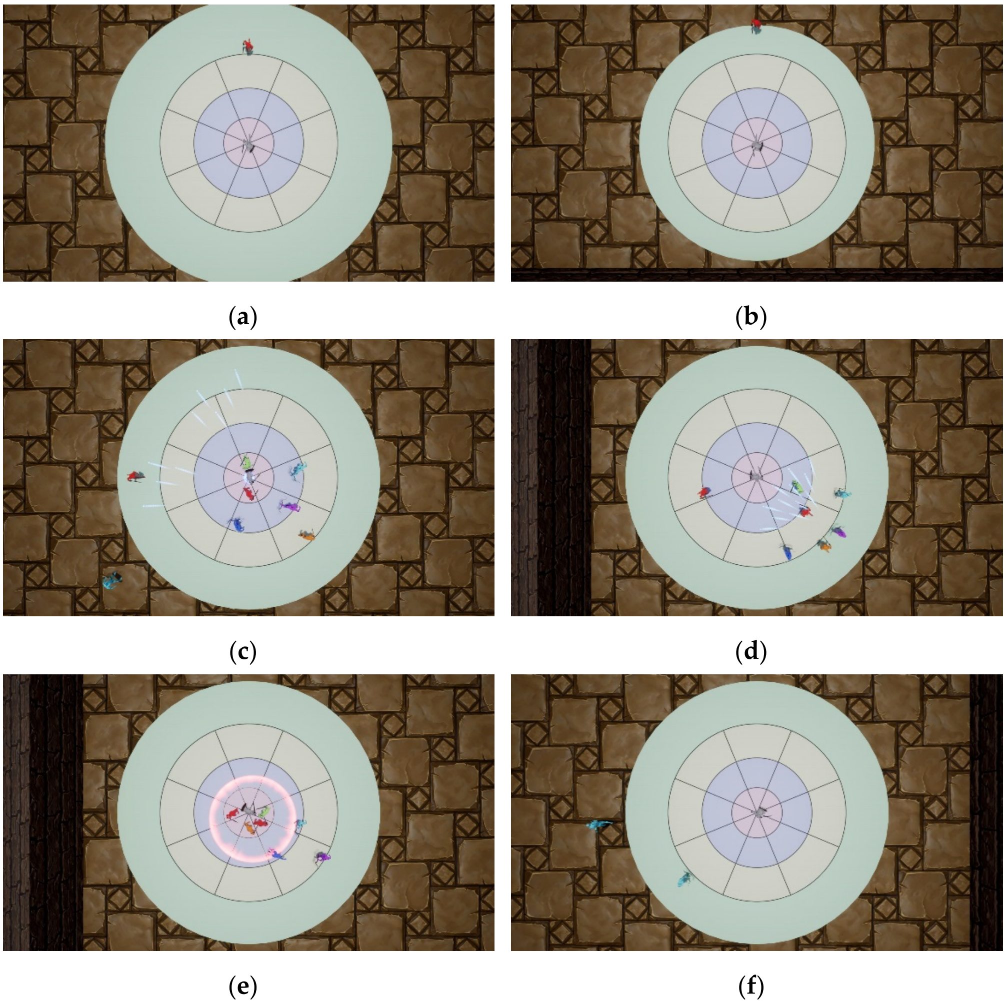

Figure 11. Tests for the execution process of RID algorithm of IBAI. (**a**) The first combat scene; (**b**) The second combat scene; (**c**) The third combat scene; (**d**) The fourth combat scene; (**e**) The fifth combat scene; (**f**) The sixth combat scene.

From the test, it can be concluded that the execution process of the RID algorithm covers all situations when the number of enemies or the ring areas where they are located in the current combat change, except for the change caused by exiting as expected.

4.2.3. DDA Test

IBAI can adjust the combat difficulty effectively without changing the overall preset game difficulty by updating the player's MGC and MAC in real-time, thereby completing the introduction of DDA. In order to compare the effectiveness of DDA with the two algorithms, the authors implemented the DDA mechanism to be consistent with IBAI in the BAI system. The DDA test sets four levels with completely the same enemies, depending on whether DDA is introduced in BAI or IBAI. The five players who are selected to represent ascending skill levels participate in combat at the four levels. The player can be resurrected with full health after dying in the game. The average health point consumption (AHPC) and average combat end time (ACET) of each combat are shown in Table 6.

Table 6. DDA effectiveness parameters of BAI and IBAI.

Algorithms	DDA	Variables	Test 1	Test 2	Test 3	Test 4	Test 5
BAI	Without	AHPC	71.3	58.1	41.1	24.9	10.8
		ACET	9′14″0	8′22″6	6′59″7	5′53″5	4′47″9
	With	AHPC	54.5	48.2	36.4	21.7	12.0
		ACET	8′31″9	7′48″5	6′37″0	5′44″1	4′47″3
IBAI	Without	AHPC	81.3	66.9	49.7	33.2	17.5
		ACET	9′25″8	8′11″7	6′56″1	5′42″0	4′26″4
	With	AHPC	63.0	55.6	48.5	39.6	31.3
		ACET	8′18″2	7′35″8	6′52″9	6′02″2	5′15″1

The test results in the table demonstrate that both BAI and IBAI can achieve the desired effect of reducing the player's AHPC and ACET, regardless of which of them is introduced with DDA. Hence, players of all skill levels can get a better game experience. In the comparison of the effectiveness of DDA, the controllable range of IBAI is larger, and the adjustment is more balanced. In a nutshell, IBAI is more effective than BAI.

4.3. System Adaptability Analysis

According to the test results above, the adaptability of BAI and IBAI in different situations is shown in Table 7. It can be concluded that IBAI is more adaptive, and the application of DDA in the algorithm is more effective.

Table 7. Adaptability of BAI and IBAI.

Parameters	BAI	IBAI
Suitable for action role playing games	Yes	Yes
Set the relationship of enemy types and attack types effectively	Yes	Yes
Enemy types can be managed	Melee	Melee and ranged
The number of enemies can be held	No more than 8	Adaptive based on the game mechanism
The interaction area between enemies and the player	Differ by grid position	All the same
Attack types of enemies can switch automatically	No	Yes
The possibility of the same type of enemies attack the player with the same attack type simultaneously	High	Low
The assigned position of enemies may not be the closest to the player	Yes	No
The assigned position of enemies may be conflicted	Yes	No
Enemies should gain the attack right, but they may not gain it	Yes	No
Multiple enemies may not attack the player while simultaneously registering requests	Yes	No
Effectiveness of DDA	Low	High

4.4. Player Experience Comparison

Confirming the improved effect on the player experience of IBAI is also necessary. "Flow", "immersion", "engagement", "presence", and "fun" are the most commonly used factors to describe the player experience [32–36]. The immersive experience questionnaire (IEQ) and the game engagement questionnaire (GEQ) are two of the most widely used player experience questionnaires in gaming [37,38]. Since the two systems cannot be defined as pure games, it is necessary to screen the questions in IEQ and GEQ and reconstitute the questionnaires applicable to the experiment. The questions in the questionnaire used in the test are shown in Table 8. The questionnaire consists of 25 questions. All questions have a 7-point Likert scale, anchored at the ends with strongly disagree and strongly agree. The order of the questions is randomized for each player to avoid order effects.

Table 8. Questionnaire Items.

ID	Items
Q1	I felt that I really empathized/felt for with the game.
Q2	I was interested in seeing how the game's events would progress.
Q3	I sometimes found myself to become so involved with the game that I wanted to speak to the game directly.
Q4	I enjoyed playing the game.
Q5	I was unaware of what was happening around me.
Q6	I felt detached from the outside world.
Q7	At the time the game was my only concern.
Q8	I did not feel the urge at any point to stop playing and see what was going on around me.
Q9	I did not feel like I was in the real world but the game world.
Q10	To me it felt like only a very short amount of time had passed.
Q11	I lost track of time.
Q12	Things seemed to happen automatically.
Q13	If someone talked to me, I did not hear them.
Q14	I got wound up.
Q15	Time seemed to kind of standstill or stop.
Q16	I felt spaced out.
Q17	I did not answer when someone talked to me.
Q18	I could not tell that I was getting tired.
Q19	My thoughts went fast.
Q20	I lost track of where I was.
Q21	I played without thinking about how to play.
Q22	Playing made me feel calm.
Q23	I played longer than I meant to.
Q24	I really got into the game.
Q25	I felt like I just could not stop playing.

Anonymity is assured in the experiment. The authors declare that all participants were fully informed of this feature, the purpose of the experiment, and the way the collected data were used. The experiment was conducted with the voluntary participation of the participants.

A total of 1014 players accepted the invitation to participate in the gameplay experience on the two systems. The gameplay experience lasted for a week. One week later, the authors distributed two questionnaires, corresponding to the BAI system and the IBAI system, respectively, to the above-mentioned players. No significant difference was found between gender, nationality, and age of different players on the experimental results by potential difference analysis, thus, excluding considerations of these factors. The mean score for each question in the questionnaires for the two systems of the 1014 players is shown in Figure 12.

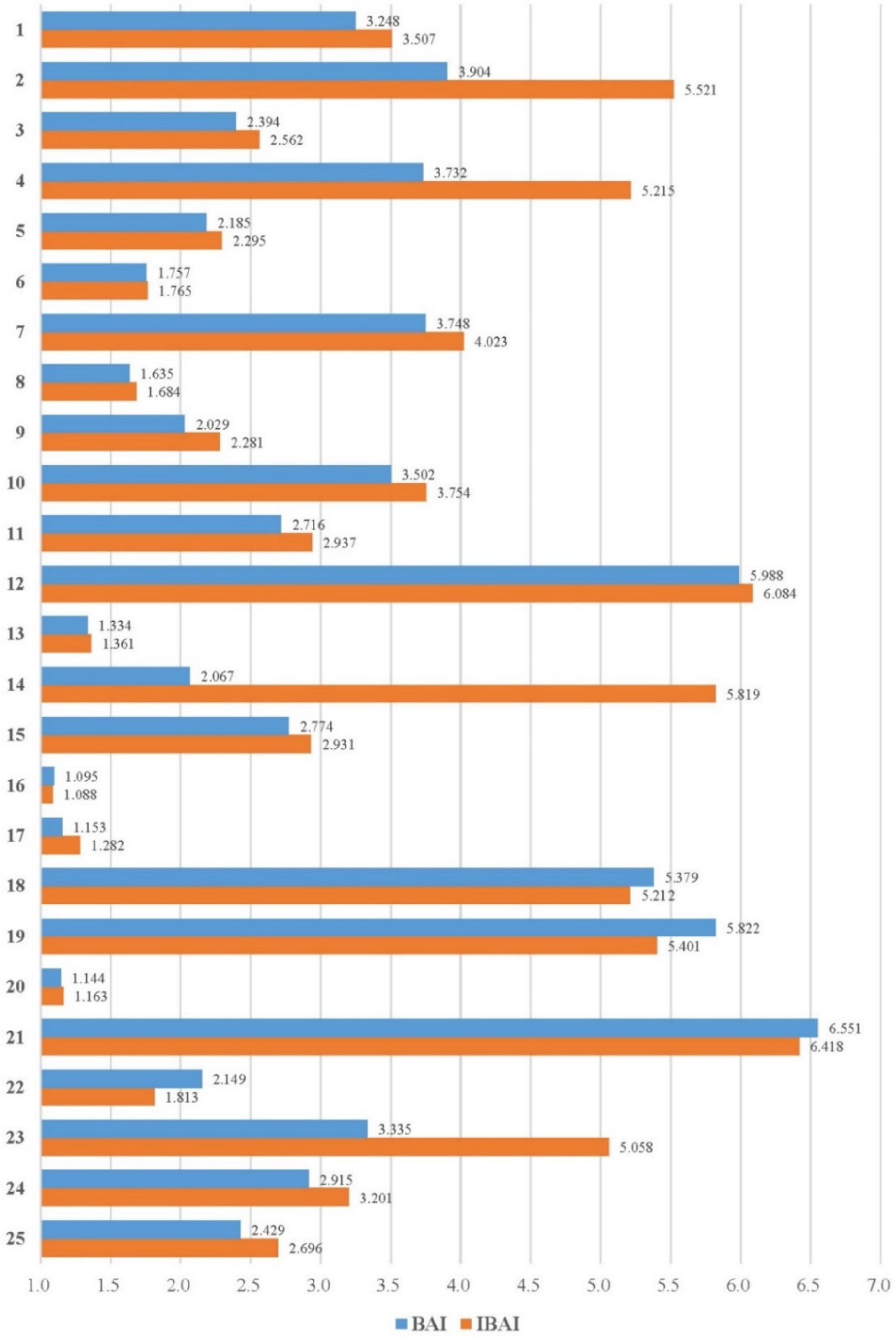

Figure 12. The result of the questionnaires.

Among the 25 questions, the mean score of the IBAI system on 20 questions is higher than that of the BAI. To make the results more convincing, the authors define a screening condition. If the absolute value of the mean score difference (IBAI minus BAI) of the question is less than or equal to 0.3, the difference will be ignored. After screening, the questions for which the absolute value of the mean score difference is greater than 0.3 are Q2, Q4, Q14, Q19, Q22, and Q23, of which the largest mean score difference is Q14 and the smallest one is Q19. It demonstrates that IBAI is more likely to get the players wound up, and they enjoy and are more interested in seeing how the system's events would progress. However, the players' thoughts move fast to ensure that they do not linger in making decisions with BAI. BAI performs better in creating a calm feeling for the players to some extent.

Overall, the mean total scores of the BAI system and the IBAI system were 74.985 (SD = 15.462) and 85.071 (SD = 15.383), respectively. The score of IBAI is much greater than that of BAI, and the standard deviation is slightly smaller. Through analysis, it can be proved that IBAI can enhance the player experience significantly.

5. Conclusions

This paper improves BAI, which is extensively used for managing combat, and proposes IBAI to enhance the adaptability of combat management systems for action role-playing games and provide players with a better game experience.

IBAI accommodates the enemy in the MRBP ring model, whose parameters can be set adaptively according to the specific game mechanism. Both melee and ranged enemies can be managed efficiently, and the interaction area between enemies and the player in each sector of the same ring area is consistent. IBAI sets GP and strengthens the control of the stage manager through RID algorithm with a sound and more logical execution. Moreover, IBAI employs AWT and GCD to enhance combat diversity and game balance to ensure the best player experience.

Various test and questionnaire results demonstrate that IBAI can provide an efficient and complete solution for action role-playing games. The algorithm is also helpful in shortening the development cycle and improving the player retention rate to a certain extent. The design and implementation of IBAI will play an active role in accelerating the widespread use of game AI technologies, especially DDA. It can also fill related research gaps and promote the common development of traditional artificial intelligence and games. Furthermore, this work provides a new insight for leading researchers in game AI, as well as extending the knowledge of nonprofessionals.

Significant strengths notwithstanding, one concern about the algorithm is that it can only be used in action role-playing games at this stage. Particular technical expertise may be required for the users to apply the algorithm effectively during game R and D in different engines. Games realize the long-term goals of general intelligence the best. In future research, IBAI will be optimized for specific game engines and will be refactored to explore its applications for other game types. The generality of the algorithm and the extensibility of its roles within games will be enhanced at that time, so as to realize the frontier of game AI research.

Author Contributions: Conceptualization, Q.M. and T.G.; methodology, Q.M.; software, Q.M.; validation, Q.M. and T.G.; formal analysis, Q.M.; investigation, Q.M. and T.G.; resources, Q.M. and T.G.; data curation, Q.M. and T.G.; writing—original draft preparation, Q.M.; writing—review and editing, Q.M. and T.G.; visualization, Q.M.; supervision, T.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by China Fundamental Research Funds for the Central Universities, grant number N2017003.

Institutional Review Board Statement: Not applicable. The anonymous questionnaires in the study were used to survey player experience. All data collected is merely used for the experiment, and any personal details of the participants other than the information related to the results of the experiment shown in the paper will not be disclosed to avoid ethical issues. Ethics approval is not required for this research according to the cases above and relevant legislation. Any individual or organization cannot use the questionnaire data in the research without the authors' approval.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Acknowledgments: The authors gratefully acknowledge the efforts of the participants of their gameplay and questionnaire who invested their valuable time to contribute to this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Heidari, A.; Navimipour, N.J.; Unal, M. Applications of ML/DL in the Management of Smart Cities and Societies Based on New Trends in Information Technologies: A Systematic Literature Review. *Sustain. Cities Soc.* **2022**, *85*, 104089. [CrossRef]
2. Wen, B.J.; Chang, C.R.; Lan, C.W.; Zheng, Y.C. Magnus-Forces Analysis of Pitched-Baseball Trajectories Using YOLOv3-Tiny Deep Learning Algorithm. *Appl. Sci.* **2022**, *12*, 5540. [CrossRef]
3. Heidari, A.; Jabraeil Jamali, M.A.; Jafari Navimipour, N.; Akbarpour, S. Deep Q-Learning Technique for Offloading Offline/Online Computation in Blockchain-Enabled Green IoT-Edge Scenarios. *Appl. Sci.* **2022**, *12*, 8232. [CrossRef]
4. Liu, J.; Togelius, J.; Pérez-Liébana, D.; Lucas, S.M. Evolving Game Skill-Depth Using General Video Game AI Agents. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 2299–2307. [CrossRef]
5. Xia, B.; Ye, X.; Abuassba, A.O. Recent Research on AI in Games. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 505–510. [CrossRef]
6. Yannakakis, G.N.; Togelius, J. A Panorama of Artificial and Computational Intelligence in Games. *IEEE Trans. Comput. Intell. AI Games* **2014**, *7*, 317–335. [CrossRef]
7. El Rhalibi, A.; Wong, K.W.; Price, M. Artificial Intelligence for Computer Games. *Int. J. Comput. Games Technol.* **2009**, *2009*, 251652. [CrossRef]

8. Jie, J.; Yang, K.; Haihui, S. The Application of AI for the Non Player Character in Computer Games. In Proceedings of the 2011 International Conference on Computational and Information Sciences, Chengdu, China, 21–23 October 2011; pp. 1049–1050. [CrossRef]
9. Wiemeyer, J.; Nacke, L.; Moser, C.; Floyd'Mueller, F. Player Experience. In *Serious Games*; Dörner, R., Göbel, S., Effelsberg, S., Wiemeyer, J., Eds.; Springer: Cham, Switzerland, 2016; pp. 243–271. [CrossRef]
10. Denisova, A.; Nordin, A.I.; Cairns, P. The Convergence of Player Experience Questionnaires. In Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play, Austin, TX, USA, 16–19 October 2016; pp. 33–37. [CrossRef]
11. Nacke, L.; Drachen, A.; Kuikkaniemi, K.; Niesenhaus, J.; Korhonen, H.J.; Hoogen, W.M.; Poels, K.; IJsselsteijn, W.A.; De Kort, Y.A. Playability and Player Experience Research. In Proceedings of the 2009 DiGRA International Conference: Breaking New Ground: Innovation in Games, Play, Practice and Theory, London, UK, 1–4 September 2009.
12. Köknar, C. Who is at the Center?: Designing Playful Experiences by Using Player-Centered Approach. In Proceedings of the International Conference on Human-Computer Interaction, Orlando, FL, USA, 26–31 July 2019; pp. 11–21. [CrossRef]
13. Mustač, K.; Bačić, K.; Skorin-Kapov, L.; Sužnjević, M. Predicting Player Churn of a Free-to-Play Mobile Video Game Using Supervised Machine Learning. *Appl. Sci.* **2022**, *12*, 2795. [CrossRef]
14. Karavidas, L.; Apostolidis, H.; Tsiatsos, T. Usability Evaluation of an Adaptive Serious Game Prototype Based on Affective Feedback. *Information* **2022**, *13*, 425. [CrossRef]
15. Zohaib, M. Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review. *Adv. Hum. Comput. Interact.* **2018**, *2018*, 5681652. [CrossRef]
16. Sepulveda, G.K.; Besoain, F.; Barriga, N.A. Exploring Dynamic Difficulty Adjustment in Videogames. In Proceedings of the 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), Valparaiso, Chile, 13–27 November 2019; pp. 1–6. [CrossRef]
17. Preuss, M.; Risi, S. A Games Industry Perspective on Recent Game AI Developments. *KI Künstl. Intell.* **2020**, *34*, 81–83. [CrossRef]
18. Williams, J.P.; Hendricks, S.Q.; Winkler, W.K. *Gaming as Culture: Essays on Reality, Identity and Experience in Fantasy Games*; McFarland: Jefferson, NC, USA, 2014; pp. 1–14.
19. Van Der Linden, R.; Lopes, R.; Bidarra, R. Procedural generation of dungeons. *IEEE Trans. Comput. Intell. AI Games* **2013**, *6*, 78–89. [CrossRef]
20. Foong, N.W.; On, C.K.; Alfred, R.; Teo, J.; Ibrahim, A.A.A. Interactive Procedural Generation for Items in Role-Playing Game. In Proceedings of the 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS), Kota Kinabalu, Malaysia, 21 October 2017; pp. 150–154. [CrossRef]
21. Ferreira, L.; Lelis, L.; Whitehead, J. Computer-Generated Music for Tabletop Role-Playing Games. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Worchester, MA, USA, 19–23 October 2020; pp. 59–65.
22. Primanita, A.; Effendi, R.; Hidayat, W. Comparison of A∗ and Iterative Deepening A Algorithms for Non-Player Character in Role Playing Game. In Proceedings of the 2017 International Conference on Electrical Engineering and Computer Science (ICECOS), Palembang, Indonesia, 22–23 August 2017; pp. 202–205. [CrossRef]
23. Huynh, E.; Nyhout, A.; Ganea, P.; Chevalier, F. Designing Narrative-Focused Role-Playing Games for Visualization Literacy in Young Children. *IEEE Trans. Vis. Comput. Graph.* **2020**, *27*, 924–934. [CrossRef] [PubMed]
24. Wang, J.Y. A Variety Weapons and Armors Design Algorithm for Role-Playing Games. In Proceedings of the 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Takamatsu, Japan, 1–3 June 2015; pp. 1–5. [CrossRef]
25. Brown, J.A. Evolved Weapons for RPG Drop Systems. In Proceedings of the 2013 IEEE Conference on Computational Inteligence in Games (CIG), Niagara Falls, ON, Canada, 11–13 August 2013; pp. 1–2. [CrossRef]
26. Hitchens, M.; Drachen, A. The Many Faces of Role-Playing Games. *Int. J. Role-Play.* **2009**, *1*, 3–21.
27. Macklin, C.; Sharp, J. *Games, Design and Play: A Detailed Approach to Iterative Game Design*; Addison-Wesley Professional: Upper Saddle River, NJ, USA, 2016; pp. 47–76.
28. Adams, E. *Fundamentals of Game Design*; Addison-Wesley Professional: Upper Saddle River, NJ, USA, 2014; pp. 255–312.
29. Dawe, M. Beyond the Kung-Fu Circle: A Flexible System for Managing NPC Attacks. In *Game AI Pro: Collected Wisdom of Game AI Professionals*; Rabin, S., Ed.; CRC Press: Boca Raton, FL, USA, 2013; pp. 369–375.
30. Sellers, M. *Advanced Game Design: A Systems Approach*; Addison-Wesley Professional: Upper Saddle River, NJ, USA, 2017; pp. 267–292.
31. Thorn, A. *Game Development Principles*; Cengage Learning: Singapore, 2013; pp. 1–30.
32. Chen, J. Flow in Games (and Everything Else). *Commun. ACM* **2007**, *50*, 31–34. [CrossRef]
33. Bastos, A.S.; Gomes, R.F.; Dos Santos, C.C.; Maia, J.G.R. Assessing the Experience of Immersion in Electronic Games. In Proceedings of the 2017 19th Symposium on Virtual and Augmented Reality (SVR), Curitiba, Brazil, 1–4 November 2017; pp. 146–154. [CrossRef]
34. Bachelder, S.; Santhanam, R.; Hayashi, M.; Nakajima, M. Engagement in Computer and Video Games. In Proceedings of the 2013 International Conference on Cyberworlds, Yokohama, Japan, 21–23 October 2013; p. 371. [CrossRef]
35. Witmer, B.G.; Singer, M.J. Measuring Presence in Virtual Environments: A Presence Questionnaire. *Presence* **1998**, *7*, 225–240. [CrossRef]

36. Bernhaupt, R.; Eckschlager, M.; Tscheligi, M. Methods for Evaluating Games: How to Measure Usability and User Experience in Games? In Proceedings of the International Conference on Advances in Computer Entertainment Technology, Salzburg, Austria, 13–15 June 2007; pp. 309–310. [CrossRef]
37. Jennett, C.; Cox, A.L.; Cairns, P.; Dhoparee, S.; Epps, A.; Tijs, T.; Walton, A. Measuring and Defining the Experience of Immersion in Games. *Int. J. Hum. Comput. Stud.* **2008**, *66*, 641–661. [CrossRef]
38. Brockmyer, J.H.; Fox, C.M.; Curtiss, K.A.; McBroom, E.; Burkhart, K.M.; Pidruzny, J.N. The Development of the Game Engagement Questionnaire: A Measure of Engagement in Video Game-Playing. *J. Exp. Soc. Psychol.* **2009**, *45*, 624–634. [CrossRef]