Adventures of AI Directors Early in the Development of Nightingale

Kristen K. Yu, [1, 3, 4] **Matthew Guzdial,** [1, 3] **Nathan R. Sturtevant,** [1, 3, 4] **Morgan Cselinacz,** [2] **Chris Corfe,** [4] **Izzy Hubert Lyall,** [4] **Chris Smith** [4]

[1]Department of Computing Science, University of Alberta
[2] Department of Digital Humanities, University of Alberta
[3] Alberta Machine Intelligence Institute
[4] Inflexion Games Inc
{kkyu, guzdial, nathanst, cselinac}@ualberta.ca, {chris.corfe, izzy.hubertlyall, cms}@inflexion.io

Abstract

Players can sometimes engage with parts of a video game that they do not enjoy if the game does not try to adapt the experience to the player's preference. AI directors have been used in the past to tailor player experience to different people. In industry, AI directors are relatively uncommon and are typically domain-specific and rules-based. In this paper, we present a reinforcement learning-based AI director developed for the industry game Nightingale with the help of Inflexion Games. We ran an experiment to evaluate the effectiveness of the AI director in creating a desired player experience, but found inconclusive evidence. In line with this year's theme, we present our negative results and their implications for future AI directors, along with general discussion from working closely with an industry partner.

1 Introduction

An AI director is an agent in a video game that attempts to improve a player's experience by changing aspects of the game. For example, in the game Left 4 Dead (Valve 2008), an AI director can alter enemy spawns, item placements and other elements in an attempt to have players experience a desired tension curve (Booth 2009). There has been significant academic interest in AI directors (Thue 2007a,b; Harrison and Roberts 2014; Khaliq and Watson 2018; Spronck et al. 2006; Jennings-Teats, Smith, and Wardrip-Fruin 2010), but these systems are not implemented in industry games, making it difficult to analyze them in their intended application. Further, it's rare to see industry developers discuss the AI directors they implement in detail (Thompson 2014).

In this paper we propose a reinforcement learning (RL) based AI director intended to reward players for engaging with content they prefer. To help evaluate this AI director, we partnered with a video game studio, Inflexion Games, and implemented the AI director in their upcoming game Nightingale. The AI director was evaluated in a human subject experiment with over 200 participants capturing both quantitative and qualitative data. The results from this experiment were inconclusive, but still offer some insights. Given AIIDE 2022's theme of mis-spun tales, the results of this experiment are submitted as negative results. We cover our experimental results and the experience of working with our industry partner, and the takeaways learned from both.

This paper contains two primary contributions. The first contribution is an interpretation of our negative results. The RL AI director failed to outperform a random AI director in all metrics. We offer potential explanations, such as a mismatch between the AI director design assumptions and the game that was played, as well as other analysis relevant to future AI director research. The second contribution is from the experience of collaborating closely with an industry partner, which includes direction on when it is appropriate to include academic research in the development cycle in the game, and other practical considerations.

2 Background

AI directors are used to influence the state of a video game in various ways. In academia, AI directors fall into the category of generalized experience management (GEM), which models the problem as a Markov decision process (MDP) where the AI director maximizes some reward (Thue 2010). Some common problems are adjusting the difficulty to a player's skill level (Jennings-Teats, Smith, and Wardrip-Fruin 2010; Tychsen, Tosca, and Brolund 2006), and adjusting the narrative to suit the player's preferences (Riedl and Young 2010; Thue 2007a; Giannatos et al. 2011; Yu and Riedl 2013; Khaliq and Watson 2018).

There are some AI directors that have explored similar settings to ours. PaSSAGE is an AI director that determines player preferences through player actions and chooses which branch of narrative based on those preferences (Thue 2007a,b), but its effectiveness was inconclusive for a generalized population of players. Harrison and Roberts developed an analytics-driven AI director that models the game state in order to choose which quests to present the player (Harrison and Roberts 2014). This AI director showed an increase in player retention, but lacked any qualitative discussion for how players perceived the adaptive experience. The Omni framework was proposed as a way to dynamically generate suitable quests based off game state, player state, and other information (Khaliq and Watson 2018), but lacks any experimental results for the performance of the system.

To support an AI director that provides quests, a cursory understanding of video game quest theory is necessary. We used a general, technical quest definition defined as $Q =$

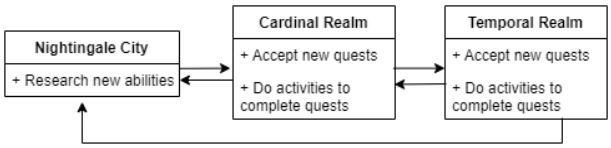

Figure 1: Boxes indicate locations, and arrows indicate how a player was allowed to transition between locations.

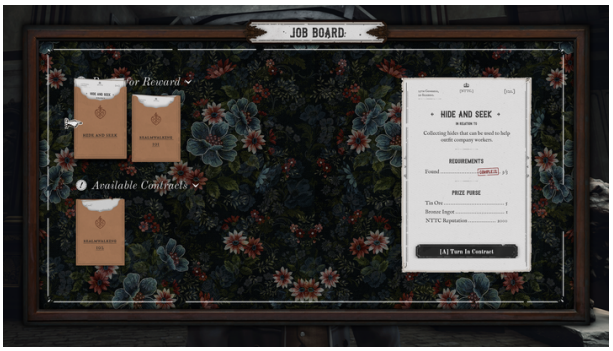

Figure 2: A screenshot of the job board that the players needed to interact with in order to accept and submit quests.

$\langle T, \leq, R, d(R) \rangle$ where T is a set of tasks for the player to complete, and R is the set of rewards (Yu, Sturtevant, and Guzdial 2020; Yu, Guzdial, and Sturtevant 2021). These are the most relevant pieces of the quest definition, and a full discussion lies outside the scope of this paper.

3 Nightingale the Game

Inflexion Games is currently working on a game that is scheduled for early access release. When this research was conducted, Inflexion Games was producing the game as a massive multiplayer online role-playing game (McKeand 2022).

In the game, players engaged with a variety of locations and gameplay. Players started their adventure in Nightingale City, where players interacted with NPCs to progress. A specific area called the "cardinal realm" was connected to the city, where players engaged with the main part of the game. Players could build houses, hunt wildlife, harvest resources, and craft items. Players looking for additional activities could go to a "temporal realm", which was an area that featured content not found in the "cardinal" realm.

3.1 Nightingale Game Loop

The main game loop is shown in Figure 1. In order to progress through the game, players needed to unlock new abilities using experience points, and players completed quests to gain experience points. Players accepted Quests at job boards in both the "cardinal" and "temporal" realms.

The AI director was deployed in the job board, which was stationed at the entrance to each realm, and was a critical part of the main game loop. When a player interacted with the job board, they were presented with three quests, as shown in Figure 2. Players used three actions to interact with quests: accept, submit, and abandon. Accept was when a player chose to complete that quest, submit was when a player turned in a completed quest for the reward, and abandon was when a player chose to stop attempting to complete an already accepted quest. Players were allowed to accept and submit quests when they interacted with the job board, but could abandon quests at any time. Every time a player interacted with the job board, they were shown a new set of quests selected by a particular algorithm. The board could be opened and closed in succession if players were looking for different quests, and players could accept a maximum of five quests at any given time.

Quest Type	*Example Quest*
Build	Build 6 Shack Foundation Structures
Craft	Craft 1 Crude Axepick
Kill	Kill 5 Wolves
Acquire	Acquire 10 Common Iron Ore

Table 1: Example quests that players could complete

3.2 Quests in Nightingale

Quests were designed such that each one had exactly one task. There were four different types of gameplay: build, craft, kill, and acquire, which represent the four quest types. Each quest of a particular type required taking the action associated with that gameplay. Table 1 shows an example of quests for each type. The number of quests was approximately balanced between quest types. The game had 33 build quests, 22 craft quests, 27 kill quests, and 19 acquire quests.

4 AI Director Design

The concept for the AI director was originally proposed by designers at Inflexion Games. They wanted to include a machine learning driven AI director because they believed that these techniques could be used to improve player experience. Because the AI director was internally approved before searching for outside, academic help, the studio planned for the incorporation of the technology, and decided the best time for integration. The AI director was designed with two goals: identify the gameplay types that players prefer to provide them with more quests of that type, and support players who create their own goals using quest rewards. The AI director was intended to shape the player experience by changing the types of presented quests.

The design also made two fundamental assumptions. First, the AI director assumed that each quest was of equal quality, such that the only difference between quests was type. Second, the AI director assumed that it would be in the final version of the game, rather than the version of the game that was played in the playtest. The second assumption is a possible explanation for some of the negative results we discuss later in the paper.

4.1 Combinatorial Multi-Armed Bandit Algorithm

The proposed problem space of the AI director naturally lends itself to be modeled as an RL problem using the GEM framework (Thue 2010). Each player has a preference for different quest types, which can be learned using RL. The reward signal for this problem is whether the quests were accepted or not. We simplified the problem model to a single state in order to model the AI director as a bandit problem, as suggested by Sutton and Barto 2020. All possible player states are simplified to one state and the arms are the quests that are presented to the player.

To learn the player preference over the quest types, we used a combinatorial multi-armed bandit algorithm (CMAB). CMABs are a class of RL algorithms that reward sets of arms, called superarms, in order to learn a distribution (Chen, Wang, and Yuan 2013; Ontanon 2013). This section describes ideas used by CMAB as a whole, and the following section will discuss the specific application of CMAB for the AI director. Let m be the number of arms, and t be time. Each arm is associated with a random variable $X_t(i)$ for $1 \leq i \leq m$ and $t \geq 1$. $X_t(i)$ denotes the outcome of each arm, which is the result of choosing arm i at time t. Each variable $X_t(i)$ is iid with expected mean μ_i. $\mu = \{\mu_1, \mu_2, ...\mu_m\}$ is the expected mean of all arms. **S** is the set of all super arms, and is the power set of all possible combinations of arms in m. $|\mathbf{S}| = 2^{|m|} - 1$ to excludes the empty set. A single super arm $S \in \mathbf{S}$ is played in a round, and the outcomes for the set of arms in S are revealed. A reward $R_t(S)$ is given to each arms $i \in S$. $N_t(i)$ is the number of times arm i has been played up to time t.

In the selection of a superarm, the CMAB attempts to maximize reward. Let R_t be reward at time t, and Q_t be the action value of a superarm. In our application, we use CMAB to learn the action value of each super arm S according to $Q_{t+1}(S) = Q_t(S) + \alpha(R_t(S) - Q_t(S))$ (Sutton and Barto 2020). A superarm is selected according to the upper confidence bound (UCB) algorithm, where $S_t = argmax_s(Q_t(S) + c\sqrt{\frac{ln(t)}{N_t(S)}})$ (Auer, Cesa-Bianchi, and Fischer 2002).

4.2 Quest Selection

In this section we describe the specific case of the CMAB used by the AI director to select quests. To do so, each arm and superarm need to be defined. Let q represent a single quest, P represent a quest proposal, and n represent the total number of unique quests in a game. A quest proposal P is a set of k quests, and is the set of quests that is simultaneously shown to the player. $P = \{q_1, q_2, ...q_k\}$. Each quest in a proposal is unique. Let **P** be the set of all quest proposals. Because of UI constraints, we propose exactly k quests to the player. $\forall P \in \mathbf{P}$, $|P| = k$. **P** can be generated by calculating all unique combination of quests.

Instead of using each quest as an arm and proposals as a superarm, we use an abstraction to create arms and superarms to reduce the computational complexity. We explain this decision in more detail later in the section. We use the abstraction of quest type, based on the recommendation of designers at Inflexion Games. Let τ be a quest type, and let T be the set of all quest types. In Nightingale, T is {craft, build, kill, acquire}. Let $f : q \rightarrow \tau$ be a function which inputs a quest q and outputs the type of that quest $\tau \in \mathrm{T}$, and is many-to-one in order to be effective at reducing the size of **P**. Let $f(P)$ be the set of τ such that $\forall\ q \in P$, f is applied, as shown in Equation 1.

$$f(P) = \{f(q_1), f(q_2), ...f(q_k)\} = \{\tau_1, \tau_2, ...\tau_k\} \quad (1)$$

f is many-to-one, which causes some quest proposals P to contain the same set of quest types $f(P)$ after transformation. From Section 3, the AI director assumed that all quests have equal quality, and the only difference between quests is the quest type. Therefore, this loss of information is acceptable according to the AI director design. Let f**(P)** be the set of all $f(P)$ where f is applied to every P in **P**. Each proposal $f(P)$ maintains the same size k, but allows multiple of the same quest type to be present in a single proposal $f(P)$. f**(P)** can be calculated using combinations with repetition. $|f\mathbf{(P)}| = \binom{|\mathrm{T}|+k-1}{k}$. This set is smaller than the set of unique quest proposals $\binom{|\mathbf{(P)}|=n}{k}$ and is easier to compute. This transformation provides a new basis for which to create arms and superarms. Instead of using P as an arm and using **P** to create superarms, we use $f(P)$ as an arm and f**(P)** to create superarms.

Let S be a superarm, where each superarm is defined based on similarity of quest types between arms. For example, one superarm is the set of all $f(P)$ that contain at least one crafting quest. Another superarm will be the set of all $f(P)$ that contain at least one crafting quest and one acquiring quest.

Next, the reward for the arms and superarms needs to be defined. The reward for each arm depends on the outcome of a quest proposal, which is not the arm. The outcome of P is considered instead of the outcome of $f(P)$ because players are shown P, not the abstracted version $f(P)$. Let $X_t(P)$ be the outcome of a quest proposal P at time t. $X_t(P)$ is how many quests in proposal P that a player accepts at time t. $X_t(P) \in \{0, ..., k\}$. To calculate $X_t(P)$, an acceptance function is needed which determines whether a particular quest $q \in P$ is accepted. Let $g_t : player, q \rightarrow \{0, 1\}$ be the acceptance function at time t which takes a player and a quest as inputs and outputs a 0 if a player does not accept, and 1 if a player accepts. g_t is a function of a quest and not a quest type because a quest is shown to the player, not the quest type. Applying g to all q in P, and including the player input, is shown in Equation 2.

$$\begin{aligned} g_t(P) &= \{g_t(player, q_1), g_t(player, q_2), ..., g_t(player, q_k)\} \\ &= \{\{0, 1\}, \{0, 1\}, ...\{0, 1\}\} \end{aligned} \quad (2)$$

The value of $X_t(P)$ is calculated by summing the results from the acceptance function transformation. $X_t(P) = \sum_{i=1}^{k} g_t(player, q_i)$. $X_t(P)$ can only be used as a reward for the associated arm $f(P)$ where P was shown to the player. In the algorithm described in Section 4.1, an entire

superarm is evaluated at a single time step. This is not possible in our domain because only one quest proposal can be shown to the player at time t. Instead, the information learned from $X_t(P)$ is generalized to the rest of the arms in the superarm. Since the superarms were constructed based on the quest proposal's similarity to each other, we assume similar quest proposals should have the same reward.

Each arm in a superarm is rewarded based on the similarity of each arm to the quest proposal that was evaluated. Let $A_t(P)$ be the acceptance set, where $A_t(P) = \{\tau_i \in f(P) \mid g_t(player, q) = 1\}$. $A_t(P)$ is the set of all quest types that were accepted by the player. $A_t(P)$ can be calculated by first applying g_t to get the set of quests that were accepted by the player, and then applying f to only the accepted quests in P. $A_t(P) \subseteq f(P)$. The powerset of $A_t(P)$ is used to identify superarms to reward, denoted as $\mathcal{P}$. For every set $a \in \mathcal{P}(A_t(P)), a \neq \emptyset$, a identifies a superarm S if a is a subset of every $f(P)$ in S. Let S be a superarm, P be a quest proposal that is shown to the player, and $f(P)$ be the associated arm. Let P' be a quest proposal that is not P, where $f(P')$ is not equal to $f(P)$, and P' is not shown to the player. Let $f(P')$ be in S. The reward $R_t(f(P))$ for $f(P)$ is $X_t(P)$. The reward for all other arms $f(P')$ in S are rewarded with $|a|$. $\forall f(P') \in S, f(P') \neq f(P), R_t(f(P')) = |a|$. a rewards $f(P')$ with a higher reward for a higher degree of similarity between $f(P)$ and $f(P')$, since $|a|$ will be larger when there are more quest types in common between $f(P)$ and $f(P')$.

For example, let a quest proposal P = {Kill 10 deer, kill 10 bears, build 5 walls, build 5 doors, acquire 10 logs}. $f(P)$ = {kill, kill, build, build, acquire}. $g_t(P)$ = {1, 1, 0, 0, 1} at time t for some player. $A_t(P) = \{kill, kill, acquire\}$. There are 5 sets in the powerset of $A_t(P)$ (not including the empty set): $a_1 = \{kill\}$, $a_2 = \{kill, kill\}$, $a_3 = \{acquire\}$, $a_4 = \{kill, acquire\}$, $a_5 = \{kill, kill, acquire\}$. Therefore, $f(P)$ is part of five superarms: S_1 where all arms have at least one kill quest as defined by a_1, S_2 where all arms have at least two kill quests as defined by a_2, S_3 where all arms have at least one acquire quest as defined by a_3, S_4 where all arms have at least one kill and one acquire quest as defined by a_4, and S_5 where all arms that have at least one acquire and two kill quests as defined by a_5. For all arms in S_1, the reward = $|a_1|$ = 1; for all arms in S_2, the reward = $|a_2|$ = 2; for all arms in S_3, the reward = $|a_3|$ = 1; for all arms in S_4, the reward = $|a_4|$ = 2, and for all arms of S_5,the reward = $|a_5|$ = 3. In the case where an arm $f(P') \neq f(P)$ can be a part of two or more superarms, $f(P')$ is included in the superarm with higher reward.

Lastly, a specific quest proposal P needs to be chosen to present to the player, since we abstracted the quests from the arms and superarms. The bandit algorithm selects the next superarm S using UCB, where S consists of a set of arms $f(P)$ at time t. All arms are assumed to be equivalent, so all arms from within a superarm are selected at random with equal probability. Let $f' : \tau \rightarrow q$ be the inverse function to f, where f' inputs a quest type τ and outputs a quest q. Since f is many-to-one, f' is one-to-many. The AI director design assumed that all quests are of the same quality, so the specific quests are randomly selected from all quests of a given type. To present a set of quests to the player, apply f' to each τ in p, shown in Equation 3.

$$\begin{aligned} f'(f(P)) &= \{f'(\tau_1), f'(\tau_2), ...f'(\tau_k)\} \\ &= \{q_1, q_2, ...q_k\} \end{aligned} \tag{3}$$

We chose to use an abstraction instead of a naive implementation of CMAB to reduce the computational complexity, which is necessary due to the computational limitations of deploying the algorithm live in a video game. A naive application of CMAB would assign each quest to be an arm, and the proposal to be a superarm. Let n be the total number of quests. $|\mathbf{P}| = \binom{n}{k}$. This results in a large set, where the size is most affected by n. For example, in Nightingale there is 101 unique quests, and 3 are shown to the player at the same time. This would result in $|\mathbf{P}| = \binom{n}{k} = \binom{101}{3}$ = 166,650. This is compared to the size of the abstracted set using quest type, where $|f(P)| = \binom{|T|+k-1}{k}$ = 20.

5 Methodology

We conducted a large scale human subject experiment, which was run within the larger context of an Inflexion Games playtest. Experiment explicitly refers to the experiment that was designed to evaluate the AI director. Playtest refers to the wider context, which includes goals outside of evaluating the AI director.

5.1 Playtest Goals

There were many different aspects of the game that were simultaneously evaluated during the playtest.

Game Studio Goals Inflexion Games had several high level studio goals, but there were two that were most relevant to the AI director results: evaluation of studio-level playtest processes, and an evaluation of the main game loop to understand how fun the game was.

AI Director Goals The AI director directly supported the two Inflexion Games goals. First, the AI director was evaluated using an AB experiment, and it was the first time the studio had run an experiment of this type. This helped developed studio-level playtest processes.

Second, the AI director was used to help evaluate the main game loop by being directly integrated into the progression. There were several research questions that were proposed to help understand how the AI director was affecting the player experience, but due to space we only discuss the most relevant one: Was the AI director algorithm more satisfying than a random algorithm? Our hypothesis was that the AI director provides a better player experience because it was able to learn gameplay preference to provide preferred content.

5.2 Experiment Design

To test the AI director, an AB test was conducted, where players were randomly assigned CMAB or random. Though a random algorithm can also be considered an AI director, for the rest of the paper we denote the AI director as the CMAB algorithm. A random algorithm was chosen for comparison because it was considered to be industry standard by

designers at Inflexion Games. To determine the effectiveness of the AI director, we collected quantitative and qualitative data. Due to space we will only discuss the data relevant to answering our research question and evaluating our hypothesis from Section 5.1.2. The quantitative data collected was as follows: the number of accepted quests, the number of proposed quests, and the total playtime in minutes.

The number of accepted quests and the number of proposed quests were used to calculate the acceptance rate: $\frac{Number\ of\ Quests\ Accepted}{Number\ of\ Quests\ Presented}$. A higher acceptance rate indicates that the player was presented with quests that they were more likely to accept. Total playtime was used to represent the overall satisfaction. If a player plays a game for longer, it could indicate that the player enjoyed the game more.

There are two types of qualitative data that were collected. The first was an optional exit survey that was prompted when players quit the game. The second was optional feedback data. If players pressed F2 during gameplay, they could write feedback. This functionality was advertised in pre-playtest instructions, but was not indicated to players while they were playing.

The exit survey asked several questions about gameplay and the overall experience. Due to space, only the six relevant Likert questions are included.

1. I felt like there was a good variety of quests
2. I felt like there were too many repeated quests
3. I felt like I abandoned a lot of quests
4. I felt like I finished most of the quests I accepted
5. It was easy to find quests that I wanted to do
6. It was frustrating to find a quest that I wanted to do.

Questions 1 and 2 allowed us to determine if there was a satisfying variety of quests being provided to the player. Questions 3 and 4 allowed us to determine how many quests the players felt they were completing. Questions 5 and 6 allowed us to determine if players felt like they were able to find quests that they wanted to complete. These questions were specifically developed for this experiment.

Friends and family of the developers were invited to play the game over the course of one weekend. The playtest started on Friday and closed on Sunday for a total of 52 hours. Developers were allowed to play with their friends and family during this time. Friends and family players did not know that the AB test was being conducted, and would not know unless a developer told them. Additionally, only developers that were directly involved in implementing the AB test knew that the test was occurring, so the chance that developers told other players is low.

6 Results

252 non-developer individuals played the game at least once during the playtest. Due to space, only results that are relevant to the AI director are included in this paper.

6.1 Playtest Limitations

This was the first playtest of this scale conducted by the studio, so there were a few technical limitations that affected the results of the experiment. First, there were errors in assigning the algorithm, where some some players switched between being assigned the AI director algorithm and the random algorithm. This affected 44 player profiles which left 208 usable player profiles. 75 profiles were assigned the AI director algorithm, and 133 profiles were assigned the random algorithm. Second, there was an inability to connect the algorithm to most of the qualitative data. From the exit survey, only 6 AI director profiles and 10 random profiles were collected. From the feedback, none of the responses could be connected to type of algorithm they were assigned in the AB test.

Third, the players were all friends and family of the developers. There could be biases in how they perceived the game based on these personal relationships. Fourth, players were allowed to play the game with developers. This could have directly impacted results such as playtime, where players could have a played longer because they wanted to play with a developer, rather than from enjoyment of the game. Developers are excluded from all data, but could potentially affect all collected data.

Last, the game was actively in development. Certain features were not developed to their intended level, or were missing entirely from the game. This directly conflicts with the assumptions in the AI director design, and is one possible explanation for our inconclusive results.

Algorithm	*Median*	*Average*	*Std Dev*
AI Director	117.0	211.3	251.0
Random	171.0	262.7	247.4

Table 2: Total playtime (min) by algorithm

6.2 Quantitative Results

Only quantitative data that most strongly relates to the AI director performance is included in this paper, due to space. Table 2 shows the median, average and standard deviation (std dev) of total playtime. The std dev is large due to the distribution of play times. Most of the players played for less than 30 minutes, but there was a section of players who played the game for a significant amount of time.

We used a Mann-Whitney U test / Wilcoxon's Rank Sum Test to compare the total time spent in game between the two different quest algorithms (W = 4076.5, p = 0.02893). The AI director group was found to have spent less time in game (AID mean = 211.25 minutes, SD = 250.99) than the random group (random mean = 262.68 minutes, SD = 247.41).

Table 3 shows the acceptance rate for each algorithm. The average acceptance rate of the AI director players is 12%, and the average acceptance rate of the random players is 9%. We used a Mann-Whitney U test / Wilcoxon's Rank Sum Test to compare quest acceptance rates between the two different quest algorithms, which found that there was no statistically significant difference between the two groups (AID mean = 0.285, SD = 0.281; random mean = 0.275, SD = 0.289; W = 5276, p = 0.4891).

Algorithm	*Harvest*	*Build*	*Craft*	*Kill*
AI Director	14%	13%	9%	10%
Random	16%	8%	7%	5%

Table 3: Acceptance rate by algorithm

	Strongly Disagree	*Disagree*	*Neutral*	*Agree*	*Strongly Agree*
Q1*	2	1	1	1	0
Q2*	0	1	3	0	1
Q3*	0	1	0	4	0
Q4*	1	1	1	0	2
Q5	1	1	2	2	0
Q6	0	1	1	3	1

Table 4: Results from the survey Likert Questions for AI director players, star (*) indicates that only 5 out of 6 players responded to this question in the survey

6.3 Qualitative Results

The feedback results are omitted from this section since we cannot attribute the player's assigned algorithm with their feedback, and therefore cannot draw any broad conclusions.

Survey Results Q1-6 refer to the questions 1 through 6 from Section 5.2. The results from the Likert questions for AI director players are shown in Table 4, and the results for the random players are shown in Table 5. To compare the results from the Likert questions, each response was assigned a number. Strongly disagree was assigned a 1, and strongly agree was assigned a 5. We compared the median value of each question, shown in Table 6. The comparison shows similar sentiments from both players, which we discuss further in Section 7.3.

7 Discussion

This was the Inflexion Game's first time running a playtest of this scale. As mentioned in Section 6.1, a few crucial data links failed. From the studio perspective these operational failures were an opportunity to improve the process for future playtests. The main game loop was clearly evaluated to inform future design. Though the results for the AI director experiment are inconclusive, the playtest was a success.

	Strongly Disagree	*Disagree*	*Neutral*	*Agree*	*Strongly Agree*
Q1*	1	1	2	5	0
Q2	1	0	3	6	0
Q3*	1	1	2	2	3
Q4*	1	3	1	4	0
Q5	4	3	2	0	1
Q6	0	1	3	3	3

Table 5: Results from the survey Likert Questions for random players, star (*) indicates that only 9 out of 10 players responded to this question in the survey

Algorithm	Q1	Q2	Q3	Q4	Q5	Q6
AI director	2	3	4	3	3	4
Random	3.5	4	4	3	2	4

Table 6: Comparison of the median value for each Likert question, with 1 being strongly disagree and 5 being strongly agree

From section 5.2.2, we asked the question "Was the AI director algorithm more satisfying than a random algorithm ?". There are three possible outcomes for this question: that the AI director provides a better player experience, the random provides a better player experience, or players perceived similar player experiences. Because the data does not point out a single clear answer to this question, we discuss the evidence that supports each outcome.

7.1 AI Director was a Better Player Experience

The results from the acceptance rate support the theory that the AI director provided a better player experience, shown in Table 3. Though the Mann-Whitney U test showed that there was not a statistically significant difference, we still believe that this shows a slight preference for the AI director. A higher acceptance rate indicates that that players were presented with quests that they wanted to do more often, which indicates that the AI director more successfully provided quests that players preferred.

7.2 Random was a Better Player Experience

The results from the total playtime support the theory that the random algorithm was preferred. Both the average and the median total play time are longer for the random players, and the Mann-Whitney U test confirmed that this is a statistically significant difference. We provide two possible explanations for these results.

The first explanation is due to the type of reward schedule each algorithm provides. Assume that a player is rewarded when they find a quest they prefer. The AI director attempted to provide consistent reward, where every time the player interacted with the job board they would see quests they preferred. The random algorithm provided an intermittent reward schedule (Cameron and Pierce 1994), where players only found quests they preferred some of the time. Intermittent reward schedules are known to be powerful motivators, which could have provided a better player experience (Luo, Yang, and Meinel 2015; Griffiths 2010).

The second explanation is a mismatch between the assumptions of the AI director and the actual game that was being played. The AI director was designed with the final version of the game in mind, rather than the version of the game in the playtest. The AI director assumed that all gameplay types were equally developed and fun, such that the players would be able to clearly identify the type of gameplay they prefer. In reality, the kill and craft loops were underdeveloped, and the harvest quests were the easiest to complete. This made it harder for players to determine the parts of the game they enjoyed. Since players might not have

had any gameplay preference, randomly providing quests to players could have provided consistent variety and could have created a satisfying player experience.

7.3 Players Perceived Similar Player Experiences

Due to playtest limitations, it is possible that players perceived a significant difference between each type of algorithm, but the data was unable to capture the difference. Another possibility is that players had different player experiences, but they were not different enough to be significant. We explore the second possibility here.

In the limited qualitative data, the results indicate that the AI director and random players perceived similar player experiences. Many of the Likert questions have the same median value for both algorithms. One explanation for the similarity is due to a negative experience from playing the game. Players felt like the game loop needed work. One player was frustrated enough to submit optional feedback, saying "I am very much not a fan of the current gameplay loop... I have to port back to [the city], go through a loading screen, fast travel, get a new research, and then port BACK to [the canonical realm] just to cash in the quests..."

Players' frustration levels from playing the game may have affected the survey results. Players had trouble divorcing their frustrated experience of trying to complete quests from the experience of accepting quests. In particular, Likert question 6 "It was frustrating to find a quest that I wanted to do", had the same median value of 4 for both AI director and random players, which corresponds to the answer agree. There could have been differing amounts of frustration experienced by players, where AI director or random players could have had a more or less negative experience. However, the Likert results do not show any significant differences.

8 Takeaways as an Academic Working with Inflexion Games

In addition to the lessons learned from the results of the experiment, there are important takeaways we learned from working closely with an industry partner.

8.1 Finding the Right Industry Partner

Video game development is an extremely expensive endeavour, which makes studios that are interested in untested technology rare to find. We were lucky to have an industry partner who sees the value of providing a space for academic research despite the risks.

Inflexion Games was willing to invest over a years worth of development resources in order to design and implement the AI director in their game. They also saw the potential of the AI director to be a powerful influence on the player experience, and decided to showcase it as a part of the main game loop, instead of as an optional part of the game. This shows their incredible support for the technology, and also their willingness to experiment and take risks in order to provide a better player experience. We were extremely lucky to have an industry partner put their faith in both the research and the academic who implemented the research in the game.

8.2 Research in an Actively Developing Game

To include research like this AI director in a game, it needs to be proposed while the game is being developed. The systems that the research should interact with need to be engineered so that they are compatible with the chosen algorithm. Additionally, it does not make sense to propose a potential AI director near the end of a game's development, because many of these systems will already have been engineered in a potentially incompatible way.

Designers at Inflexion Games proposed an AI director, which meant they knew the proper time to include it in the development cycle. This was proposed over a year before the playtest. The AI director was included at an early stage in the development cycle such that the quest system could be designed to accommodate the CMAB algorithm. However, this was about a year and a half into the development of the game, so gameplay and a core loop were semi-established. A game does not need to originally propose an AI director to include one, but if the game is too far along its development cycle, it can be difficult to make the necessary changes.

8.3 Evaluating Research on an Industry Timeline

Within industry, it is common practice for studios to run playtests in order to evaluate different parts of their game. Playtests provide an opportune time to run experiments for academic research, and provide a natural way for the needs of academic research to align with the needs of the company. Studios schedule these playtests in a way that works best for them, and is dependent on the development state of the game. If the game is not ready for outside players, the playtest can easily be moved to a later date. This flexibility makes it difficult to schedule with research deadlines such as conferences or degrees.

For example, the playtest that was ultimately run in 2021 was originally scheduled in 2020, and the playtest was moved several times. The shifting nature of these playtests, and additional uncertainty of when the next playtest might be, makes a collaboration like this impractical for individuals who are on a tight academic timeline.

9 Conclusion

We partnered with Inflexion Games in order to develop an AI director to suit the needs of their game Nightingale. We ran an experiment to compare how the AI director performed against a random algorithm, and the results were inconclusive. Some results indicated that the AI director was preferred, other results indicated that the random algorithm was preferred, and other results indicated that players did not perceive a difference in the experience. To help others interested in similar industry collaborations, we provided insight into the practical limitations of working closely with industry.

Acknowledgements

This work was funded by MITACs and Inflexion Games. Special thanks to Aaryn Flynn, Kris Schoneberg, and Kaelin Lavallee for their extra support in facilitating this partnership between academia and industry.

References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47: 235–256.

Booth, M. 2009. The AI Systems of Left 4 Dead. *Game Developers Conference*, 95.

Cameron, J.; and Pierce, W. D. 1994. Reinforcement, Reward, and Intrinsic Motivation: A Meta-Analysis. *Review of Educational Research*, 64(3): 363–423. Publisher: American Educational Research Association.

Chen, W.; Wang, Y.; and Yuan, Y. 2013. Combinatorial Multi-Armed Bandit: General Framework and Applications. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of Machine Learning Research*, volume 28, 151–159. Atlanta, Georgia, USA: PMLR.

Giannatos, S.; Nelson, M. J.; Cheong, Y.-G.; and Yannakakis, G. N. 2011. Suggesting New Plot Elements for an Interactive Story. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 6.

Griffiths, M. 2010. Online video gaming: what should educational psychologists know? *Educational Psychology in Practice*, 26(1): 35–40.

Harrison, B. E.; and Roberts, D. 2014. Analytics-driven dynamic game adaption for player retention in a 2-dimensional adventure game. In *Tenth artificial intelligence and interactive digital entertainment conference*.

Jennings-Teats, M.; Smith, G.; and Wardrip-Fruin, N. 2010. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 1–4.

Khaliq, I.; and Watson, Z. 2018. The Omni Framework: A Destiny-Driven Solution to Dynamic Quest Generation in Games. In *IEEE Games, Entertainment, Media Conference*, 306–311. Galway, Ireland: IEEE. ISBN 978-1-5386-6304-2.

Luo, S.; Yang, H.; and Meinel, C. 2015. Reward-based Intermittent Reinforcement in Gamification for E-learning:. In *Proceedings of the 7th International Conference on Computer Supported Education*, 177–184. Lisbon, Portugal: SCITEPRESS - Science and and Technology Publications. ISBN 978-989-758-108-3.

McKeand, K. 2022. Nightingale interview – BioWare values in a co-op survival sim with top hats. https://ftw.usatoday.com/2022/04/nightingale-interview-aaryn-flynn. *USA Today*, Accessed on 2022-05-08.

Ontanon, S. 2013. The Combinatorial Multi-Armed Bandit Problem and Its Application to Real-Time Strategy Games. 9(1): 58–64. Number: 1.

Riedl, M. O.; and Young, R. M. 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research*, 39: 217–268.

Spronck, P.; Ponsen, M.; Sprinkhuizen-Kuyper, I.; and Postma, E. 2006. Adaptive game AI with dynamic scripting. *Machine Learning*, 63(3): 217–248.

Sutton, R. S.; and Barto, A. 2020. *Reinforcement learning: an introduction*. The MIT Press, 2 edition.

Thompson, T. 2014. In the Directors Chair: The AI of Left 4 Dead. https://medium.com/@t2thompson/in-the-directors-chair-the-ai-of-left-4-dead-78f0d4fbf86a. *Medium*, Accessed on 2021-12-01.

Thue, D. 2007a. Interactive Storytelling: A Player Modelling Approach. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 6. Stanford University, Stanford, California, United States: AAAI.

Thue, D. 2007b. *Player-Informed Interactive Storytelling*. Master's thesis, University of Alberta.

Thue, D. 2010. *Generalized Experience Management*. Ph.D. thesis, University of Alberta.

Tychsen, A.; Tosca, S.; and Brolund, T. 2006. Personalizing the Player Experience in MMORPGs. In Hutchison, D.; Kanade, T.; Kittler, J.; Kleinberg, J. M.; Mattern, F.; Mitchell, J. C.; Naor, M.; Nierstrasz, O.; Pandu Rangan, C.; Steffen, B.; Sudan, M.; Terzopoulos, D.; Tygar, D.; Vardi, M. Y.; Weikum, G.; Göbel, S.; Malkewitz, R.; and Iurgel, I., eds., *Technologies for Interactive Digital Storytelling and Entertainment*, volume 4326, 253–264. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-49934-3 978-3-540-49935-0. Series Title: Lecture Notes in Computer Science.

Valve. 2008. Left 4 Dead. PC, XBox 360.

Yu, H.; and Riedl, M. O. 2013. Data-Driven Personalized Drama Management. In *Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 191–197.

Yu, K. K.; Guzdial, M.; and Sturtevant, N. R. 2021. Towards Disambiguating Quests as a Technical Term. In *"Proceedings of the Sixteenth International Conference on the Foundations of Digital Games (FDG)"*. ACM.

Yu, K. K.; Sturtevant, N. R.; and Guzdial, M. 2020. What is a Quest? In *Intelligent Narrative Technologies Workshop*.