The Omni Framework: A Destiny-driven Solution to Dynamic Quest Generation in Games

Imran Khaliq[1] and Zachary Watson[2]

Abstract— **Video games have had the capability to generate dynamic quests for years. Such quests are generally generated from tables of values or randomly assigning properties without properly regarding player progress, situation, or preference. This paper proposes a framework, called Omni framework, for creating quests that are dynamic and contextually linked by incorporating player statistics and play-style. The core of th framework is the Destiny Concept - the idea that a player's 'destiny' can be uniquely 'discovered' and followed throughout the game, rather than scripted.**

Index Terms— **Destiny driven, Omni framework, dynamic quest, quest generation, Video games**

I. INTRODUCTION

Procedural generation is a component of video games that has become increasingly prevalent in the past few years, whether it deals with weapons (Borderlands 2 [1]) or an entire universe (such as No Man's Sky [2]), procedural generation has become a viable method of generating content for video games. Quest generation has not benefited from many of the advancements made. However, while large amounts of terrain or infinite variations of weapons can be generated, tasks for the player to accomplish rarely reach a sophisticated level. This paper will propose a framework, called Omni framework, more complicated than the standard "collect x number of bear pelts" example, utilizing theoretical examples to demonstrate interaction between systems of the framework. The Omni framework comprises two main components: data collection and quest generation. The Omni framework observes data in three different categories: Player, Non Player Character, and World. Quest generation happens at two levels: basic quests and super quests. Basic quests are generated by the basic quest system. There are two main goals for the basic quest system. The first goal is to identify the skill level (beginner, intermediate, expert) of the player by observing how quests were handled by the player. The second goal is to identify the player's style of play (or play-style) by observing if the quests were played aggressively or non-aggressively. Super quests are generated by the Destiny system. The main goal for the Destiny system is to gradually uncover the destiny of the player by authoring the storyline for the player. The overall aim of the omni framework is to create an individualised experience for the player in which both reward and quests are fit to the player's play-style, skill level, and contextually linked to the destiny of the player.

The rest of the paper is organized as follows. Section II discusses previous major frameworks for dynamic content generation. Section III introduces our proposed framework and outlines its main components. Section IV, V, VI, VII, and VIII describes the Player system, NPC system, World system, Basic quest system, and Destiny system respectively, in detail. Section IX discusses improvements of the Omni framework over other frameworks. Finally, Section X concludes the paper.

II. RELATED WORK

In this section, we discuss three main approaches to the dynamic generation of content in video games.

A. The Grail Framework

The Grail Framework's [3] primary aim in quest generation is "completion through a goal" set by the designer rather than through a series of player's actions. By creating "completion states", the Grail Framework creates "end-points" for the player to achieve, and does not acknowledge that how players achieve such end-points. Our proposed framework, Omni framework, aims to go one step further. In addition to taking into account the world state, the Omni framework, analyses the statistics generated by the player's style of play so far. If a player has been favouring a particular quest solving style (such as ranged combat) quests will be generated that suit the players preferences. This emulates the style of the Grail Framework, while aiming for a more statistically powered approach. This approach will be discussed further in Section IV.

Unfortunately, the actual structure of the Grail Framework, in particular GrailGM, has a few issues that restrict it from offering more dynamic quest design. The issues largely stem from the quest definitions being narratively defined, rather than logically based. In the example of breaking up two characters that GrailGM gives, the option for "Violet and James are planning to elope" as the failure state but even antagonistic (that is, counter to the quest design.). This indicates that the system is intended to support quest development particular to a designer's desires. The "elope" state being considered as a failure state is counterintuitive to the design of the system. Because the intention here is to offer freedom and interesting quest design. A more effective fail state would be the death of all of the characters involved, as the quest is now unable to be completed (due to lack of a target to turn in the quest to.)

[1]I. Khaliq is with the Department of Game Development, Media Design School, Auckland, New Zealand imran.khaliq at mediadesignschool.com

[2]Z. Watson is with the Department of Game Development, Media Design School, Auckland, New Zealand zachary.misra at mediadesign.school.nz

The Grail Framework found it simpler to write quests for specific people within the game rather than creating generic characters slotted into roles dynamically. The database provided by the Grail Framework aims to go even further to create nodes for designers to quickly concept and create quests. However, this runs counter to the dynamic intentions behind the framework.

B. Hierarchical Generation

Hierarchical Generation [4] proposes a method for the generation of dynamic quests that are logically based. The method uses the components - “planning, execution, and monitoring”, to generate dynamic ordering of objectives. Furthermore, the method also utilises “re-authoring” to continuously generate the current task based on how the task progresses. As the quest goals change (such as an NPC dying) overall goal of the quest changes. The example that is given results in the goal “Save wife” being changed to “Escape house” if the character’s wife passes. Hierarchical Generation allows for multiple ways that these sub-quests (such as saving the wife) to impact the overall result of the quest. The “narrative context system” in Hierarchical Generation avoids creating narrative conflicts. For example the system does not suggest to the player to go to the hospital if the player is already at the hospital.

The issue with Hierarchical Generation is the limitations that it possesses. Because quests need to be defined with elements crucial to the objectives, also, their dynamic nature is lost before options are even created. While nondeterministic elements allow for repeated playthroughs of quests, with different results, the player is able to experience the same result every time, as long as they follow the same interactions. As opposed to the Grail Framework, which adds degrees of social interaction that are modified based on the players opinions, the quests can be repeated in the same way without fail every time.

C. The Ethos Framework

The Ethos Framework [5] proposes a concept titled Contextually linked dynamic quests, which is an approach that has the overall goal of generating a quest that makes “Logical sense” utilising the four components - “Character, World State, Story, and Library”. The overarching story of the game is static. The story events that occur in the context of the Ethos Framework serve as data to be passed into the quest processing. For example, if a quest in the main narrative involves the death of an NPC, that NPC will no longer be involved in the dynamic quests. This allows the quests to maintain the goal of making “Logical Sense”. The Character component ties into this, going even further than the Grail Framework with character associations and relationships with the player. This occurs with the possibility of a “player’s nemesis” being generated through the dynamic quests. This generation of an artificial “nemesis” that is not predetermined at the start of the game creates an extra attribute for the player to become invested in. The player character’s interactions with other characters form the core of the Ethos Framework, as it quantifies player relationships. The utilisation of “memories” as a concept is an element of this, as it influences dialogue options or choices later on in the game. As opposed to the Hierarchical or Grail proposals, the Ethos Framework utilises these interactions to mould the core narrative. This leads to the aforementioned “nemesis” concept, as the interactions can modify NPC relationships with the character - for better or worse.

The Library component that the Ethos Framework conceives provides diverse content for creating “mini-quests” to make the quest generation more dynamic. The “Monster Problem” presented in “Figure 11” and “Figure 12” in [5] could be created through players wiping out a particular type of monster, for example. This level of complexity goes deeper than the Grail Framework, offering varied approaches depending on the current world state. In quests where the player does not interact on a social level, the Grail Framework fails to achieve the same level of complexity as the Ethos Framework does. In a similar way, the lack of reliance on specific items - such as the “antidote” makes the Ethos Framework more complex than Hierarchical Generation, as well. Rather than creating simple associations with objects and the quests, the Ethos Framework goes further, creating a series of possibilities within nodes. Then, the nodes are combined with contexts to create interesting combinations. For example, the framework describes the solutions to a wolf infestation can be many. Training, killing, or scaring away wolves are just a few of the examples given. The Ethos Framework is designed to be easily customisable. The Library component allows for database entries in particular fields to be created and slotted in efficiently and easily, allowing quest resolutions and components to be generated dynamically via association.

III. The Systems within the Omni Framework

Now, we define our proposed framework for the generation of dynamic quests in video games. We call it Omni framework. There are five systems that interact within the Omni framework. They are as follows:

1) Player System,
2) Non Player Character System,
3) World State System,
4) Basic Quest System,
5) Destiny System.

The Player, NPC, and World State systems comprise the main data collection component of the framework. The Basic Quest and Destiny systems comprise the questing generation component of the framework. The core principle behind the Omni framework is to deliver an experience to the player that is customized to their play-style and also offers quests in a dynamic way. Part of this is developing systems that allow the player to solve problems that they enjoy, rather than ones created specifically by the developer. As discussed in [6], player characters are unique in digital media. The personal desirability that is discussed in [6] is a core component of player understanding. The more that a player personally identifies with a characters role, the more likely they are to enjoy their experience. Moreover, a player who finds war abhorrent would not identify easily with a combat role. In the following sections, we will discuss the five systems of the framework in detail.

IV. Player System

The Player System is the first of the three components, and arguably the most important. The system aggregates data from all of the player's actions. This includes combat, discussion and interaction with Non Player Characters, and whatever other activities the player may perform. The goal of the Player System is as a data collection system for quest generation. The player system has a primary focus on gathering data relevant to a player's play-style, which is broken down into two major sections: aggressive, and non-aggressive.

Aggressive actions include primarily combat actions, as well as dialogue options that could be considered confrontational. The goal with gathering aggressive data is to tailor rewards and challenges specific to the way that the player wants to play. This avoids the issue where the player receives an item that may not be specific to their play-style. According to [7], a valuable reward is significantly higher in terms of quality or usefulness. Moreover, [7] also suggest "withholding details" about rewards to increase player's interest in them. The Omni framework takes into account suggestions in [7] and proposes quest rewards not only at the end of the quest but also according to how the player has completed the quest.

Non-Aggressive actions primarily deals with the player's actions relating to characters as to how they interact with them. Specific conversation choices generate a considered "profile" of the player, which aids in generation of quests later on. If the player chooses to solve quests in a particular fashion, such as resolving conflicts peacefully, then the data will reflect that. This preference for non-aggressive action will be reflected later on, when the player progresses further. The player may receive rewards, or unlock new dialogues and quests from different factions, depending on their preference of action type. This is not an exhaustive list; many different player actions and NPC interactions are envisioned in this area of the system.

The player system keeps track of player-involved activities, with knowledge of:

- Quests (both pre scripted and generated)
- Character interactions (positive/ negative/ aggressive/ passive/etc.)
- Areas visited within the game world
- Current progression relative to the world
- Tools available to the player
- Current tool/weapon use statistics of the player.
- Current player preference based on combat/non-combat.

V. Non Player Character System

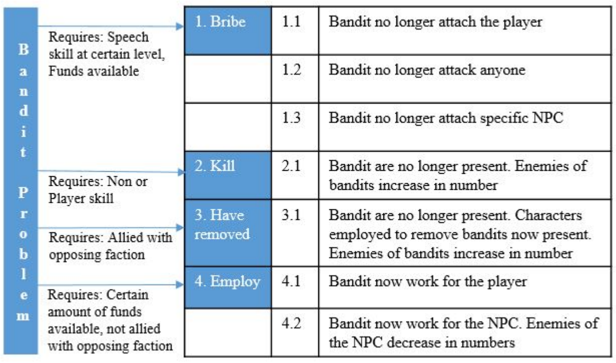
The NPC system takes the interactions between players and NPCs and utilises it when generating quests. Specific dialogue choices, NPC relationships with the player, and particular faction tasks completed are taken into account. For example, if the player chooses to interact with one particular character a lot, that character's allies may be more inclined to offer the player more tasks to complete. Alternatively, characters that oppose that character may become less 'friendly'. This leads to emergent 'competition' between NPCs for the player's allegiances, or may result in hostility from the NPC the player chooses to not befriend.

The NPC system takes information from NPCs in the area where the main quest is generated and altered, which includes:

- Previous interaction(s) with the player
- Current state of the area that the NPC occupies
- The NPCs attributes (health/ equipment/ role/ affiliation etc.)
- How the current world state affects the NPC?

VI. World State System

The world state system includes data such as quest locations - start and end points - as well as the currently regional faction, and the region's attitude towards the player. The world state also takes into account the current state of each NPC in the region, which directly affects the type of quests offered. For example, if the player takes a quest in a region with lower socio-economic levels, the quest rewards are unlikely to be financial, or as extravagant as richer areas. The goal with the world state system is to make different regions of the game world reward quests in a different way, while providing logical rewards.



Requires: Speech skill at certain level, Funds available	1. Bribe	1.1	Bandit no longer attach the player
		1.2	Bandit no longer attack anyone
		1.3	Bandit no longer attach specific NPC
Requires: Non or Player skill	2. Kill	2.1	Bandit are no longer present. Enemies of bandits increase in number
Requires: Allied with opposing faction	3. Have removed	3.1	Bandit are no longer present. Characters employed to remove bandits now present. Enemies of bandits increase in number
Requires: Certain amount of funds available, not allied with opposing faction	4. Employ	4.1	Bandit now work for the player
		4.2	Bandit now work for the NPC. Enemies of the NPC decrease in numbers

Fig. 1. Different solutions to bandit problem

The figure above indicates a typical example of a quest available. The 'Have removed' option, for example, requires the player to have enough of a reputation with another faction, opposed to the bandits. If the player chooses to solve the majority of bandit problems with violence, nonviolent options will become less frequent. The results of other quests the player has completed may also affect quest generation locking off a particular path for a faction, or opening up new quests, for instance. Each of these midpoint 'process' nodes appears and disappears, according to the player's actions in similar quests. Each of the end states has requirements specific to the end state.

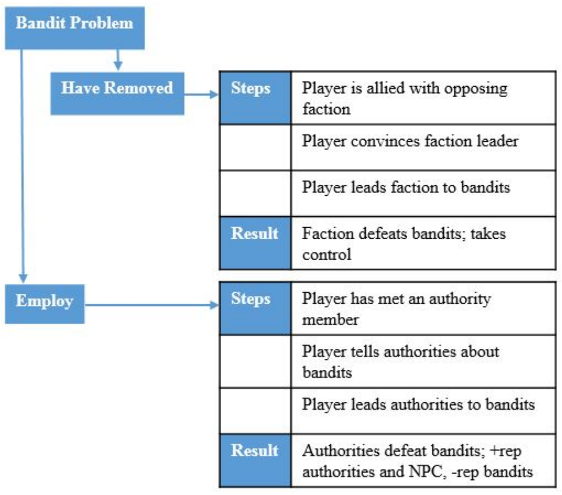

Bandit Problem
Have Removed
Steps
Player is allied with opposing faction
Player convinces faction leader
Player leads faction to bandits
Result
Faction defeats bandits; takes control
Employ
Steps
Player has met an authority member
Player tells authorities about bandits
Player leads authorities to bandits
Result
Authorities defeat bandits; +rep authorities and NPC, -rep bandits

Fig. 2. Two solutions to bandit problem

The Figure 2 shows two of the four solutions to bandit problem. The 'Have removed' option ties up the quest nicely, with no further progression, while the 'Employ' resolution state opens up more options for the player. The unlocking of 'Bandit Quests' could be in the form of antagonizing the same NPC that provided the player with the initial quest, for example. The options taken all contribute to the overall data for the player. Taking the 'Employ' option would increase the player's non aggressive quest completions. This may mean that later on, other bandit groups may be more likely to join the player. This is due to the player's previous success with reasoning with bandit groups.

The world state system takes information from the current state of the game world, including:

- Areas within the game world
- Interactions that the player has had with regions
- Current state of the NPCs in each region
- Quests (scripted and generated) that the player has completed in each region

VII. Basic Quest System

In order to avoid issues of a lack of data early, the Omni framework is intended to work around quest scale. Initially, the Omni framework generates simplistic quests that can be very simply broken down. The ratio of completion states will be 50/50 every time, in terms of the player being able to complete the quest. As players complete more quests, and more data is available, quests of greater complexity become available. This progression mirrors the character's progression within the world, and creates a natural feeling of advancement in the game. While these initial quests may seem simplistic and repetitive to the player, they are required in order to start generating more complex quests.

The process of quest generation takes all of the data provided to it by the three core systems, and when generation begins it takes the data, linking it contextually, to create a quest. The quest generation process is done by taking the player's preferred completion states for questing, and selecting a random amount of them. This random amount is then populated with the options proportional to the non-aggressive/aggressive completion rate. The ways that the player solves quests (killing/bribing/joining/employing etc.) are then taken into account while generating quests, with the aim of providing as many options as possible that the player will enjoy. An example of one of the more simplistic quests can be seen in Figure 3.

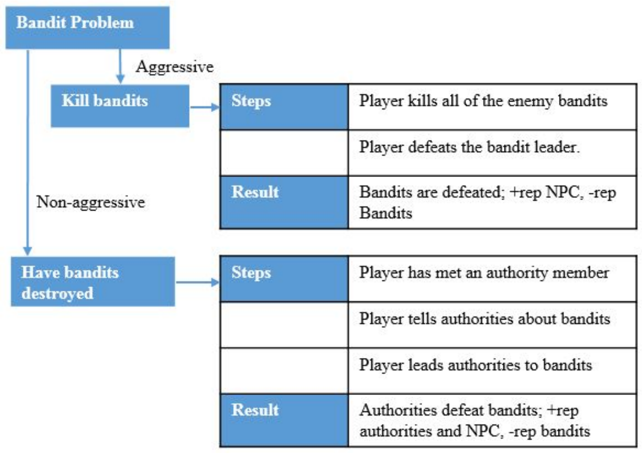

Bandit Problem
Aggressive
Kill bandits
Steps
Player kills all of the enemy bandits
Player defeats the bandit leader.
Result
Bandits are defeated; +rep NPC, -rep Bandits
Non-aggressive
Have bandits destroyed
Steps
Player has met an authority member
Player tells authorities about bandits
Player leads authorities to bandits
Result
Authorities defeat bandits; +rep authorities and NPC, -rep bandits

Fig. 3. Aggressive and Non-aggressive play-style to bandit problem

The purpose of defining actions so early on as aggressive and non-aggressive is to give the player options early on. Players early on do not traditionally have as many skills available, and as a result may be constricted by speech or talent checks. To combat this issue, requiring the split to be 50/50, at least initially, avoids the issue of biasing the data in one direction initially. As the player's skills increase, the options (as well as the ratio) will begin to vary, depending on the player's actions in the game.

VIII. Destiny System

Much like Paul Tillich's belief that Destiny and Freedom are a 'basic polarity' [8], the Destiny System is designed to offer a structural basis for the Omni framework to build on, while also allowing for that basic structure to be generated by the player. The Destiny System is designed to slot in alongside the Omni framework, and forms the other half of the framework's entirety. The premise of the system is simple: to create a methodology of determining the player character's overall 'destiny'. This destiny is formed over data compiled mostly from the other systems in the framework. Destiny goals can be thought of as 'super-quests', and are revealed to the player over time. The aim behind the Destiny System is to create a more personal, integral goal that the player can identify with. The key to this is the gradual unraveling of the player character's destiny. If players uncover their destiny by themselves, they are more likely to "safely embrace" [4] the circumstances.

The Destiny System in the example illustrated in Figure 4 is intended to mirror the classic 'RPG destiny', where the player is 'chosen' to fulfill a particular task. Rather than

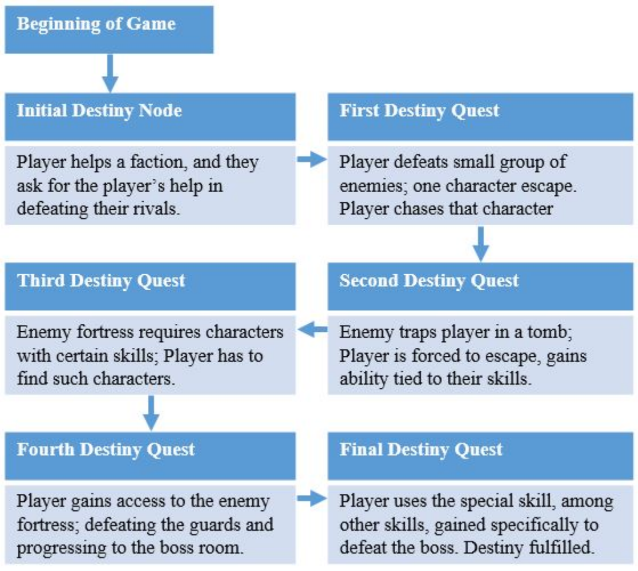

Beginning of Game
Initial Destiny Node
Player helps a faction, and they ask for the player's help in defeating their rivals.
First Destiny Quest
Player defeats small group of enemies; one character escape. Player chases that character
Second Destiny Quest
Enemy traps player in a tomb; Player is forced to escape, gains ability tied to their skills.
Third Destiny Quest
Enemy fortress requires characters with certain skills; Player has to find such characters.
Fourth Destiny Quest
Player gains access to the enemy fortress; defeating the guards and progressing to the boss room.
Final Destiny Quest
Player uses the special skill, among other skills, gained specifically to defeat the boss. Destiny fulfilled.

Fig. 4. The destiny progression

fulfilling a predefined one in a predefined manner, however, the Destiny System offers an alternative. In this example, the skill gained in the Second Destiny Quest is integral to the destiny completion. This choice is dependent on the way that the player has played up until that point in the game. As this type of system would typically be employed in a Role-Playing Game, it can change after it is generated. Depending on the players actions within quests, players may alter or shape their destiny with their actions. While destiny is essentially the 'basic polarity' to freedom, it does not mean the player is unable to shape it as they desire. The "offered role" [4] that the player takes on in the context of the narrative is more compelling, due to the players participation. Destiny in the Omni framework only follows Tillich's concepts as far as cementing it in the narrative context.

The Destiny System generates the components of the main quest, which includes:

- A 'Nemesis' type character.
- An object of worth to 'return' or 'restore'.
- Specific enemy factions or types to remove.
- Types of quests (violent/nonviolent).

The components listed above are merely indicators of what the systems could contain, and are not by any means absolute.

A. *Improvements as a result of the Destiny System*

The Destiny System is an alternative way of creating the typical 'main quest' of an Role-Playing Games. However, unlike many other Role-Playing Games, the player is actually encouraged to 'role-play', and the game will reward them for it. If the player wishes to play a completely non-aggressive style, the Destiny System will adjust to that. Resolving issues peacefully and with minimal consequences is a viable playstyle, due to the strategies that the game uses to generate quests. Where most Role-Playing Games require the player to defeat the boss by completing an 'Epic Battle', or fighting off waves of enemies, the Destiny System offers an alternative. The system avoids the disconnect that a player may experience through their playthrough - a player who prioritises summoning creatures or debuffing enemies will encounter bosses that their spells are effective against, for example. Monolith Productions' title - Shadow of Mordor [9] - utilises their "Nemesis System" to make enemy minibosses invulnerable to certain attacks, such as specific dagger or bow attacks. This is done to compel the player to change up their playstyle, as the title is an action-focused game. The Destiny System seeks to avoid this; instead offering escalating challenges within preferred skill sets - such as speech challenges or accuracy challenges - which encourage players to explore the skill that they have already put time into. The player should receive challenges specific to their style of play, allowing the world to react in the "authentic ways" [6] that Hefner *et al.* discuss. This avoids a narrative and mechanical disconnect.

Furthermore, the system allows for experimentation and entertainment before the player jumps into the main quest of the game. Rather than being presented with the main quest from the start, the player is allowed to discover it on their own. Tillich's concepts of destiny include the idea that freedom "participates in shaping my destiny" [8]. Tillich's explanation of freedom coexisting and generating destiny is the core belief of the Destiny System which is shaped and altered by the players (free) actions within the game world. Tillich's concept of freedom can be broken down to the normal view of RPG gameplay - the equivalent of freedom, while a typical 'main quest' represents Tillichs own concept of destiny. The Destiny System allows for further identification with the player character, and allows for a title utilising such a system to "maximise player enjoyment" [6].

IX. Omni Framework Improvements

In this section, we will discuss Omni framework improvements over the aforementioned frameworks.

A. *The Grail Framework*

The Omni framework deals with the shortcomings of the Grail Framework by refusing to define quests as narrative elements. Instead, the components of the quests are generated as a result of the players actions, instead of being predetermined. The end-states that the Omni framework creates are conceptually similar to the Grail Framework, but are narrower and more specific in focus. Taking into account a players preferences for a particular method of solution, the Omni framework achieves quest generation in a way that responds to the players actions. The major differences come as a result of the method of structure for each system. Because the Grail Framework is tailored towards designed quests that change dynamically, the end results are finite. The Omni framework, by contrast, allows specifically tailored quests that respond and are geared towards a players particularly preferred playstyle. The Omni framework adapts the node-based structure that is part of the Grail Framework's "re-orderable storylines" idea. The focus on character and narrative is disregarded in favour of creating a structure that focuses on the player, rather than the narrative. Rather than changing the quest states during the quest, the Omni framework aims to create unique paths and interactions before the quest starts.

B. Hierarchical Generation

Hierarchical Generations node-based, logically structured system was a major source of inspiration for the development of the Omni framework. The constant changing of the over-arching goal was retooled into the redevelopment of quests from a player standpoint. Rather than constantly updating the quest structures, the possible nodes is continually updated depending on the players actions. Dynamically changing quest possibilities is an interesting component that is somewhat included in the Omni framework itself if the player fails a check or a task, then that path could become locked, for example. Expansion is made to the world state component to make it a core component of the experience, rather than using it only as a manager for quest data. Compared to the Hierarchical Generation's world state, the Omni framework's state also contains data that affects the type of quests offered, as well as whether or not they can be offered at all. The Omni framework aims to avoid the finite solutions problem that each instance of Hierarchical Generation could create.

C. The Ethos Framework

The Ethos Framework's focus on "narratively contextual dynamic quests" mirrors the focus on mechanics in the Omni framework. Where the Ethos framework's goal is to create quests that make sense contextually, the Omni framework aims to achieve that through mechanical interaction and feedback for the player. The Ethos Framework's use of a library as a storage unit and aggregator for the large amount of data in other systems is noticeably absent for the Omni framework's systems. This is because of the nature of the way components are generated. As the player progresses through the game, the "generic problem" that the library has is now replaced by issues directly caused by the players actions. Rather than taking particular problems and generating ways for the player to solve them, the Omni framework takes the tasks the player has already accomplished, and iterates over them to create escalating levels of challenge. The goal of the Ethos Framework is to make the quests narratively focused; the goal of the Omni framework is to make them mechanically interesting and rewarding for the player. This is done simultaneously with a goal of also delivering narrative feedback that the player feels they are impacting.

X. CONCLUSIONS

The strategy when developing the Omni framework was to create a method of developing quests dynamically, specifically at runtime, that catered to player data. The implementation of the Omni framework takes the three aforementioned frameworks and focuses on a particular element of game design - player mechanical interaction. The framework focuses on the actions that the player makes, and adjusts the ways that quests are generated to keep the player interested. The generation of quests that challenge the player in relevant ways creates a unique style of challenge for the player. In order to fully examine the potential of the Omni framework, a testbed would need to be created. This testbed would require vast amounts of content in order to test on a meaningful level. For practical implementation of the Omni framework, an imitation of the structure would have to be set up (Player $\rightarrow$ NPC $\rightarrow$ World State) to experiment with. Then, example quests could be generated by taking predefined skills, NPCs, and World states, and permutating them to create different quests. These generated quests are then marked as completed, which would affect the predefined data. Further iteration over this data set would create more interesting and dynamic permutations, as the data sets grow larger and larger.

REFERENCES

[1] *Borderlands 2.* [Digital Game], California, US: 2K Games, 2012.

[2] *No Man's Sky.* [Digital Game], United Kingdom: Hello Games, 2016.

[3] A. M. Sullivan, "The Grail Framework: Making Stories Playable on Three Levels in CRPGs," Ph.D. thesis, Santa Cruz, California: University of California, 2012.

[4] E. S. d. Lima, B. Feijo and A. L. Furtado, "Hierarchical generation of dynamic and nondeterministic quests in games," in Proc. 11th Conference on Advances in Computer Entertainment Technology, Funchal, Portugal, 2014, pp. 24-33.

[5] J. Dodunski and D. I. Khaliq, "The Ethos Framework: Generating Contextually Linked Dynamic Quests," *The Journal of Creative Technologies*, vol. 6, pp. 1-18, 2017.

[6] D. Hefner, C. Klimmt and P. Vorderer, "Identification with the Player Character as Determinant," in Endertainment Computing - ICEC 2007, Shanghai, China, 2007.

[7] H. Wang and S. Chuen-Tsai, "Game Reward Systems: Gaming Experiences and Social Meanings," in DiGRA, Utrecht, 2011.

[8] P. Tillich, "Systematic Theology," City of Chicago: University of Chicago, 1967.

[9] Middle Earth: *Shadow of Mordor.* [PC-DIGITAL], Kirkland, Washington: Warner Bros. Interactive Entertainment, 2014.