Procedural Character Generation for Narrative Games

Larry LeBron
UC Santa Cruz
1156 High St. Santa Cruz, CA
llebron@soe.ucsc.edu

ABSTRACT

In this paper, I discuss general challenges associated with authoring characters for interactive narrative games. I then highlight some significant issues in this area, including challenges associated with authorship and improvised character development. Continuing, I propose one avenue for addressing these issues: procedural character-structure generation. I then outline the *Personage* prototype system which I created as a first step in demonstrating a potential solution to these concerns. I conclude by discussing *Personage's* merits and limitations, along with potential for future development.

Categories and Subject Descriptors

K.8.0 [**Personal Computing**]: Games

General Terms

Algorithms, Human Factors, Theory

Keywords

Narrative, Story, Characters, Procedural Content Generation, Games

1. INTRODUCTION

This section presents some background on the role of characters in narrative games, along with some key challenges faced in their development. It then introduces the procedural character generation technology I developed to address this issue.

Nearly all narrative games rely heavily on characters and character development to convey their story. Interaction with such characters serves as a crux of gameplay in a variety of genres, including tabletop role playing games, social board games and digital story games. Game characters fall into two general categories, NPCs and PCs, non-player-characters and player-characters, which I will now discuss.

1.1 Authoring Non-Player-Characters (NPCs)

Non-player-characters are controlled by the system, or game master, allowing for different levels of interaction. The deepest NPCs typically feature hand-authored backstories and personas. These characters are designed to develop and arc along with their character abilities. They are core to the development of the game story, which requires them to allow for extensive player interaction. These NPCs are therefore very labor intensive to create, which leads most developers to include only a limited number.

Unfortunately, story worlds tend to feel empty and lifeless when inhabited by only a few characters. This has lead some developers to embrace the idea of a largely devoid worlds, such as those found in *Myst*[1] and *Shadow of the Colossus*[2]. This solution is not widely applicable, and many story game creators flesh out their worlds with shallower NPCs.

In table-top games, these shallow NPCs may be used as window dressing or simple plot-point deliverers. Imagine a classic inn scenario in *Dungeons and Dragons*[3].When the players enter the inn's pub, the Game Master might describe the setting, along with expanded, pre-authored descriptions for an inn-keeper, bartender, and a mysterious stranger. (S)he might add in that the bar is occupied by a collection of local rabble. The Game Master might have extensive personas developed for the highlighted characters, but what happens if a player approaches one of those rabble?

The Game Master might wave the player off with a canned line like "*The drunk stares into space completely oblivious to you.*" On the other hand, the Game Master might try to improvise that character, allowing the player to interact as much as s(he) desires.

This type of improvisation is difficult, and only gifted storytellers are able to handle this type of situation without disrupting player immersion. The best storytellers are able to seamlessly weave the new NPC into the story, perhaps even fleshing them out into a fully realized persona, with abilities and a backstory to match. At the same time, the players might never even interact with the barkeep, for whom the Game Master had pre-authored a rich story and ability set.

This issue is even more challenging to address in digital games. While a human Game Master can improvise a character based on prior knowledge and life experience, a computer has no such intuitive talent. This leads most game developers to laboriously pre-author both core and ancillary NPCs.

Additionally, digital developers must work within a limited system resource budget. In light of this, developers typically restrict the variety and interaction ability present in ancillary NPCs; prioritizing those essential to story progression.

The simplest of these shallow digital characters will merely navigate the space, seemingly unaware of the player's actions. In some cases, they will also perform simple, pre-scripted actions such as rote interactions and behaviors.

More complex systems will imbue these NPCs with limited agency, allowing them to pursue autonomous actions based on limited mental models. Regardless, these types of NPCs will seldom support a deep model for interaction. As all content associated with these characters is hand-authored, developers will also likely wish to reuse it. Therefore, the player may encounter models and behaviors which are repeated numerous times.

In the end, players who come up agains an interaction barrier will experience a lack of agency and immersion. A player who wishes to learn more about that street-sweeper or to go out adventuring with a random messenger is just out of luck, which may lead him/her to question the believability of the game world.

1.2 Authoring Player Characters (PCs)

As the name suggests, player characters are those directly controlled by human gamers. In narrative games, these characters also have stories and abilities. In digital and non-digital games, these characters will either be created by the player, the developer or the Game Master. Regardless, someone is faced with the issue of character authoring before being able to play the game.

Additionally, this strictly pre-authored approach may limit the depth of the created character's backstory. After all, a developer probably won't require a player to write their character's biography in order to begin playing. If there is information that a player cannot learn about his/her character, the player may end up feeling disconnected in their role-play.

Imagine a scenario in which a PC is created with a high level of magical ability. The player might desire an explanation for this skill-set. As before, the developers or Game Master will be faced with either prohibiting or improvising this unpredicted interaction.

1.3 Hand-Authoring and Replayability

In both the case of PCs and NPCs, strictly pre-authored characters also limit the variability of the experience on replay. If a game's main focus is on pre-authored characters, then this key part of the game experience will play out the same every time. Developers can take this into account by authoring a wider selection of characters, but that may be prohibitively labor intensive. In the end, this lack of variability may cause players to limit their time with a game to a single playthrough.

1.4 Problems With Character Authoring

In examining these cases, some common problems appear.

- Pre-authoring characters is a challenging, time consuming task.
- In games with increased player freedom, there is a chance that pre-authored character content will not be experienced.
- Hand-authored character content provides less variability and replayability.
- Pre-authored character interactions cannot allow for maximal player agency.
- Player interaction with pre-authored or shallow characters requires improvised character authoring.

1.5 Is Procedural Character Generation the Answer?

The procedural character generator described in this paper was developed as a first step towards addressing these issues. This generator is able to dynamically create

text descriptions of characters with a simulated life story, as well as abilities and values that correspond with their life experience. In developing this technology, I sought to answer these specific questions in the realm of character generation:

1. Can procedural character generation create interesting characters?

2. Can procedural character generation ease the burden of authoring characters?

3. Can procedural character generation allow for "improvised" characters in digital games?

2. RELATED WORK

In this section, I will discuss some significant research in procedural character generation. I should mention that a good deal of work has been dedicated to procedurally generating character models, animations and other related art assets. These are certainly important components of the larger research problem at hand, but, as they are mainly graphical problems, remain out of scope for this discussion.

Instead, I will be focusing on work relevant to the generation of underlying interactive character representations, which includes character backstories, traits and abilities. Moving forward, I will refer to this as *character structure,* in an effort to separate it from other bordering research areas. I will then discuss how the system I developed addresses a previously uncovered area in this field.

In his 2007 thesis work, Jeff Lininger created a system for character backstory generation, examining whether "simulations of actors with differing traits and needs can create a unique and seemingly logical series of events that players interpret as authentic back-stories."[4] Each of his simulated characters has needs, such as hunger and comfort, which they must satisfy by undertaking various activities. Additionally, his characters have personality traits which influence their activity selection.

This simulation system runs at the initialization of an interactive murder mystery scenario, culminating in the murder of one of the characters. The player is then able to interact with the environment and the living characters, while collecting clues that were generated by the scenario. Although the simulated character representations themselves are relatively shallow, this work succeeds in procedurally generating interactive characters. One of the main shortcomings of the research was that players were unable to uncover some of the underlying motivations for simulated character actions, as they were only informed through vague, narrative text descriptions.

Another interesting work in this domain is Chen et. al.'s system, *Role Model*, which generates story's organized around explicit, formal models of character roles [5]. Their system allows authors to specify story goals by asserting abductive logic constraints, and their research focuses on the story variations that can emerge from such a specification. Their character models are far more complicated than Lininger's, and include roles, traits, and sentiments towards actions along with action models which build off of causal properties and constraints.

Role Model produces believable story output based on causal logic, but the described system was a relatively incomplete prototype. At time of publish, it allowed only three character roles, and produced only static, categorical output that would be difficult to implement in an interactive story setting. Nevertheless, it shows promise as a novel approach in the area of constraint-based character generation.

Mei et. al.'s work on the *Thespian* system focuses on managing story characters in interactive drama[6]. Their system uses autonomous agents to control characters by determining character goals based on a starting goal script. Additionally, the system encodes the character's "personality" into their agent goals. Although the system does rely on a pre-authored script, it is flexible enough to allow for drama that diverges from this plan.

In these points of deviation, *Thespian's* agents ensure that their managed characters stay true to their initial motivations. This is largely possible because *Thespian* was constructed on top of *Psych-Sim*, a deep autonomous-agent management system, developed by Marsell et. al. [7]. This underlying structure allows *Thespian* to focus on drama management and character integrity, without needing to model all of the autonomous agent behavior.

In their evaluation, Mei et. al. admit that this system still requires hand authoring. Nevertheless, they believe that the system alleviates the burden of the authoring process by allowing for higher level declarations Authors implementing thespian can generate compelling characters by encoding their story goals alone, instead of having to specify all of the possible inter-character interactions. Thanks to its deep underlying *Psych-Sim* architecture, *Thespian* characters will be able to adapt to different dramatic situations while maintaining an internal consistency.

Yet another relevant work in this area is Cavazza et. al.'s research in interactive, character-based storytelling[8]. They focused on applying Hierarchical Task Network (HTN) planning to govern character behavior

management in interactive storytelling. Their system requires pre-encoded character plans, and then simulates variable, interactive drama situations as the managed characters attempt to satisfy their goals. The system does support player interaction, but it is limited to simple interactions with items from a disembodied role. The player is not able to control his/her own character in the story.

Cavazza et. al.'s system successfully demonstrates an architecture model which enables compelling and variable virtual character performances. Other researchers have also made progress in this vein of character behavior management, including McCoy et. al. in their work on *Comme il Faut,* an architecture for social interaction simulation[9]. While these systems do allow for robust character generation, they still require some degree a significant degree of character pre-authoring.

Taking a different angle on dynamic character management, Sullivan et. al.'s *Mismanor* focuses directly on deepening PC-NPC interactions in digital role playing games[10]. Obviously, even if a NPC is richly developed, a player can only enjoy this content if is discoverable. Typically, the player can learn about such NPC characters through an in-game interaction system.

Their research attempts to expand the emergent possibilities in these digital interactions, by building off of the aforementioned *Comme il Faut(CiF)* system. Whereas *CiF* was constructed to allow the player to simulate interactions between any two characters, *Mismanor* puts the player in the role of an individual character.

By utilizing *CiF*, *Mismanor* enables a richer type of social interaction with NPCs, allowing the player to experience a more dynamic, two-way conversation than is found in most digital role-paying-games. This system additionally allows for player-character creation, which results in a PC that the system can reason about in the same way it reasons about its NPCs. Although this still requires some pre-authorship of NPC and PC social models, it does provide a useful model for dynamic, playable interaction between such characters.

A final piece of relevant work, which serves as a guiding piece for research in this area, is Riedl and Young's *Objective Character Believability Evaluation Procedure for Multi-Agent Story Generation Systems*[11]. In their work, Riedl and Young present a procedure for objectively evaluating characters in generated stories. Specifically, their work focuses on character intentionality, which they posit is a core aspect of character believability. As they describe it, "Character intentionality addresses the relationship of actions and behaviors to an agent's beliefs, desires, and intentions as well as internal and external motivation."

Their model uses the pre-existing *QUEST* system to evaluate believability of characters present in generated stories. *QUEST* is a human-validated model of human question-answering, which allowed Riedl and Young to effectively determine whether generated stories matched their stated character believability requirements. Their implementation of this evaluation system then allowed them to prune a set of generated stories, presenting only those which passed the believability metric.

In a controlled experiment, readers rated stories that were selected by their system favorably, demonstrating the effectiveness of their model. Their work therefore specifies a valid approach to evaluating the quality of generated characters. Their model is currently restricted to evaluating character intentionality, which is only one aspect of character believability. Therefore, further expansion will be necessary to provide a complete solution to determining character believability. Additionally, their model is limited to static generated story output. Therefore, the system would likely need to be modified to evaluate characters in real-time interactive work.

As is evident, much of the focus in this area has been on managing dramatic, autonomous characters. This is obviously incredibly valuable research, as developing rich interactive characters is only worthwhile if players can experience their depths. This work in autonomous character management and dynamic interaction is therefore crucial for addressing the larger issue of creating immersive, interactive story worlds.

Of the work I examined, only a small portion focused on the specific generation of characters stories and traits, and, most of the generated characters were used in non-interactive pieces. This creates a clear opening for a system which can generate general-purpose characters for use in interactive narrative games.

In my research, I was unable to find any work that focused specifically on generalized character structure generation for interactive games. I was, therefore, inspired to make a prototype attempt at addressing this need. The rest of this paper will discuss the system I created in response.

3. THE PERSONAGE SYSTEM

In this section, I will present the architecture underlying my procedural character structure generation system, which I call *Personage.* For this prototype development,

I decided to focus on generating general character-structures, which can be used in tabletop, social interpersonal games, or imported into digital games.

I did not design this system to target a particular game, environment or character manager, in the hopes of keeping it general enough to function for multiple applications. I therefore decided to present the generated character structures based on loosely real-world representations. This will be discussed in further detail below.

3.1 System Overview

Modeled after real-world character-structure generation (i.e. human life), *Personage* functions by simulating a generated character's life up to "the present/" For the purposes of interactive narrative, "the present" is equivalent to the point where the character will be encountered or played in the narrative.

The resulting generated character-structure has these main components:

- **Name:** The character's name
- **Gender:** the gender of the character
- **Age:** the age of the character
- **Birth-story:** The birth story of the character including:
  - birth-date
  - birth location
  - sibling order (i.e. fourth child)
  - parent names
- **Event History:** The character's formative life story up to this point. This is represented as a list of descriptions of the events the character has been through. The displayed descriptions are derived from an underlying event model which includes their associated value types, intensities, and outcomes.
- **Value Type Ratings**: how much the generated character cares about different types(from 1 to 10). These are shaped by the character's event history. This prototype includes 9 value types:
  - Family
  - Health
  - Appearance
  - Wealth
  - Education
  - Social Recognition
  - Romance
  - Friendship
  - Spirituality
- **Ability Ratings:** How able the character is in different areas(from 1 to 10). These abilities are used to determine the outcome of events during the simulation, and are likewise shaped by these outcomes. The ability ratings used in *Personage* are:
  - Intellect
  - Physical Health
  - Mental Health
  - Luck
  - Intuition
  - Charisma
  - Wisdom

3.2 The Generation Process

3.2.1 Initialization

The graphical user interface for this prototype allows users to specify the gender and viable age range for characters. Character gender does not have a significant impact on the generation, but will result in the appropriately gendered display of text.

The generation begins by randomly selecting the character's age, based on this user input. Character age is significant, as older characters will experience more life events, which will therefore result in a longer event history and a more nuanced character outcome.

The generator continues by randomly selecting the character's name and birth story. All names of people and locations are imported from United States census data, although first and last names are randomly matched.

Next, the system allocates the character's initial ability and value ratings. Abilities are randomly allocated from a system-specified amount, representing the character's genetic predisposition. Hence, one character will start with a drastically different ability profile from another. Character values all begin at the mid-point, representing an innocent, blank slate.

3.2.2 Building the Event History

The generator then begins presenting the sim-character with life events. Because of the formative nature of these events, I decided to begin tracking character lives at 7 years of age. The system contains a parameter for number of formative events per year. The current prototype generates 3 events per year, but this setting can easily be modified to suit the user's needs.

All of the event information for this system is authored externally and imported at run-time. It is therefore possible for users to modify the event associations and specifications without needing to touch the system's underlying code.

In this external event specification, each event is associated with a specific value type, such as family or education. Likewise, each of these value types is correlated with associated abilities and their percentage importance to this value type. For example, the specification I created declares that romantic events are determined by 50% intuition, 30% charisma, 10% physical health and 10% wisdom. In my specification, I balanced these proportions so no abilities are overall more significant than others. The system does not enforce this constraint, although it does require that the component ability percents sum to 100.

When presenting a sim-character with an event, the system presents a choice between uniquely value-typed events. Therefore, a character might need to choose between a wealth or romance event. The character will always choose the event-value they rate higher, choosing randomly in the case of a tie.

3.2.3 Event Evaluation

Events are evaluated as good or bad, with an intensity level of 1 to 5. Event evaluation is based on the associated abilities. In evaluation, *Personage* begins by calculating the associated weighted ability average, as described above. The system then factors in the characters luck. Luck functions differently from the other abilities, in that it is not associated with specific event value types. Instead, a “boost” factor is calculated by randomly selecting from a range determined by the luck rating. This allows lucky characters to have a greater chance of success despite difficult challenges or low ability ratings.

The event's challenge is selected by randomly selecting a number in the possible ability average range, which represents the event's challenge. In order to promote balanced characters, it is always possible for a character to attain a good or bad outcome when facing an event. Nevertheless, high or low related ability ratings will significantly affect the probability of a good or bad outcome.

If a character's luck-adjusted ability average is greater than or equal to the event's challenge, they will succeed. If not, they will fail, and the event will have a bad outcome. The intensity of the event outcome is determined by the percentage degree by which the character succeeds or fails.

3.2.4 Adjusting Abilities and Values

Additionally, events affect the character's associated value and ability ratings. After evaluating an event, the system will adjust the associated abilities based on the event intensity and percentage importance. In the example used above, a romantic event would have a more significant effect on the character's intuition than their wisdom, as intuition was 50% important to this event, whereas wisdom was only weighed at 10%.

Values are also adjusted after event evaluation, but only high intensity events can affect value ratings. The current system allows only events of intensity of 3 or greater alter value ratings. This is meant to ensure that only especially significant events in the characters life will alter his or her core values. The system will repeat this process, building the event history until the character reaches the desired age.

3.2.5 Testing the Generation

Once the event history is built, *Personage* enforces a number of validity tests. These are in place to promote the generation of balanced, interesting characters with varied stories and abilities. These tests can also be easily modified in the code, although some configurations will result in impossible constraints, which will not generate any characters.

The current constraints include:

- 30% - 70% good outcome events
- limit of 3 “extreme” values and abilities
  - extreme means a rating of 1 or 10
- At least one maximum intensity event
- limit of 10% maximum intensity events
- At least one good and one bad level 4 intensity event

I came to this configuration through testing the system, and it seems to allow well-balanced characters with varied stories and ratings.

be saved in text form, or regenerated using the same

If a generation fails this test, the system will restart the entire generation process. This is an admitted weak-point in the system, which will be discussed in the evaluation below.

3.2.6 Outputting the Generated Character

Once a generation attempt succeeds, the character's description will be displayed to the user. This prototype interface allows the user to display or hide text based on associated value type and event intensity. At this point in the process, the character's data is held in the system. It would therefore be simple to export it to an external application for a variety of purposes. Characters can also be saved in text form, or regenerated using the same random seed and settings.

4. EVALUATION

4.1 Successes

The *Personage* system successfully generates characters with fleshed out back-stories, descriptions and correlated ability and value ratings. A character-structure generated by *Personage* could therefore be used as an NPC or a PC in either a digital or non-digital game. *Personage* demonstrates that procedural character-structure generation can ease the burden of character authorship discussed earlier in this paper. Obviously, the current

style domain and ability/value specifications are not appropriate or compatible for all applications, but they could be modified to fit.

It therefore seems feasible for a GameMaster or game developer to use *Personage* for character authoring. *Personage* could also be used in real-time during non-digital games, reducing the pressure on a Game Master who is suddenly faced with an unexpected ancillary NPC interaction. Additionally, *Personage* could be used as an assistive tool for authoring; giving authors a starting character template which they can adjust to suit their needs.

Personage also shows promise for real-time application to digital games. While the system is currently stand-alone, it could be modified to export generations to another program, such as one of the character behavior managers mentioned above. Obviously, for this type of export to function properly, the systems would need to be modified to support a common character-structure representation. This would allow for generated characters to be interpreted meaningfully by the external system. Such a system would then be able to use *Personage* to dynamically generate an interactive, autonomous dramatic character.

4.2 Limitations

One main limitation of *Personage* is that it requires pre-creation of the event, ability and description associations. In order to adapt this system to suit their needs, a developer will therefore need to customize this representation and fill it out with content. This issue could potentially be addressed by authoring universal modules which could fit into differently stylized character universes. This might allow developers to incorporate the system with "off-the-shelf" components and be up-and-running quickly.

This system allows for deeply inter-related and complex character stories to emerge, and also allows for seemingly random generation of non-causally linked text. The responsibility falls on the domain author to properly implement the specifications that work for a certain application. If the domain is not fully fleshed out, the system may be forced to select the same event descriptions multiple times in a character lifetime. This issue is even more likely to appear in long-lived characters. Therefore, the authoring burden still exists, but it seems it must fall on someone.

Additionally, the constraint system on authored events is currently slim. Although events can specify simple pre-conditions, this system is limited, and allows for event selections that might not fit well within a character's story. This system will need to be further developed in order to ensure that *Personage* consistently produces coherent characters.

Another limitation of *Personage* is its generate-and-test methodology. This is not the most efficient method for searching the solution space, and it is likely that a constraint-language-based solution would be more effective. For example, the current *Personage* prototype can be fed impossible constraints, but it does not have the ability to recognize this. It will then dutifully try to generate the specified character, relying on random number generation to yield a match.

This short-coming forced me to incorporate a generation-counter and quit value into the system. Currently, after failing 10,000 times, the system will abort. Re-implementing *Personage* utilizing a logically-based constraint solver would likely address this concern. This re-implementation strategy would also allow *Personage* to solve for low-probability constraint sets, which might currently fail.

Lastly, the user interface for this prototype is sadly limited, and does not expose all of the interesting controls to the player. This would be relatively easy to implement, but would also give users access to the aforementioned options which lead to failed generation attempts.

Personage has not been rigorously tested and it is possible that long-term use will reveal shortcomings in the underlying simulation system. If this comes to be, hopefully this work will still serve to show that such a simulation-based character-structure generator can solve some of the important issues discussed here.

5. FUTURE WORK

I believe *Personage* is a promising start in this area, and that it demonstrates that this approach to character-structure generation has merit. My first steps in moving forward will be to address the limitations discussed above.

Another significant area for expansion would be to make the generated character stories dynamically expandable. In the introduction of this paper, I discussed scenarios where players wish to learn more about story-characters they encounter, or more about their own player-characters.

Because the *Personage* system itself contains the character generation model, dynamic, interactive

character expansion may be possible. A character-structure's depth could therefore be dynamically expanded to suit a player's desires.

This will require further thought and research, but would be an exciting avenue to pursue.

6. CONCLUSION

Characters are a core component of interactive narrative games. Character authoring is a laborious, challenging task, and authored characters are typically static. Game Masters are often faced with the need to improvise character authoring on the fly, while their intensively hand-authored characters might be ignored. In digital games, due to time and system resource limitations, developers frequently limit or eliminate interaction with many NPCs in the world. The *Personage* system described here offers a potential solution for some of these big problems. Procedural generation of character-structures is possible, and shows great promise for dynamically adaptive narrative settings, in which the player can feel a more full sense of immersion and agency in their interactions.

7. ACKNOWLEDGEMENTS

I wish to thank Jim Whitehead for his patience as I completed this project. Additionally, I extend sincere thanks to Amanda Triplett, Paul Alexander, Carmina Eliason, and Maitland Vaughan-Turner for their assistance in authoring prototype event descriptions.

8. REFERENCES

[1] Myst. Cyan. Broderbund. September 1993.

[2] Shadow of the Colossus. Team Ico. Sony Computer Entertainment. October 18, 2005.

[3] Dungeons and Dragons. Gygax and Arneson. TSR. 1974.

[4] Lininger, Jeff. 2009. *Procedural Back-Story Generation in the Framework of a Murder Mystery.* Southern Methodist University Guildhall.

[5] Chen et. al. 2004. *RoleModel: Towards a Formal Model of Dramatic Roles for Story Generation.* University of California Santa Cruz.

[6] Mei, Marsella and Pynadath. 2005. *Thespian: An Architecture for Interactive Pedagogical Drama.* Center for Advanced Research in Technology for Education. UCS Information Sciences Institute.

[7] Marsell, Pynadath and Read. 2004. *PsychSim: Agent-based modeling of Social Interactions and Influence.* Information Sciences Institute, University of Southern California.

[8] Cavazza, Charles and Mead. 2001. *Planning Characters' Behaviour in Interactive Storytelling.* School of Computing and Mathematics, University of Teesside.

[9] McCoy et. al. 2010. *Authoring Game-based Interactive Narative using Social Games and Comme il Faut.* University of California Santa Cruz.

[10] Sullivan et. al. 2011. *Extending CRPGs as an Interactive Storytelling Form.* Center for Games and Playable Media, UC Santa Cruz.

[11] Riedl and Young. 2005. *An Objective Character Believability Evaluation Procedure for Multi-Agent Story Generation Systems.* University of Southern California. North Carolina State University.