

Automated Generation of Conversational Non Player Characters

Grant Pickett, Foad Khosmood and Allan Fowler

Department of Computer Science
California Polytechnic State University
San Luis Obispo, USA

Abstract

An integral part of social believability in role playing games is believability of non-player characters (NPC). In this paper we argue for the importance of believability in NPCs, even those that are completely outside of any pre-written quest or plot. We present NPCAgency, a system designed to generate many conversational NPCs as packaged narrative assets that can be shared and imported into various projects to increase story-world immersion. We believe such a system can help solve two problems. First, the authorial burden of the game designer is lessened, allowing renderings of large numbers of NPCs, each with their own unique background and conversation topics, all conforming to the norms of a predefined “universe”. Second, the immersive aspect of the game is heightened as the player can engage complex characters with lengthy dialogue affordances. We demonstrate the concept by generating fifty characters with attributes drawn from “Game of Thrones” (GOT) / “A Song of Ice and Fire” universe, and exporting them as Inform 7 code, a popular declarative interactive fiction (IF) programming language and authoring tool. A user study of thirty-seven Inform 7 programmers demonstrates that a 62% majority find the tool useful enough to consider for their own work. Further 70% said they would use the system to create “Game of Thrones” background characters for their own projects.

Introduction

NPCs have become integral to role playing games (RPG) or adventure games. Some of these characters are written to play major or minor roles in game plots and the player’s interaction with them is typically part of advancing the narrative. Others have no relationship to the plot, but their presence is meant to make the story world scenery richer and more believable. Much work has been done, mostly for the former category of NPCs, resulting in more entertaining plot devices, personable shopkeepers and mysterious strangers.

Yet still in today’s AAA adventure and RPG games, the vast majority of NPCs, with the possible exception of a

handful of major plot characters, remain bland, robotic and lifeless. Most of the NPCs that serve plot purposes offer very limited interaction having to do with their specific purpose. The ones that are not part of any plot or quest are barely distinguishable from the background scenery, often offering no interaction at all.

The reasons offered for these shortcomings are understandable. The industry would like to focus on delivering better official plots and quests that players experience. Writing dialogue trees, backstories and believable personality traits for hundreds of characters is cost prohibitive. Creating voice assets are expensive. We believe significant automation and asset-reuse in this area is possible and hope to contribute to creation of tools and processes that allow efficient and economical generation of higher quality NPCs.

In this paper, we present NPCAgency, a system of generating conversational non-player characters (NPC) based on predefined universe models and author specification. The NPCAgency tool is designed to create and store large numbers of NPCs associated with a story universe such as *Star Wars*, *Arthurian tales* or *Game of Thrones*. These characters would have universe-appropriate attributes and backstories, and would be able to hold limited conversations about themselves and others in their cohort. The system also supports ways of exporting these characters to game engines for use in populating game worlds to make them more content rich, immersive and believable.

The system consists of a model store, a generator and multiple translators (Fig. 1). The model store is where universe-specific sets of traits are kept independent of any particular engine or environment. Various models are possible with trait layers appropriate for their respective universes. To support a new universe, we encode facts with sets of tokens (names, places, characteristics). Some traits are physical, affecting only appearance; others are psychological and social affecting behavior or mannerism. Characters also have life stories in form of event memories within temporal eras such as “childhood” and “elder years” depending on the universe. These eras serve as containers for

“life events” which manifest themselves as memories or psychological dispositions for each character. Some NPC may have had an adverse childhood and perhaps lost a parent at that time, leading to traits such as “self-reliance” or “bitter”. Most traits have numerical “likelihood” data associated with them, as well as inclusion/exclusion lists preventing them from appearing in certain combinations. For example in the GOT universe, the social class into which one is born highly influences the possible career and marriage options.

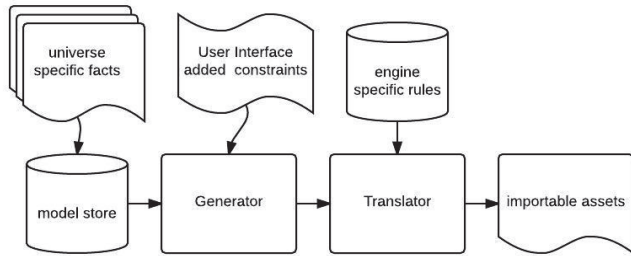


Figure 1. The NPCAgency components

The generator uses traits, hard-coded, universe-specific likelihoods and restrictions, as well as user adjustments to further constrain the possible characteristics of an NPC, before stochastically generating them.

Besides traits and life stories, the characters also have limited contextual dialogue assets allowing them to interact with the player. The template-based dialogue assets are designed to answer questions about the character traits, background events and knowledge regarding other characters. The dialogue system also features a limited amount of delivery mannerisms, further reinforcing certain traits such as “verbosity” and “patience”.

The last major component of the system is a set of “translators” which transform the generated traits into code that can be used by various game engines or interactive simulation development environments. Due to the extra burden of having to support 3D models and artistic depictions, we concentrate on interactive fiction translators. While physical characteristics and attributes are very much part of our NPC models, only verbal descriptions are necessary to support for interactive fiction projects.

Background

While there are many basic character generators for pen and paper type games available online, these focus on specific rule-sets and providing simple numerical attributes (JavaScript D&D, 2015). These still depend on a person to translate the basics into character actions and emotions. Others generate all the necessary identifying information for a fictional person, but this presumably is just random

data from a database without much thought to the specifics of a story universe.

Story generation systems have a long history in the field. The work of Lebowitz is a prominent example. The author designs a system for creating entire stories emphasizing a need for consistency and coherency (Lebowitz, 1984). Lebowitz’ UNIVERSE features pre-defined characters, character traits and historical plots that can be generated with any of the available characters playing the available roles. UNIVERSE also modeled relationship between characters and use them as preconditions for certain plots. Corradini’s group sets out to focus on creating a realistic depiction of a very specific character, in this case one by Hans Christian Anderson (Corradini, 2005). This is still a generated character, created for the purpose of interaction, but is only one specific person rather than an extendable generalized approach. The work by Kienzle demonstrates a generational model to create variety in non player character actions (Kienzle, 2007). Most recently, (Veale, 2014) created Flux Capacitor, a system for creating interesting character arcs that fit into the overall dramatic action.

Merrick discusses a technique for making characters react to changes in the environment over time (Merrick, 2006). She does not discuss the initial generation of the characters however so it presumably is hand written. Lankoski and Bjork examine what makes a character believable in role-playing games in the context of *The Elder Scrolls* series in particular (Lankoski, 2007).

The work presented in this paper is influenced by many of the above works, but has fundamentally a different aim. It is focused on making conversational characters that are not part of the main plot, or the main dramatic experience, or play any kind of protagonist/antagonist roles. NPCAgency characters are not aware of any quest or the plot progression of the game. Their function is solely to provide a more believable world based on what one is likely to encounter in the particular fictional universe.

System Architecture

Universe

To allow flexibility for generating characters the tool requests from users a collection of universe appropriate facts. The imported files use JSON conventions and internal consistent reflexive references for the tool to process and parse the material in these files.

A parsing convention between the database and generator allows for more flexibility. A line of dialogue can have wildcards that the generator will resolve. For example the segment “{get() social}” in will be replaced with “rich” or “poor” or another social class.

A specific mandatory field defines first what traits a person will have and from that root, a person object is generated by copying segments of the database to the character

object. The database used for the demonstration user study contains over a 1000 facts about GOT characters.

```

PersonForm.html | *GoTdatabase.js | PersonForm.js
1 var GOTDB = {}
2 {}
3 person:{"firstName":"birth", "lastName":"birth", "age":"birth",}
4 region:"birth", "gender":"birth", "job":"yngAd", "eye color":"birth",}
5 hair_color:"birth", "house":"birth",}
6 nodes:{"life":["childhood", "adolescence", "young adulthood", "mid-life",}
7 elder years"],}
8 ages:["Child", "Adolescent", "Young Adult", "Mid-life", "Elder"],}
9 cities:["a large town in ", "a small village in ", "the middle of ", "",}
10 that famous part of "],}
11 eye colors:["blue", "brown", "green", "gray"],}
12 genders:["male", "female"],}
13 hair colors:["yellow", "black", "brown", "white"],}
14 clothes:{"}
15 [Random]:"what most young {get() social} people wear,",}
16 farmer:"sun faded clothes and shoes caked in mud",}
17 blacksmith:"an apron smeared black over brown clothes and thick gloves",}
18 fisherman:"clothes smelling of fish and tall black boots",}
19 merchant:"almost gaudy clothing of many colors",}
20 rich:"adorned with fancy markings and wearing some jewelry.",}
21 guard:"a leather jacket over some padded clothes",}
22 common:"of average quality" }

```

Figure 2. Example database content

Character Creation

To use the tool a “universe” must first be selected. The universe of *A Song of Ice and Fire* from the book series by George R.R. Martin is chosen to demonstrate the proof of concept in this project. Specific reasons include the familiarity of the authors and evaluators with the popular book and TV series. The variety of regions and social classes in the world allows for many characters to be generated that are very different from one another. The GOT world is much like a medieval world with a feudal system lacking a high chance of social mobility.

The Character Model and Generator

A character or a group can be generated with a single button press that triggers the tool to read the database and generate all requested parts. A web-based user interface allows for the user to create more constraints on the kind of characters to be created. The system works within the constraints provided by the user and stochastically generates the rest of the traits and relationships accordingly.

The features selected for a character must relate to one another in a way that paints a picture of a person rather than a collection of random events or traits. All features are discrete string choices within sets, sometimes a set of only one item. Traits have a default numeric probability value at creation but some unique traits only exist if they are generated by an event, like “gaining occupation”. Furthermore, the system allows specification of “influences” from any trait to any other. For example a choice of hair color has a default probability distribution of {black: 0.5, brown: 0.2, blonde: 0.2, red: 0.08, silk:0.02}. But a selection of “house Targaryen” trait adjusts the hair color distribution to make it extremely probable for silk to be chosen.

Figure 3 shows the main components and data dependencies for the character generation process. Once the universe is chosen the system reads all known facts and probabilities from a database. Next a location is chosen based on

the main regions of Westeros, or whatever regions the universe object had specified. Like all facts, the list of Westerosi regions is accompanied with a probability distribution that was hand crafted based on population distributions known in Westeros. After location, a name is chosen from the database of GOT first and last names obtained from 1300 known characters listed on *A Wiki of Ice and Fire* (westeros.org). Since the idea in this project is to generate a random typical name, and not an actual existing character, we first stochastically choose a last name (House name) and then a random first name from all first names available in the entire set.

The next phase is creation of character traits. The first character traits are those, present from birth. These would include any number of physical traits (blue eyed, tall, strong, etc.) and a limited number of psychological traits, basically anything not physical (such as being temperamental or drunkard). Lastly a small set of social traits, such as class and wealth are chosen for the character. As can be seen in Figure 3, there is a mutual dependency between traits and events. For this reason, not all traits are chosen at this stage. Some traits are only triggered based on life events and will be filled in if the appropriate event is chosen in the next phase.

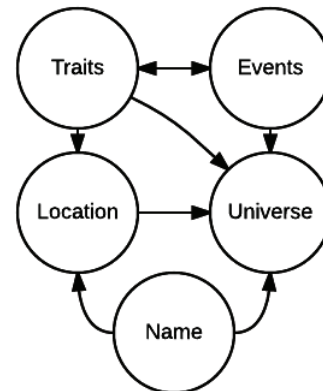


Figure 3. Main character generation components. Arrows indicate data dependencies.

Figure 4 shows the basic algorithm for generating initial character traits, and storing them in the “Person” object. The function accepts as arguments the universe object containing all the facts (U), a matrix (I) containing all the trait-to-trait influence adjustments, and the object (P), which represents a character and is meant to contain all traits. P may be initially empty or it may be partially filled by the web-based character creation tool. The algorithm systematically creates new traits, by looking up initial distributions of each valid trait-set, but it also calculates a new effective distribution by adjusting the initial with all the influences that may have been triggered, and then re-normalizing the

new distribution. The final stochastic selection is made using the effective probability distribution.

```

function Generate-Traits(U, P, I) returns P, the updated object
containing all person traits

inputs: U, an universe object containing trait-sets
P, a person object with possible partial traits specified
from user interface
Ix,y, a two dimensional table specifying mutual
influence of trait x on trait y

for each trait-set in U if trait-set not in P do
  adjusted-trait-set ← {}
  for each trait in trait-set do
    adjusted-trait ← trait
    for each t in P do
      if It,trait is not NULL then
        adjusted-traitprob ← adjusted-traitprob * (1 + It,trait)
      add Max(0,adjusted-trait) to adjusted-trait-set
    adjusted-trait-set ← Normalize(adjusted-trait-set)
  chosen-trait ← Stochastic-Choice(adjusted-trait-set)
  add chosen-trait to P
return P

```

Figure 4. Algorithm for creating new character traits

Character Life Events

The next step in creating a realistic character is to give them a series of life events to build a plausible history for the individual. For the purpose of this generator, the character’s life has been broken into different life stages like childhood, young adulthood, marriage, middle aged, and old. Every era has one or more event-sets associated with it. Much like UNIVERSE’s historical plots (Lebowitz, 1984), a superset of events is pre-defined. Examples include “War experience”, “lost a parent”, “spouse death” and “imprisonment”. All eras other than childhood can produce new traits as a consequence of events.

In the translation system, every event is associated with one more sentences that describe it from a first-person point of view. Much like traits, events are also listed with an existing probability. The probability distributions are subject to change due to side affect of new traits.

A character after being generated in a child state will “age” by having events pushed on to the character's timeline. Using the correlations and restrictions mentioned before, the system can avoid generating paradoxical events. Social class plays the strongest influence on which events can happen to characters in the GOT example world. A life event can change attributes about a person however, which then changes which future events can trigger. A rich male merchant might lose most of his money and become poor. An event can require a character to have at least a 75% level in generosity to trigger while another event might reduce or increase a character's generosity value. The type of job a character takes can be based on what the database model

indicates the character would be good at doing within the social restrictions of the GOT universe.

Emergent narratives can form from this process. Characters might get married only to have their spouse die for one of several reasons and then have a chance to remarry. These characters in this world are either married or unmarried and events which attempt to change this state first check if the character can gain a spouse. In the demonstration world, polygamy is not possible. However another world using this tool might not have a restriction against it from happening.

The end result of the generator is a person model with attributes relating to their life events, their traits, their location, their name, and other key features. It is up to the translation system to take this model of a person and transform it into an interactive format consistent with the game engines that it supports.

The Translator

The translator takes the model and transforms it into usable Inform 7 code. The Inform 7 language was chosen due to the ease of generating text as opposed to visual representations of characters. The plan for this system is that translation plugins could exist for multiple engines like Unity and Unreal in addition to Inform 7.

Inform code is generated line by line as a script reads the JSON person file and follows conventions about which pieces end up doing specific things inside of Inform. Appendices A and B provide an example translation from JSON to Inform 7.

The tool creates an Inform 7 extension file to download and import using the Inform IDE. Currently the user must place the characters wherever they want into the game world, consistent with what an author would choose to do with characters. The author could naturally add more material to the characters as desired. The characters provided by this tool are primarily meant to be NPCs that fill out the background of scenes. Figure 5 shows a sample interaction with one of the characters in the sample game world.

The conversations that the users have with the characters are facilitated by one of the many extensions available form Inform 7, “Conversation Rules” by Eric Eve (2015). This system allows the user to ask the character in the game about specific topics and provides a framework for more complex dialogue interactions. The characters' dialogue content is distributed into relevant tables in Inform and into “ask” queries by the translator module. The “topics” command provides a list of suggestions for a player to ask a character about.

A character’s significant physical traits are listed when “looking” at the character in the game. When multiple characters are generated, they are aware and able to talk about some of the other characters’ traits.

User Study

To validate the system, a demonstration and survey is conducted at a college campus with students familiar with the Inform 7 programming language. Students with some Inform 7 practice, who are taking part in a course with an Inform 7 game creation assignment, are asked to participate in the study.

A website hosting the most recent version of the project is made available for live testing². After hearing a short presentation on this tool, the students collectively create a custom character using the web interface and then use a different website to play an in-browser version of the Inform 7 game world with 50 characters generated by the tool including the customized character. The students are instructed to explore the world and try to locate the character that they helped create. After some conversation, 37 users, about half of them familiar with GOT, fill out a survey asking about the project. The results are provided in Table 1.

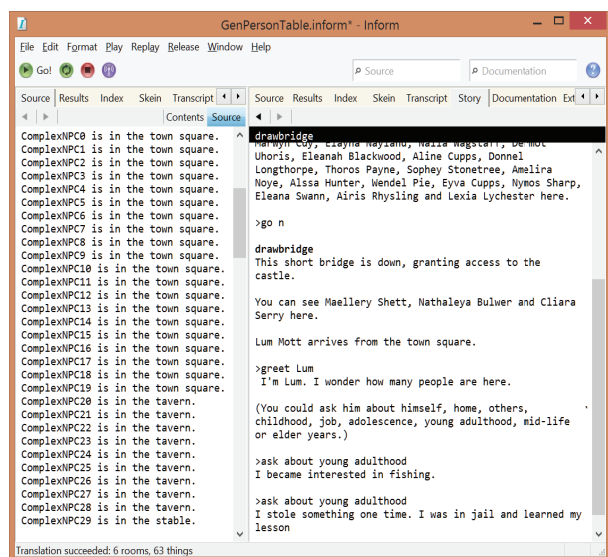


Figure 5. Example Game in the Inform 7 IDE

We use three point scale responses in this survey and ask 11 questions. The responses generally indicate the characters are perceived as believable and interesting. Most of those familiar with the GOT universe state they did not find characters out of place and did in fact represent the GOT world as they know it. They think the tool would be easy to use whether familiar with the Inform 7 language or just starting out. A majority state it would consider using this tool in creating Inform stories. Some of the questions seem more neutral due to a lack of familiarity with the Game of Thrones world. A vast majority understood the utility of a tool like this for several types of developers.

² Readers are encouraged to try a version of the character generation tool here: <http://users.csc.calpoly.edu/~foaad/NPCA> and the live playable example game here: <http://users.csc.calpoly.edu/~foaad/NPCA/NPCAagency/play.html>

Many gave the impression that this tool was a good start and liked what features it promised but at this time one programmer's efforts over a few months was not convincing as a final product. Interesting focus was given toward dialogue being somewhat terse which is a consequence of the proof-of-concept nature of this demonstration where writing procedural Game of Thrones fan-fiction is not a priority but rather creating a modular engine which would accommodate another user's development of the world.

Table 1. Survey Results

Question	Disagree	Neutral	Agree
Intuitive for someone familiar with Inform 7.	0	8	29
Intuitive for someone unfamiliar with Inform 7.	2	18	17
Easily adaptable tool	0	6	31
Would save a lot of time for me.	5	13	19
Would consider this tool for my own Inform 7 stories	5	9	23
Characters' backgrounds believable	0	20	17
Would consider using a character like these as a main character for my own Game of Thrones	7	14	16
Encountered memorable characters	7	19	11
Most of the characters I met were believable Game of Thrones characters	0	22	15
Wanted to continue engaging with some interesting characters	5	14	18
Would consider using characters like these as background characters	1	10	26
Village scenario was believable	2	14	21
Characters represented populations found in Game of Thrones	1	23	13
Found and interacted with the character we created in class	2	5	30
Character we created consistent with custom creation choices	1	7	29

Conclusion and Future Work

In this paper, we demonstrate a proof-of-concept tool capable of producing realistic universe-specific characters that can be easily imported into game engines (in this case Inform 7 IF works), thus saving resources. The concept is validated with a survey that shows users familiar with the

genre and technology consider adopting the characters in their own stories.

A significant part of the project has been learning how to generate Inform code in a modular fashion that can extend indefinitely. Inform has some interesting conventions that made it difficult to translate the object-oriented ideas into something the inform compiler would use.

Planned features include extending the generation process to provide locations from large to small towns and areas for the people to move around and live in. Also character can be given branching conversation trees to provide more interactive choices when interacting. Characters will be able to be part of larger factions that shape their traits and behaviors.

Appendix A: Sample Portion of the JSON NPC Specification

```
{
  "gender": "male",
  "firstName": "Wendell",
  "lastName": "Saan",
  "marriage": "none",
  "timeline": [
    {
      "event": "Cool Older Sister",
      "line": "It was too bad that my big
sister moved away to live with her husband.
We don't see each other much.",
      "era": "Child"
    },
    {
      "event": "Kind Family",
      "line": "I had a kind family.",
      "requires": {
        "childTone": {
          "is": "Positive"
        }
      }
    }
  ],
}
```

Appendix B. Sample Generated Inform7 Code

```
Table of ComplexNPC0 Answers
subject response rule response table suggest
ComplexNPC0 ComplexNPC0 ask-self rule -- 1
home--Table of ComplexNPC0 home 9
childhood --Table of ComplexNPC0 childhood 9
job --Table of ComplexNPC0 job 9
adolescence --Table of ComplexNPC0 adolescence 9
young adulthood --Table of ComplexNPC0 young
adulthood 9
mid-life--Table of ComplexNPC0 mid-life9
elder years --Table of ComplexNPC0 elder years 9
Table of Table Types (continued)
tablename index tabtype
Table of ComplexNPC0 Default Quiz Responses 0
stop-list
Table of ComplexNPC0 Default Inform Responses 0
stop-list
Table of ComplexNPC0 childhood 0 stop-list
Table of ComplexNPC0 home 0 stop-list
```

```
Table of ComplexNPC0 job 0 stop-list
Table of ComplexNPC0 adolescence 0 stop-list
Table of ComplexNPC0 young adulthood 0 stop-list
Table of ComplexNPC0 mid-life0 stop-list
Table of ComplexNPC0 elder years 0 stop-list
```

Appendix C. Sample Interactive Fiction Dialogue with a Generated Character

(User entered lines is prefixed with ">")

town square

A large statue of an ancient lord stands overlooking the crowded marketplace that has been setup in the center of town.

You can see Lenda Fell, Helman Sand, Anysa Royce, Malina Mage, Alaysha Ryswell, Alerah Staedmon, Kaysie Blackthumb, Ronnet Lydden, Jurne Swyft, Hosman Bolling, Lexia II, Joenne Bolton, Andrew Liddle, Otter Byrch, Lennocks Cerwyn, Larence Mullendore, Milah Serry, Tomard Estren, Edmyrn Hewett, Lucamore XII, Arlenna Quaynis, Brigot Wynch, Enith Waters and Fryda Blacktyde here.

>Examine Helman Sand

This person is wearing sun faded clothes and shoes caked in mud adorned with fancy markings and wearing some jewelry. This person is not tall. This person is large. This person is agile.

>Talk to helman

I'm Helman. I'm busy right now, what is it? (You could ask him about himself, home, others, childhood, job, adolescence, young adulthood or mid-life.)

>Ask Helman about his job.

I am a farmer.

Miranna Bolton arrives from the tavern.

Cosgrove Tattersalt arrives from the town entrance.

>Ask Helman about his childhood.

I became more brave in my childhood years.

Miranna Bolton heads to the drawbridge.

>Ask Helman about his childhood.

I became less trusting in my childhood years.

>Ask Helman about himself.

The traveller looks annoyed by your question, for some reason, and declines to answer.

Jaide Penny heads to the town entrance.
Milah Serry heads to the drawbridge.

>Ask Helman about his home

I am from a small village in The Iron Islands

>Ask Helman about his home

I was born in 267 to a rich family.

>Ask Helman about his home

I live in the lands of the noble House Goodbrother

>Ask Helman about his home

Our sigil is a gilded black horn on red

Alerah Staedmon heads to the stable.

>Ask Helman about his childhood.

I became more creative in my childhood years.

>Ask Helman about his mid-life.

I was drafted to fight in the war for a time.

Alerah Staedmon arrives from the stable.

>Ask Helman about his mid-life.

Having a job isn't half bad.

Joenne Bolton arrives from the tavern.
Cosgrove Tattersalt heads to the drawbridge.

>Ask Helman about his mid-life.

It seems like I'm often bored of my routine.

>Ask Helman about his mid-life.

That is about it.

Luwin Noye arrives from the tavern.

>Ask Helman about others.

Alerah is somewhat brave

Breonna Bracken heads to the tavern.

>Ask Helman about others.

Ronnet is reliable

>Ask Helman about others.

Ronnet is somewhat assertive

References

- Corradini, A., Mehta, M., Bernsen, N. O., & Charfuelan, M. 2005. Animating an interactive conversational character for an educational game system. In Proceedings of the 10th international conference on intelligent user interfaces, 183-190. San Diego, Calif: ACM.
- Eve, E., 2015. Conversation Rules. Inform 7 Extension. Online: inform7.com.
- Inform7. 2015. Online: inform7.com.
- Javascript Dungeons and Dragons 3.5 character generator. 2015. Online: <http://archive.wizards.com/DnD/Tools.aspx>
- Kienzle, J., Denault, A., & Vangheluwe, H. 2007. Model-based design of computer-controlled game character behavior. In Model Driven Engineering Languages and Systems 650-665. Springer Berlin Heidelberg.
- Lankoski, P., & Björk, S. 2007. Gameplay design patterns for believable non-player characters. In Situated Play: Proceedings of the 2007 Digital Games Research Association Conference 416-423. Toyko, Japan: DIGRA.
- Lebowitz, M. 1984. Creating characters in a storytelling universe. Poetics 13(3), 171 – 194.
- Merrick, K., & Maher, M. L. 2006. Motivated reinforcement learning for non-player characters in persistent computer game worlds. In Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology. Hollywood, Calif: ACM.
- Veale, T. 2014. Coming good and breaking bad: Generating transformative character arcs for use in compelling stories. In Proceedings of ICCG-2014, the 5th International Conference on Computational Creativity, Ljubljana.