Simulating Character Knowledge Phenomena in *Talk of the Town*

James Ryan and Michael Mateas

[1 leave this spacer to make page count accurate]
[2 leave this spacer to make page count accurate]
[3 leave this spacer to make page count accurate]
[4 leave this spacer to make page count accurate]
[5 leave this spacer to make page count accurate]
[6 leave this spacer to make page count accurate]
[7 leave this spacer to make page count accurate]
[8 leave this spacer to make page count accurate]
[9 leave this spacer to make page count accurate]
[10 leave this spacer to make page count accurate]
[11 leave this spacer to make page count accurate]
[12 leave this spacer to make page count accurate]
[13 leave this spacer to make page count accurate]
[14 leave this spacer to make page count accurate]
[15 leave this spacer to make page count accurate]
[16 leave this spacer to make page count accurate]
[17 leave this spacer to make page count accurate]
[18 leave this spacer to make page count accurate]
[19 leave this spacer to make page count accurate]
[20 leave this spacer to make page count accurate]

1 Introduction

There are many examples of stealth and action games that use complicated knowledge, perception, and alertness models to produce short-term NPC beliefs that are a core part of gameplay [Diller 04, Welsh 13, Walsh 15]—here, notable examples include *Thief: The Dark Project* [Leonard 03] and *Third Eye Crime* [Isla 13]. Few projects, however, have supported characters whose perceptual systems instantiate memories or lasting beliefs, and there are even fewer examples of the modeling of *fallible* character memory [Ryan 15]. Meanwhile, issues of belief—especially false belief—are often central in other fictional media [Palmer 04]. Some games *are* about character beliefs, to be fair, but in these cases beliefs are typically handcrafted, as in *LA Noire* [Team Bondi 11]. In games that do model character knowledge procedurally, the AI architecture that handles such concerns is often called a *gossip system* [Crawford 04]. A classic example of this type of architecture drives the reputation system in *Neverwinter Nights* [BioWare 02], while more recent examples include the rumors system of *Dwarf Fortress* [Adams 15, Ryan 15] and the beliefs system of Versu [Evans 14]. Frequently, however, gossip systems in games provide only ancillary support to core gameplay. As such, we find that games that are about character beliefs model them with

human-authored scripts, whereas games that model such knowledge procedurally tend to do so secondarily to core gameplay.

In this chapter, we present an architecture that, to our knowledge, simulates character knowledge phenomena more deeply than any earlier system has [Ryan 15], all in service of a game that is fundamentally about character beliefs, called *Talk of the Town*[1]. While the game is still in development, the architecture that we present in this chapter is fully implemented. Relative to other gossip systems, like the ones in *Neverwinter Nights* and *Dwarf Fortress*, our system takes a fiercely agent-driven approach, with character knowledge propagating as a result of discrete character interactions. This contrasts the more typical method of abstractly modeling information flow across the gameworld. While the latter approach is more computationally efficient, it would undermine a number of our goals for the *Talk of the Town* player experience, as we discuss in depth at the end of this chapter.

2 *Talk of the Town*

Talk of the Town is an asymmetric multiplayer *dwarflike* (a game in the mold of *Dwarf Fortress* [Adams 15]) that features character knowledge propagation as a core mechanic. In this section, we describe its story, simulation, and our gameplay design; the simulation is fully implemented, but the gameplay layer is currently being developed.

2.1 Story

The story that frames gameplay surrounds the death of a very important person in the town in which gameplay takes place. This person had accumulated considerable wealth and produced several descendants who now constitute an aristocracy in the town. Many of these family members had been anticipating the person's death for the inevitably large inheritances that would thereby be disbursed, but in his or her last moments the person apparently signed a document willing everything to a secret lover whose existence had not been known to the family. In one week, the town will gather at a theater to remember the deceased and to hear the will be read, but the family plans to ascertain the identity of the lover and apprehend this person before the document can ever be delivered to the presiding attorney. Meanwhile, the town is abuzz with rumors about the mysterious lover, whom a handful of witnesses briefly observed on the night of the death.

2.2 World Generation

Prior to gameplay, the town is simulated from its founding in 1839, when a handful of families converge on an empty townscape to establish farms, through the death of the central character in the summer of 1979. As in *Dwarf Fortress*, this *world generation* procedure causes a number of structures that are critical to gameplay to emerge bottom-up from the simulation itself. Specifically, these are the town's physical layout (namely the locations of

---

[1] The development of *Talk of the Town* is being carried out by a growing team comprising James Ryan, Michael Mateas, Noah Wardrip-Fruin, Adam Summerville, Tyler Brothers, Tim Hong, Joyce Scalettar, and Jill Yeung. Adam Summerville also contributed to the design of the architecture described in this section.

its businesses and homes), its residents' daily routines, and, most importantly, the town's social and family networks that knowledge propagates over. Elsewhere, we provide a detailed account of how character social networks, in particular, are produced [Ryan 16c]. Throughout this simulation procedure, NPCs act out daily routines across day and night cycles by either going to work, going on errands, dropping by places of leisure, visiting friends and family, or staying home. Additionally, characters may, for instance, start a business, hire an employee, build a house, marry another character, give birth to a new character, and so forth. NPCs decide what to do by utility-based action selection [Mark 09]. When at the same location (a home or business), characters may interact at probabilities that depend on their relationship and their personalities. From a simple affinity system, continued interaction may breed contempt, friendliness, or romantic feelings (these work unidirectionally and may be asymmetric) [Ryan 16c]. The combinatorics of these simple character behaviors over more than a century of game time is enough to generate rich social, family, and work networks by the time that gameplay takes place, at which point around 300-500 NPCs will live in the town.

2.3 Gameplay

Talk of the Town's gameplay layer is still being implemented, so in this section we will describe its design. The game is multiplayer and asymmetric: one player, the *lover*, controls the lover character and the other player, the *family member*, controls a member of the deceased person's family. The lover's goal is to go undetected until the will ceremony, while the family member works to ascertain the lover's appearance before that time.[2] Gameplay culminates in a scene showing the town's citizens filing into the theater for the will ceremony, during which time the family member must select the person who best matches his or her conception of the lover—if this player selects correctly, he or she wins; otherwise, the lover wins.

The town is represented as an isometric, tile-based 2D world, spanning nine-by-nine city blocks. Each tile contains either part of a street, part of a building (home or business), or part of an empty lot. Players navigate their characters across the town by moving across tiles using directional inputs. When a building is close, the player can click on it to have her character enter it. Building interiors are also rendered as tile-based 2D environments, with tiles containing furniture and characters. When an NPC is close enough to the player character, the player can click on him or her to engage in conversation. This is *Talk of the Town*'s core gameplay interaction, since it is how the player will solicit and spread information. Additionally, player characters can patronize certain businesses through dialogue interaction with employees—this is critically how the lover can change her character's appearance (*e.g.*, getting a haircut at a barbershop or buying glasses at an optometrist). We are ambitiously aiming for dialogue interaction in *Talk of the Town* that is fully procedural, extending our earlier work on *Façade* [Mateas 04]. Conversations will

---

[2] A given character's appearance is the composite of 24 facial attributes—*e.g.*, hair color, hair length, eye color, nose size—which are inherited from the character's parents.

proceed by turns, with NPCs producing generated dialogue and players typing in their dialogue in free text. We have already developed a fully implemented *dialogue manager*, which is a module that handles conversation flow, updates NPC beliefs according to the semantic content of utterances, and reasons about conversational norms to form content requests on behalf of NPCs [Ryan 16a]. Content requests are then processed by a *natural language generation* (NLG) module, which produces NPC dialogue on the fly; this system is also fully implemented, with a prototype that allows NPCs to engage in small talk using a pool of nearly three million generable lines of dialogue [Ryan 16d]. For *natural language understanding* (NLU)—the task of processing arbitrary player utterances into semantic representations—we are in the early stages of developing an approach that uses deep neural networks [Summerville 16]. NLU is a notoriously difficult challenge, and so we have contingency plans for another dialogue-system design in the event that our method doesn't work well enough; this interface would enable players to construct modular utterances out of dialogue components, somewhat in the style of Chris Crawford's Deikto language [Crawford 07] or *Captain Blood*'s UPCOM interface [Exxos 88]. In any event, deployed utterances will be displayed as speech bubbles emitting from the characters who speak them. By virtue of our dialogue manager and NLG module, NPCs may engage in background conversation with one another. If the speech bubbles they emit are in view, the player can eavesdrop on the conversation; we plan to use this both as a storytelling device and a way of improving the believability of background characters [Ryan 16e].

Gameplay will proceed by day and night timesteps that span the week leading up to the will ceremony, with players taking a turn on each timestep. We hope for gameplay to be networked, but we have contingency plans involving local multiplayer and AI opponents. Player turns will be constrained either by a time limit or by a notion of resources (to be spent on a combination of navigation steps, conversation turns, and elapsed time). Between turns, character knowledge phenomena are simulated (see next section), crucially allowing for the propagation of information originating in the player activity of the last turn.

Because their character is well-established in the town, the strategy of the family member will be characterized by *management* of the town's knowledge network. This is because the dialogue manager reasons about how NPCs will respond (including whether to divulge information) by considering their affinities toward their interlocutors [Ryan 16c]. Being quite established in the town, the family member is more likely to encounter NPCs who are comfortable being open with him or her. As such, this player will likely spend her turns soliciting town residents for gossip about the lover (whose mysterious identity is the titular talk of the town). Here, both apparently true and apparently false information are useful. True information obviously helps the family member to ascertain the lover's identity, but patently false information could have actually originated with the lover—a fundamental family-member strategy thus becomes homing in on the sources of apparently deliberate misinformation.

The lover's strategy, then, is to *pollute* the town's knowledge network by changing her character's appearance and spreading misinformation about the identity of the

mysterious lover (through dialogue interaction with NPCs). Given the above, however, it is critical for this player to not pollute the network too extravagantly, since this could lead the family member right to the lover character's identity. One core lover tactic will be using *false flags*, that is, intelligently claiming other characters as the original sources of any misinformation that he or she attempts to spread. We also want the lover to be able to reason about whom exactly to impart misinformation to. Since NPCs are more open to characters they know and like, this will promote tactics that require the lover to build up trust relationships with NPCs so that they will be more likely to believe and propagate misinformation. As noted above, gameplay ends with the town filing into the theater for the will ceremony, at which point the family member must attempt to select the lover from a lineup by clicking on the character that best matches her conception of that person.

Broadly, we want the gameworld to feel like it's sprawling with rich NPCs who each have their own unique experiences that are captured by the particular beliefs that they have adopted. In this way, we hope that navigating the town and interacting with NPCs will feel like exploration, but with the depth of the gameworld being expressed more in its character belief structures than in its modest physical size.

3 Simulating Character Knowledge Phenomena

Characters in *Talk of the Town* build up knowledge about the world as they go about their daily routines. In this section, we describe our simulation of character knowledge phenomena, including the mechanisms by which knowledge may originate, propagate, deteriorate, and terminate according to the procedures of our architecture.

3.1 Overview

People and places in the gameworld have perceptible features, which characters may directly observe to form beliefs about them. Such knowledge may then propagate across characters during social interactions. Critically, character knowledge may also be misremembered (in multiple interesting ways), or be altogether forgotten. All throughout, the system keeps track of belief histories and knowledge trajectories, since we anticipate visualizing summaries of this kind of information at the end of gameplay.

3.2 Requirements

Our method has some architectural requirements, which we will list in this section. First, characters must have *perceptible attributes*, meaning attributes that are directly observable by other characters. In *Talk of the Town*, these are mainly physical features, like hair color, but we also model *conditionally* perceptible attributes—for instance, a character's workplace is observable while they are in the act of working.

Next, a *radius of perceptibility* must be modeled, where characters within the radius of another character may observe his or her perceptible attributes. This radius is also used to determine whether a nearby character is close enough to eavesdrop on a conversation, as we discuss later. Because we model space discretely in *Talk of the Town*, we simply say that

characters at the same location in town are near enough to perceive one another.

Additionally, system authors must craft a procedure for determining *character saliences*. Character saliences work together with attribute saliences, described next, to determine the probability of knowledge phenomena occurring for a given character and attribute—that is, the probability of a perceptible attribute being observed, as well as the probability of a belief about any attribute being propagated, misremembered, or forgotten. In Section 3.6, we explain salience computation in more depth.

Similarly, our architecture utilizes specified *attribute saliences*, which prescribe how likely given character features are to be observed and to be talked about among characters. In *Talk of the Town*, this captures, for instance, that aspects of a person's hair and eyes will be more salient than those of his or her nose and chin.

Finally, authors must also produce a *belief mutation graph*, which specifies how particular character beliefs can mutate, and at what probabilities. We discuss belief mutation in more detail in Section 3.8, and Figure 3 shows excerpts from the belief mutation graph authored for *Talk of the Town*.

3.3 Ontological Structure

As illustrated in Figure 1, a character's composite knowledge of the world is structured as an *ontology* of interlinked mental models that each pertain to a single person or place. The interlinking occurs when a character's belief about some attribute of a character or place resolves to some other character or place for whom or which they have another mental model. For instance, a character may believe that a person works at some business in town, and so his or her belief about that person's workplace would itself link to his or her mental model of that business. We use this ontological structure for elegance and convenience, since it allows characters to reason about entities in terms of knowledge they may already have about related entities (rather than by instantiating redundant or potentially inconsistent knowledge). In the case of a character knowing where another character works, this allows the former to reason about, for example, the character's work address in terms of existing knowledge about that place that can be stored and accessed independently of knowledge about the character who works there.

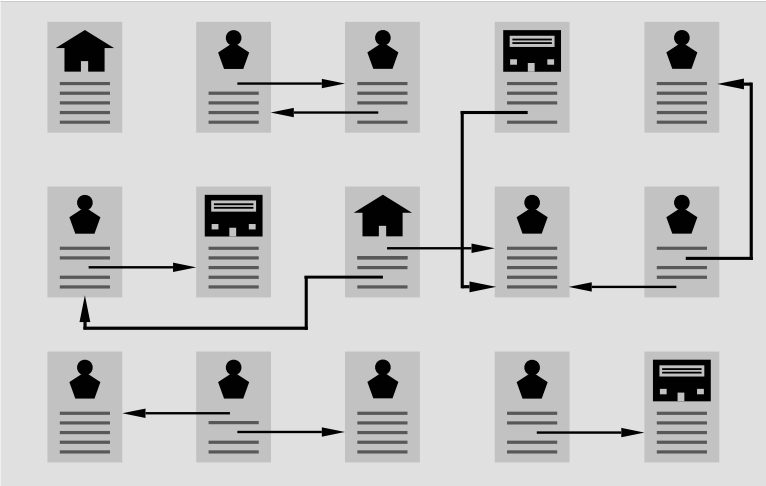

Figure 1 An illustration of the *ontological* structure of character knowledge: characters build up mental models of the hundreds of people, homes, and businesses in their towns, each of which might include pointers to other mental models (*e.g.*, a belief about a character's workplace will resolve to a pointer to the mental model for that business).

3.4 Mental Models

Characters in *Talk of the Town* form *mental models* about the residents and places in their towns. Each character mental model pertains to a specific individual entity and is structured as a list of *belief facets* that correspond to individual attributes of that entity. A given character attribute will have a *type* (*e.g.*, hair color) and a ground-truth value (*e.g.*, `brown`), but a belief facet corresponding to it will represent a character's potentially false belief about that attribute (*e.g.*, `black`). The attribute types that we have implemented for *Talk of the Town* match the domain and central concerns of the game and are as follows:

- For mental models of characters:
    - **Status.** Condition (`alive` or `dead`), year of departure from the town (if any; *e.g.*, `1972`), marital status (`single`, `married`, `divorced`, or `widowed`).
    - **Age.** Birth year (*e.g.*, `1941`), death year (if any), approximate age (*e.g.*, `30s`).
    - **Name.** First name, middle name, last name, suffix, surname ethnicity (*e.g.*, `German`), whether surname is hyphenated.
    - **Appearance.** Each of the 24 facial attributes that we model (*e.g.*, hair color).
    - **Occupation.** Company (links to mental model of that place), job title (*e.g.*, `bartender`), shift (`day` or `night`), status (`retired`, `employed`, or `unemployed`).
    - **Home.** Home (either an apartment unit or house; links to mental model of that place).
    - **Whereabouts.** Where a person was on a given day or night (links to mental model of that place). This facet is central to the *Talk of the Town* game

design, since the lover character is known to have been at the home of the central character on the night of the death.

- For mental models of businesses/homes:
    - **Employees/Residents.** Listing of its employees/residents (each links to mental model of a character).
    - **Apartment.** Whether it's an apartment unit (for homes only).
    - **Block.** For example, `800 block of Lake Street`.
    - **Address.** For instance, `613 Fillmore Street`.

We would like to emphasize that these example facet types are only meant to serve as examples, as our method is agnostic to the type of knowledge that it's used to represent. Each facet is structured as a collection of data about the belief. In addition to its *owner* (the character who has constructed the mental model), *subject* (the entity to whom it pertains), and facet type, this data includes:

- **Value.** A representation of the belief itself, *e.g.*, the string `brown` for a belief facet pertaining to hair color, or the integer `1944` for a facet corresponding to a character's birth year.
- **Mental Model.** If the value of this facet resolves to another entity for whom the owner of this facet has formed a mental model, this will point to that mental model. This is how the linking that we have mentioned in earlier examples is handled.
- **Predecessor.** The belief facet that the owner previously held, if any. This allows the system to track supplanted or forgotten character knowledge. Since a given chain of predecessors represents a perfect history of an NPC's beliefs about some attribute, we do not plan to give NPCs access to this data.
- **Parents.** If this knowledge originated in information from other characters, this will point to the belief facets of those characters that spawned this current facet. This allows the system to trace the history and trajectory of any piece of information.
- **Evidence.** A list of the pieces of evidence by which the owner of this facet formed and continues to substantiate it; evidence may accumulate as the simulation proceeds. In Section 3.5, we outline our evidence typology.
- **Strength.** The strength of this particular belief. This is the sum of the strength of all pieces of evidence supporting it, the determination of which we also explain in Section 3.5.
- **Accuracy.** Whether or not the belief is accurate (with regard to the *current* true state of the world).

3.5 Evidence

All character knowledge is formed in response to evidence, and may also propagate, deteriorate, or terminate in ways that can be described using pieces of evidence. We will illustrate these details by explaining our evidence typology, which comprises eleven *types*

across five categories. This is the most important part of this chapter.

- How knowledge originates:
  - **Reflection.** A reflection represents the case of a character inherently knowing something about himself or herself. We don't spend any computation on actually simulating this phenomenon.
  - **Observation.** When a character directly observes a person or place, he or she may form knowledge about attributes of that entity. Whether knowledge is formed about a particular attribute depends on the salience of the entity and the attribute type, which we explain in Section 3.6.
  - **Transference.** If one entity reminds a character of another entity (determined by feature overlap between his or her respective mental models of them), he or she may unconsciously copy beliefs about one to the mental model of the other.
  - **Confabulation.** By confabulation, a character *unintentionally* concocts new knowledge about some entity; this happens probabilistically. The particular belief-facet value that gets confabulated is determined probabilistically according to the distribution of that feature type in the town. For instance, if 45% of characters in the town have black hair, then confabulation of a belief about hair color would have a 45% chance of producing the value `black`.
  - **Lie.** A lie occurs when an NPC *intentionally* conveys information to another character that she herself does not believe. We call this a type of origination (and not propagation) because the knowledge in question is *invented* by virtue of the lie—*i.e.*, no existing knowledge is propagated by the lie.
  - **Implant.** For efficiency reasons, some essential character knowledge will be directly implanted in character minds at the end of world generation, and thus will have no explicit point of origination. We discuss knowledge implantation in depth in Section 3.10.
- How knowledge reinforces itself:
  - **Declaration.** Whenever a character delivers a statement, the strength of his or her own belief (being declared by the statement) will slightly increase. That is, the more a person retells some belief, the stronger that belief becomes for him or her, which is realistic [Wilson 85]. By this mechanic, an NPC who frequently tells the same lie might come to actually believe it.
- How knowledge deteriorates:
  - **Mutation.** As an operationalization of memory fallibility, knowledge may mutate over time. We explain this in detail in Section 3.8.
- How knowledge terminates:
  - **Forgetting.** To further incorporate memory fallibility, knowledge may be forgotten due to time passing; this is affected by a character's memory attribute and the salience of the facet subject and type.

Characters are not consciously aware of transferences, confabulations, or mutations,

and recipients (and eavesdroppers) of lies treat them as statements. That is, the recipient will reason about a lie as if it were a statement (and so the strength of a lie, as a piece of evidence, is equal to that of a statement), but the system will still track that it was in fact a lie, to allow for the presentation of true knowledge trajectories after gameplay. Additionally, each piece of evidence has metadata of the following types:

- **Source.** With a statement, lie, or eavesdropping, this specifies the character who delivered the information. This allows the system to trace the history and trajectory of any piece of information, which is a design goal.
- **Location.** Where the piece of evidence originated (*e.g.*, where an observation or statement took place).
- **Time.** The timestep a piece of evidence originated (either a day or night of a particular date).
- **Strength.** The strength of a piece of evidence is a floating-point value that is determined by its type (*e.g.*, a statement is weaker than an observation) and decays as time passes. In the case of statements, lies, and eavesdroppings, the strength of a piece of evidence is also affected by the affinity its owner has for its source and the strength of that source's own belief at the time of propagation.

3.6 Salience Computation

When a character observes some entity in the simulation, a procedure is enacted that determines, for each perceptible attribute of the observed entity (as defined in Section 3.2), the probability that the character will remember what he or she saw; this procedure crucially depends on the *salience* of the entity and attribute being observed. Salience computation in *Talk of the Town* considers the relationship of an observed character (subject) to the observer (*e.g.*, a coworker is more salient than a stranger), the extent of the observer's friendship with the subject, the strength of the observer's romantic feelings toward the subject, and finally the subject's job level (characters with more elevated job positions are treated as more salient). For places, salience computation currently only considers whether the observing character lives or works at the observed place. Additionally, our salience-computation procedures consult a hand-authored knowledgebase specifying attribute saliences—this was one of the requirements listed in Section 3.2. Salience is also used to determine the probability that a character will misremember or altogether forget (on some later timestep) knowledge pertaining to a given subject and attribute—here, the probability of memory deterioration decreases as these saliences grow larger.

3.7 Knowledge Propagation

The salience of the subject and attribute type of a piece of information also affects whether a character will pass it on (via a statement, defined in Section 3.5). Currently, what subjects of conversation come up in an interaction between two *conversants* is determined by the salience of all entities that either of them know about (*i.e.*, the sum salience to both

conversants). The *n* highest-scoring[3] entities are then brought up in conversation, with *n* being determined by the strength of the conversants' relationship and also their respective *extroversion* personality components. This is illustrated in Figure 2. For each subject of conversation, the conversants will exchange information about individual attributes of that subject at probabilities determined by the salience of each attribute type. Because a character may bring up subjects that an interlocutor does not (yet) know about, our propagation mechanism allows characters to learn about other people and places that they have never encountered themselves. It's even possible for a character to learn about another character who died before he or she was born; this often occurs when parents tell their children about deceased relatives (who score highly in salience computation due to being related to both conversants).

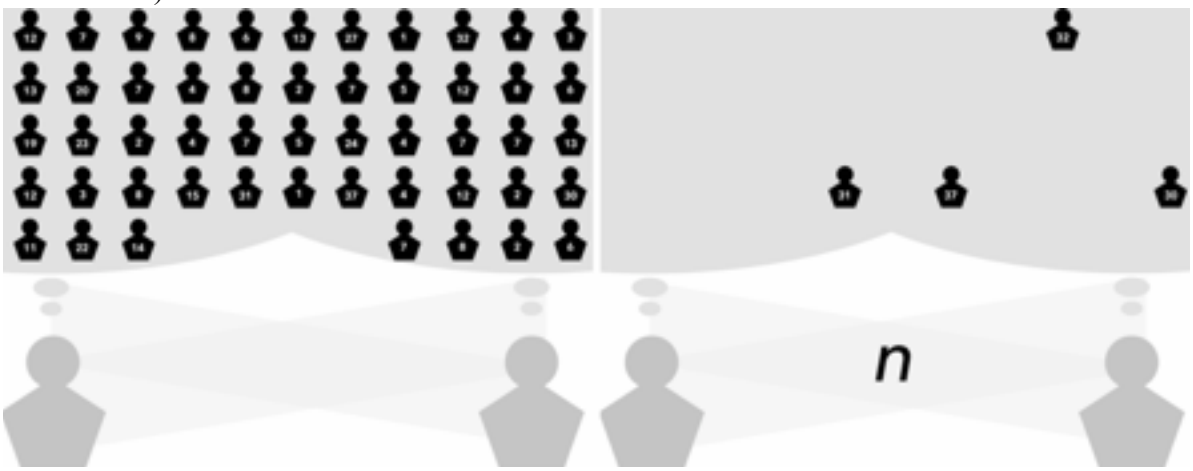

Figure 2 An illustration of the procedure by which NPCs decide whom to exchange information about. A pair of conversants score everyone they know about for their saliences to them both, and then select the *n* highest scoring (where *n* is determined by the characters' relationship and personalities) to be the subjects of their knowledge exchange.

3.8 Fallibility Modeling

As an operationalization of memory fallibility, characters may adopt false beliefs for reasons other than lies. On each timestep after world generation, the four phenomena associated with memory fallibility—transference, confabulation, mutation, and forgetting (see Section 3.5)—are probabilistically triggered for character mental models. When this happens, the facets of that mental model are targeted for deterioration at probabilities determined by the character's memory attribute (modeled as a floating-point value inherited from a parent), the facet type (*e.g.*, a whereabouts belief will be more likely to mutate than a first name belief), and the strength of the existing belief (weaker beliefs are more likely to deteriorate). For mutation, the system relies on a handcrafted schema that specifies for a given facet value the probabilities of it mutating to each other viable facet value. Figure 3 shows excerpts from

[3] For greater efficiency, we amortize this computation by keeping track of all pairwise character salience scores as relevant changes to the social state occur.

this schema, which we call a *belief mutation graph*; earlier, in Section 3.2, we noted the specification of this graph as an architectural requirement.

Figure 3 Illustrative excerpts from our hand-authored *belief mutation graph*. Probabilities specify how particular beliefs about hair color (left) and facial-hair style (right) might mutate (to model character misremembering).

3.9 Belief Revision

Currently, an NPC will always adopt a new belief upon encountering a first piece of evidence supporting it, assuming there is no current belief that it would replace. As a character accumulates further evidence supporting his or her belief, its strength will increase commensurately to the strength of the new evidence. As noted in Section 3.5, the strength of a piece of evidence depends on its type; if the evidence has a source, its strength will also depend on the affinity that its recipient has for that character and the strength of the corresponding belief held by the source. If at any time an NPC encounters new evidence that contradicts his or her currently held belief (*i.e.*, supports a different belief-facet value), the character will consider the strength of the new evidence relative to the strength of his or her current belief. If the new evidence is stronger, she will adopt the new belief that it supports; if it's weaker, she will *not* adopt a new belief, but will still keep track of the other *candidate belief* and the evidence for it that she had encountered. If she continues to encounter evidence supporting the candidate belief, she will update its strength accordingly and if at any time that strength exceeds the strength of the currently held belief, the NPC will adopt the candidate belief and relegate the previously held belief to candidate status. Belief oscillation is possible, as such, but that's an appeal of the design. For an example illustrating how this procedure works, see our earlier paper on the system [Ryan 15].

3.10 Knowledge Implantation

In our architecture, procedures related to character knowledge phenomena are expensive. If we enacted them during *world generation*—the period of game time that is simulated prior to gameplay and spans from the town's founding in 1839 up to 1979 (see Section 2.2)—we would spend a lot of computation simulating the knowledge of hundreds of characters who would have died long before the period of gameplay. Instead, world generation employs all aspects of the simulation *besides* ones related to character knowledge (*e.g.*, characters forming relationships, starting businesses) and then terminates one week prior to the death

of the central character (the event that kicks off gameplay, as explained in Section 2.1). At this point, however, living characters have no knowledge at all—to resolve this, the system employs a procedure that *implants* into each character's mind the knowledge that would believably be ingrained in them. This procedure is illustrated in Listing 1.

Listing 1 Pseudocode for our *knowledge implantation* procedure, which is carried out at the end of world generation.

```
for resident of town
    implants = []
    for immediate family member of resident
        add immediate family member to implants
    for friend of resident
        add friend to implants
    for neighbor of resident
        add neighbor to implants
    for coworker of resident
        add coworker to implants
    for every other character who has ever lived
        chance = 1.0 - (1.0/salience of that character)
        if random number < chance
            add other character to implants
    for character in implants
        for attribute of character
            chance = attribute salience
            chance += -1.0/salience of that character
            if random number < chance
                have resident adopt accurate belief for
attribute
```

3.11 Core Procedure

After implanting character knowledge, we simulate all character activity, including all knowledge phenomena, for one week of game time (up to the death of the central character). Listing 2 shows the operation of this method, with a focus on knowledge-related activity. A variant of the loop is also carried out during gameplay, between player turns.

Listing 2 High-level pseudocode for the core procedure of our simulation of character knowledge phenomena.

```
do world generation  // See Section 2.2
do belief implantation  // See Listing 1
while central character still alive  // One week of game time
    advance one timestep
    for resident in town
        enact resident routine  // Puts them somewhere in the
```

```
town
    for resident in town
        for nearby character at same place as resident
            if characters will interact  // See Section 2.2
                do knowledge exchange  // See Section 3.7
    for resident in town
        do simulation of fallibility phenomena // See Section
3.8
```

3.12 Tunable Parameters

Our approach has very many parameters that can be tuned for both authorial control and greater computational efficiency. In our implementation, belief mutation is quite expensive, in part because we represent many facet values as strings. While this aspect of our implementation is currently efficient enough for our needs, it presents a basic opportunity for optimization. Beyond this, mutation rates could simply be turned down (as a way of targeting either greater efficiency or gameworlds with more accurate character beliefs). Other tunable parameters include the salience of characters and attributes; the probabilities of social interactions, eavesdropping, lying, and different kinds of knowledge deterioration; the base strengths of each type of evidence; and more. We currently do have worries about the complexity of interaction between all these parameters; eventually, we may end up simplifying some of these systems.

3.13 Some Statistics

A typical *Talk of the Town* town will be inhabited by between 300-500 NPCs, each of which will maintain approximately 250-400 mental models; some highly extroverted characters will have as many 500-600 mental models. Across all her mental models, a typical character will own around 800-1200 belief facets by the time gameplay begins. The entire world-generation procedure lasts (on the order of) a few minutes, and the simulation of knowledge phenomena between turns takes about a minute; we expect these durations to decrease as we begin to explore optimization strategies closer to release.

4 Discussion

Talk of the Town gameplay would not be possible without the architecture we have presented here. First, as we stated above, a fundamental goal of ours is to provide a gameworld that feels like it's sprawling with rich NPCs who each have unique subjective experiences that are captured by the particular beliefs that they have adopted. If we were to model information flow abstractly, as other gossip systems have done, we would lose all this depth and complexity. Further, we want dialogue with NPCs to be a core gameplay interaction—with a coarser, more abstract representation of character knowledge, there wouldn't be much for characters to say. Beyond supporting our intended gameplay aesthetic, our architecture critically enables the kind of player strategies that we want to support, which were also detailed above. First, we want the family-member player to be able to home

in on the identity of the lover by investigating apparent sources of deliberate misinformation. This requires information flow to be modeled at a considerable level of fidelity, and it also necessitates the possibility of misinformation propagating from specific sources at specific times. Further, we want the lover to be able to tactfully spread *some* amount of information without making her identity obvious, but hitting this sweet spot could be tough with coarser-grained modeling of information flow. With our approach, we're more confident about signals of deliberate misinformation being readable to both players, since knowledge propagation is modeled at the agent level; intuitively, this fidelity of simulation is more likely to match player expectations than abstract models. Still, we want it to be possible for the lover to be successful at clandestinely propagating misinformation—this requires that benign misinformation also be present in the gameworld, which our architecture supports through its simulation of memory fallibility. Moreover, we wish to support a specific tactic that may underpin the lover's larger strategy of spreading misinformation: the ability to propagate *false flag* sources for her own misinformation, meaning characters whom she claims told her what are in fact her own lies. False flags are easily represented in our architecture as metadata attached to discrete character beliefs—namely the *source*, *location*, and *time* attributes of pieces of evidence supporting an NPC's belief—that can be surfaced in generated dialogue, *e.g.*, "Gary Stuart told me last night at the 3rd Street Diner."

Because our game is still in development, we can't speak conclusively yet about the success of our architecture from an authorial standpoint. One potentially huge challenge that we anticipate involves balancing the architecture's considerable array of tunable parameters —attribute saliences, mutation rates, etc. While *Talk of the Town* isn't fully implemented yet, we have actually already used this framework in a completed mixed-reality experience called *Bad News* [Ryan 16b]. Over the course of performing this piece dozens of times, we've encountered thousands of generated character beliefs. While these beliefs have generally appeared to be well-formed and believable, in early performances we noticed a prevalence of misremembered home addresses, including cases where characters couldn't remember where their own parents lived. To fix this, we simply turned down the mutation rate for this attribute, which seemed to be a good preliminary indication of the prospects for authorial control in the face of so many tunable parameters. As another fundamental limitation, our method is not very computationally efficient, though in Section 3.12 we named a few opportunities for optimization. For *Talk of the Town*, we aren't particularly worried about this, since the heavy computation takes place prior to gameplay (generating the town) and between player turns (simulating knowledge phenomena).

It's not easy for us to articulate how this architecture could be utilized for games in mainstream genres. We do think there are probably viable opportunities for this (at least in cases where computational efficiency is not a major concern), but we're more excited about fundamentally new kinds of gameplay experiences that could be made possible by our architecture. Here, we hope that *Talk of the Town*, once completed, will do well to demonstrate the appeal of gameplay surrounding the character knowledge phenomena

whose simulation we have described in this chapter.

Acknowledgments

We would like to thank Damián Isla for invaluable feedback that greatly improved this chapter.

References

[Adams 15] Adams, T. 2015. Simulation principles from Dwarf Fortress. In *Game AI Pro²: Collected Wisdom of Game AI Professionals*, ed. S. Rabin, 519-522. Boca Raton: CRC Press.
[BioWare 02] BioWare. 2002. *Neverwinter Nights*. New York: Infogrames/Atari.
[Crawford 04] Crawford, C. 2004. *Chris Crawford on interactive storytelling*. Berkeley: New Riders Press.
[Crawford 07] Crawford, C. 2007. Deikto: A language for interactive storytelling. In *Second Person: Role-Playing and Story in Games and Playable Media*, ed. P. Harrigan and N. Wardrip-Fruin. Cambridge: MIT Press.
[Diller 04] Diller, D.E., W. Ferguson, A.M. Leung, B. Benyo, and D. Foley. 2004. Behavior modeling in commercial games. *Behavior Representation in Modeling and Simulation*.
[Evans 14] Evans, R. and E. Short. 2014. Versu—A simulationist storytelling system. *IEEE Transactions on Computational Intelligence and AI in Games* 6(2):113-130.
[Exxos 88] Exxos (ERE Informatique). 1988. *Captain Blood*. Lyon: Infogrames.
[Isla 13] Isla, D. 2013. Third Eye Crime: Building a stealth game around occupancy maps. *Proceedings of the 9th Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.
[Leonard 03] Leonard, T. 2003. Building an AI sensory system: Examining the design of Thief: The Dark Project. *Game Developers Conference*.
[Mark 09] Mark, D. 2009. *Behavioral mathematics for game AI.* Boston: Cengage Learning PTR.
[Mateas 04] Mateas, M. and A. Stern. 2004. Natural language understanding in Façade: Surface-text processing. *Proceedings of the 2nd International Conference on Technologies for Interactive Digital Storytelling and Entertainment*:3-13. Springer Berlin Heidelberg.
[Palmer 04] Palmer, A. 2004. *Fictional minds*. Lincoln: University of Nebraska Press.
[Ryan 15] Ryan, J.O., A. Summerville, M. Mateas, and N. Wardrip-Fruin. 2016. Toward characters who observe, tell, misremember, and lie. *Proceedings of the 2nd Workshop on Experimental AI in Games*.
[Ryan 16a] Ryan, J., M. Mateas, and N. Wardrip-Fruin. 2016. A lightweight videogame dialogue manager. *Proceedings of the 1st Joint International Conference of DiGRA and FDG*.
[Ryan 16b] Ryan, J.O., A.J. Summerville, and B. Samuel. 2016. Bad News: A game of death and communication. *Proceedings of the 2016 CHI Conference Extended Abstracts on*

Human Factors in Computing Systems:160-163.
[Ryan 16c] Ryan, J., M. Mateas, and N. Wardrip-Fruin. 2016. A simple method for evolving large character social networks. *Proceedings of the 5th Workshop on Social Believability in Games*.
[Ryan 16d] Ryan, J., M. Mateas, and N. Wardrip-Fruin. 2016. Characters who speak their minds: Dialogue generation in Talk of the Town. *Proceedings of the 12th Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.
[Ryan 16e] Ryan, J., M. Mateas, and N. Wardrip-Fruin. 2016. Generative character conversations for background believability and storytelling". *Proceedings of the 5th Workshop on Social Believability in Games*.
[Summerville 16] Summerville, A.J., J. Ryan, M. Mateas, and N. Wardrip-Fruin. 2016. CFGs-2-NLU: Sequence-to-sequence learning for mapping utterances to semantics and pragmatics. Technical Report UCSC-SOE-16-11.
[Team Bondi 11]. Team Bondi. 2011. *L.A. Noire*. New York: Rockstar Games.
[Walsh 15]. Walsh, M. Modeling perception and awareness in Tom Clancy's Splinter Cell Blacklist. *Game AI Pro²*:313-326.
[Welsh 13] Welsh, R. 2013. Crytek's Target Tracks Perception System. *Game AI Pro*: 403:411.
[Wilson 85] Wilson, R.M., L.B. Gambrell, and W.R. Pfeiffer. 1985. The effects of retelling upon reading comprehension and recall of text information. *Journal of Educational Research* 78(4):216-220.

Biography

James Ryan is a PhD student in computer science at the University of California, Santa Cruz, working with the Expressive Intelligence Studio. He earned his BA in linguistics and MS in health informatics (with a minor in cognitive science) from the University of Minnesota. His current research agenda spans two main topics: building autonomous agents who construct personal narratives out of their subjective experience in simulated worlds, and developing new technologies for freeform conversational interaction in games (by integrating systems for dialogue management, natural language generation, and natural language understanding).

Michael Mateas is the codirector of Expressive Intelligence Studio and director of the Center for Games and Playable Media at the University of California, Santa Cruz. His research in game AI focuses on enabling new forms of gameplay through innovative AI solutions. The Expressive Intelligence Studio has ongoing projects in autonomous characters, interactive storytelling, game design support systems, AI models of creativity, and automated game generation. With Andrew Stern, Michael created *Façade*, which uses

AI techniques to combine rich autonomous characters with interactive plot control to create the world's first, fully-produced, real-time, interactive drama. Michael received his PhD in computer science from Carnegie Mellon University.