How to do dialogue in a fairy-tale world

Johan Boye and Joakim Gustafson
TeliaSonera R&D, Sweden
johan.boye@teliasonera.com, joakim.gustafson@teliasonera.com

1 Introduction

The work presented in this paper is an endeavor to create a prototype of a computer game with spoken dialogue capabilities. Advanced spoken dialogue has the potential to considerably enrich computer games, where it for example would allow players to refer to past events and to objects currently not visible on the screen. It would also allaow users to interact socially and to negotiate solutions with the game characters. The game takes place in a fairy-tale world, and features two different fairy-tale characters, who can interact with the player and with each other using spoken dialogue. The fairy-tale characters are separate entities in the sense that each character has its own set of goals and its own perception of the world. This paper gives an overview of the functionality of the implemented dialogue manager in the NICE fairy-tale game system.

2 Game scenario

The NICE fairy-tale game system takes place in the fairy-tale world of HC Andersen. The fairy-tale world is a large 3D virtual world. Five animated fairy-tale characters have been developed, out of which three have been used in the dialogue game so far (Gustafson et al 2005). Charles and Cavazza (2004) distinguish between two types of characters in their character-based story telling system – *feature characters* and *supporting characters*. In the fairy-tale game, a third kind of character has been added - a *helper character*. Cloddy Hans is a friendly helper, that is, a character that guides and helps the user throughout the whole fairy-tale game. He has no goals except doing what the user asks him to, and helping the user along when necessary. Helper characters need conversational capabilities allowing both for grounding and cooperation, and for dialogue regulation and error handling. They need to have knowledge of all plots and subtasks in the game. Finally they need simple visual perception so that they can suggest actions that involves objects in the scene that the user have not noticed yet, and they have to be aware of the other characters actions as well as of their verbal output.

The player is introduced to Cloddy Hans in a training scene which takes place in HC Andersen's fairy-tale laboratory where their task is to get Cloddy Hans to build the fairy-tale by putting fairy-tale objects into the right slot of a fairy-tale machine. The task is deliberately simple and repetitive in order for the users to acquaint themselves with the capabilities and limitations of spoken input understanding. Thumbelina is a non-verbal supporting character who points at slots in the machine where she wants an object to be placed, and if the user gets Cloddy Hans to put it in another slot she shows her discontent with large emotional body gestures. When Cloddy Hans pulls the machine's lever a hidden trap door opens below Cloddy Hans and he falls into the fairy-tale world (see Figure 1).

Cloddy Hans tells the user that they have landed in the fairy-tale world, and soon they encounter their first problem. Together with Cloddy Hans, the user is trapped on a small island, from which he can see the marvels of the fairy-tale world – houses, fields, a wind mill, etc. – but they are all out of reach. A deep gap separates him from these wonders. There is a drawbridge, which can be used for the crossing, but it is raised, and the mechanism which operates it is on the other side. Fortunately, a girl, Karen, is standing on the other side (Figure 1). She has a different kind of personality compared to Cloddy Hans. Instead of having Cloddy Hans's positive attitude, she is sullen and uncooperative, and refuses to close the drawbridge. The key to solving this deadlock is for the player to find out that Karen will comply if she is paid: she wants to have one of the fairy-tale objects that are lying in the grass on the player's side of the gap. Thus, it is the task of the player to find the appropriate object, and use this object to bargain with Karen.

Figure 1. Karen and Cloddy Hans at the drawbridge

3 World and task representation

In order to be able to behave in a convincing way, a fairy-tale character needs be aware of other characters and all physical things in its vicinity. The character also needs to understand how past events and interactions influence the current situation. To this end, each fairy-tale character has an inner state consisting of

- a *world model*, representing the character's beliefs about the state of things and characters in the world, as a set of interrelated objects;
- a *discourse history*, representing past interactions;
- an *agenda*, a set of tree-structures representing the character's current goals, past and future actions, and their causal relations.

A task specification is a set of declarative rules encoding relationships between actions and propositions concerning the state in the world. The dialogue manager uses the rules to build agenda tree-structures that encode current goals, as well as the causal relationships between the current goals and past and future actions. The agenda trees constitute the main driving force for a fairy-tale character's behavior, and the trees can at times attain great depth and complexity. Note that the tree also represents the causal relations: Cloddy Hans is walking over to the shelf because he wants to stand there, and he wants to stand there because he wants to hold the axe. These relations can be verbalized, enabling Cloddy Hans to explain to the user what he is up to and why.

This approach is highly reminiscent of traditional STRIPS-like planning (Fikes and Nilsson 1971). The most important difference is non-monotonicity in the sense that true propositions do not necessarily stay true. For instance, the proposition available(axe) may turn from true to false because of unexpected changes in the environment (some other character might take the axe). For this reason, the system checks all necessary preconditions before executing any action, also those preconditions that have been found to be true at an earlier point in time.

4 The characters' interactions

4.1 Input and output messages

The dialogue manager receives a stream of input messages and generates a stream of output messages. The output messages it can produce are of two types:

convey *<dialogue act>*: The dialogue act will be turned into words by the Natural Language Generation module, and then an utterance will be produced using speech synthesis and animation.

perform *<action>*: A command to perform the action is sent to the Animation Planner.

Input messages belong to one of the following types:

parserInput *<dialogue_act>*: The user has said something, and <*dialogue act*> is the representation of that utterance.

gestureInput *<object>*: The user has made a graphical reference to a specific object.

recognitionFailure: The user has said something the recognizer could not interpret.

broadcast *<message>*: Another fairy-tale character has said or done something. <*message*> is either a dialogue act or an action.

performed *<id>* *<flag>*: The character itself has completed a specific action which had previously been requested by a perform message. <flag> is either "ok" or "failed" depending on whether the action was carried out or not.

trigger *<id>*: The system has detected that the character has moved into a trigger with <*id*> (see section 4.4).

It might seem odd that the performed message is needed. After all, the character had itself requested the action to be performed. The reason is that due to the fact that some actions take considerable time to carry out, the character cannot consider an action as completed before it has actually been carried out in full on the screen. Moreover, requested actions may fail for a variety of reasons. The message is a feedback to the character that the action has been carried out without problems, so that the character can turn to the next action on the agenda.

4.2 Dialogue acts

Utterances are represented by tree-strucured expressions, called *dialogue acts* (Boye et al 2006). The kind of user utterances the system can interpret can be categorized as follows:

Instructions: "*Go to the drawbridge*", "*Pick it up*"…

Domain questions: "*What is that red object?*", "*How old are you?*"…

Giving information: "*I'm fourteen years old*"…

Negotiating utterances: "*What do you want in return?*", "*I can give you the ruby if you lower the bridge*"…

Confirmations: "*Yes please!*", "*Ok, do that*"…

Disconfirmations: "*No!*", "*Stop*!", "*I didn't say that!*"…

Problem reports and requests for help: "*Help!*", "*What can I do?*", "*Do you hear me?*"…

Requests for explanation: "*Why did you say that?*", "*Why are you doing this?*"…

The fairy-tale characters use an overlapping but not completely identical set of classes of utterance:

Responses to instructions: either **accepting** them ("*OK, I'll do that*") or **rejecting** them, ("*No I won't open the drawbridge!*", "*The knife already is in the machine*").

Answers to questions: "*The ruby is red*", "*The knife is on the shelf*", "*I am 30 years old*"…

Stating intentions: "*I'm going to the drawbridge now*"…

Confirmation questions: to check that the system has got it right, e.g. "*So you want me to go to the shelf?*"

Clarification questions: when the system has incomplete information, e.g. "*Where do you want me to go?*", "*What should I put on the shelf?*"...

Suggestions: for future courses of action, e.g. "*Perhaps we should go over to the drawbridge now?*".

Negotiating utterances: "*I won't do that for nothing*", "*What a piece of junk! Find something better*"...

Explanations: "Because I want the axe in the machine".

4.3 Turn-taking

The fairy-tale character with whom the player is talking is always in camera (i.e. is shown on the screen). The player can control the camera by saying the name of a character. For example, by saying "*Cloddy*", the camera pans over to show Cloddy Hans. This is also the way for the player to change dialogue partner.

The system can also initiate a camera change and a change of dialogue partner, by triggering on certain events. For instance, whenever Cloddy Hans reaches the gap, the camera automatically pans over to show Karen, and Karen starts talking. There is also a possibility for a character to make side-comments (without being in camera). Cloddy Hans is able to trigger on certain utterances by Karen to provide hints to the user ("*Maybe she will lower the bridge if we give her something nice*", "*Girls like shiny things, don't they?*", or commenting on what Karin said like "*she is a bit grumpy today!*").

4.4 Triggers

A *trigger* is a three-dimensional area with specific coordinates in the 3D virtual world. Whenever a character moves into a trigger (or out of a trigger), a message is generated and sent to the corresponding dialogue manager, which can be made to react on the event. A typical use for triggers is to make the character turn its head or make an utterance when it passes an object of interest. Triggers are also used in the fairy-tale world to generate walk paths between locations that are far apart.

5 Scenes

There are two instances of the dialogue manager in the fairy-tale game system, one per fairy-tale character. The functionality of these two dialogue managers are somewhat different, reflecting the fact that the personalities of the two fairy-tale characters are supposed to be different. Moreover, the functionality of any dialogue manager varies over time, reflecting supposed changes in the characters' knowledge, attitudes and state of mind. However, when considered at an appropriate level of abstraction, most of the functions any dialogue manager needs to be able to carry out remain constant regardless of the character or the situation at hand. As a consequence, the dialogue management software in the NICE fairy-tale system consists of a *kernel* laying down the common functionality, and *scripting code* modifying the dialogue behaviour as to be suitable for different characters and different situations. Such a model of code organization is common in the computer games field (see e.g. Varanese and LaMothe 2003). In the spoken dialogue systems field, it is desirable for both practical and theoretical reasons. On the practical side, it allows for the development of systems that are simpler to understand and maintain. On the theoretical side, it helps distinguishing the dialogue management concepts that are actually generic from those that are situation- or character-specific. Such knowledge can then increase our understanding of dialogue in general.

As mentioned the NICE fairy-tale game is divided into different scenes. These scenes may be divided into subscenes, the subscenes further divided into sub-subscenes, and so on. From a dramatic point of view, a (sub)scene can be thought of as a element of the overall plot, and the transition from one scene to another marks the passing of some significant event (for instance, the introduction of a new character, or the change of locale). From a gaming point of view, a scene on the top-level corresponds to a level (in the gaming sense), each new level introducing a new environment and a new set of problems. In any case, there is a need for a method of defining scenes and subscenes in a modular way, so that new scenes and subscenes can be added to the system without the need to modify the dialogue management kernel. Here, "adding a scene" should mean modifying the behaviour, or adding new behaviour, to the characters participating therein. Thus we need to find primitives at an appropriate level of abstraction, in which to express this modified behaviour. Our solution is based on the concept of a dialogue event, (see Section 6).

The second scene (in the fairy-tale world) is divided into four subscenes:

- *Introduction*, in which Cloddy Hans tells the user a few things about the fairy-tale world.
- *Exploration*, in which the player and Cloddy Hans explore the world together.
- *Negotiation* starts when the player meets Karen for the first time, and tries to persuade her to lower the drawbridge.
- Finally, the *bridge crossing* subscene takes place after successful negotiation with Karen.

At any moment there is exactly one active phase in the game, and this active phase changes occasionally according to some algorithm. In the scene above, the phases were arranged in a predefined sequence, but this needs not be the case. The scene might change according to which geographic location the player chooses to visit, or because of the action of one of the characters in a scene (that either was initiated by the user or that was initiated by the character itself)

6 Dialogue events

The dialogue management kernel issues dialogue events at important points in the processing. Some kinds of dialogue events, the so-called external events, are triggered from an event in a module outside the dialogue manager (for instance, a recognition failure in the speech recognizer), whereas the internal events take place within the dialogue kernel. Dialogue events can be caught by the scripting code by use of a callback procedure.

As an example, if the speech recognizer detects that the user is speaking but cannot recognize any words, it sends a "recognition failure" message to the dialogue manager. The dialogue management kernel receives this message, generates a RecognitionFailureEvent, and calls the onDialogueEvent procedure of the current scene. The current scene may then re-direct the procedure call to its current phase. In this way, different pieces of scripting code can be provided for different characters, scenes and phases, facilitating the creation of different personalities and scene-dependent behaviour in a modular systematic way. As for external dialogue event, there is one type of dialogue event for each input message that the dialogue manager can receive (see section 4.1), i.e.

BroadcastEvent: Some other character has said and done something.

GestureEvent: The Gesture Interpreter has recognized a gesture, and found one or several objects gestured at.

ParserEvent: The parser has arrived at an analysis of the latest utterance.

PerformedEvent: The animation system has completed an operation, either an utterance or an action such as goTo, pickUp etc.

RecognitionFailureEvent : The speech recognizer has detected that the user has said something, but failed to recognize it.

WorldEvent: An event has occurred in the world (e.g. the drawbridge has changed position, or an object has been inserted into one of the slots of the fairy-tale machine).

TriggerEvent: The animation system has detected that the character has moved into a trigger.

Similarly, there are internal dialogue events:

AlreadySatisfiedEvent: A goal which already is satisfied has been added to the character's agenda.

CannotSolveEvent: An unsolvable goal has been added to the character's agenda.

IntentionEvent: The character has an intention to say or do something.

NoReactionEvent: The character has nothing on the agenda.

PossibleGoalConflictEvent: A goal is added to the agenda, but the agenda contains a possibly conflicting goal.

TimeOutEvent: A timeout has expired.

Internal events such as RequestEvent, QuestionEvent are also generated as an effect of specific dialogue acts made by the user (e.g. Request, Question).

The kernel provides a number of operations through which the scripting code can influence the dialogue behaviour of the character. These are:

- *interpret an utterance* in its context
- *convey* a dialogue act
- *perform* an action
- *add a goal* to the character's agenda
- *remove a goal* from the character's agenda
- *find the next goal* on the agenda, and pursue it

The convey operation ultimately leads to an utterance with accompanying gestures from the character (via text generation, graphics generation, and speech synthesis). The perform operation ultimately leads to an action being performed by the animated character.

The interplay between the instructions in the scripting code and the dialogue events generated by the dialogue management kernel creates the overall dialogue behaviour of the character. For instance, consider the case where the user requests Cloddy Hans to "*Go to the fairy-tale machine*". This would lead to the following sequence of events:

1. A message from the parser arrives and generates a ParserEvent.
2. The ParserEvent is caught by the scripting code of the current scene, which calls the contextual interpretation procedure of the dialogue kernel.
3. Contextual interpretation establishes that the user's utterance is a request from the user for Cloddy Hans to go to a specific spot (the fairy-tale machine, in this case). A RequestEvent is generated.
4. The RequestEvent is caught by the scripting code, which calls convey to produce an utterance acknowledging the request, and then adds to Cloddy Hans's agenda the goal that he should be standing next to the fairy-tale machine.
5. When Cloddy Hans has eventually reached his destination, a message arrives from the animation system. This message generates a PerformedEvent, which can again be caught to produce a new utterance from Cloddy Hans, etc.

This event-driven model allows for asynchronous dialogue behaviour (see e.g. Boye et al 2000). This means that a character in the fairy-tale system is not confined to a model where the user and character have to speak in alternation. Rather, a character may take the turn and start speaking for a number of reasons: because the user has said something (RecognitionFailureEvent or a ParserEvent), because some other fairy-tale character has said or done something (BroadCastEvent), because of an event in the fairy-tale world (PerformedEvent, TriggerEvent

or WorldEvent), or because a certain amount of time has elapsed (TimeOutEvent). Such events arrive asynchronously; hence they give rise to a more flexible dialogue model. For instance, in the example above, more input from the user may arrive when Cloddy Hans is walking over to the fairy-tale machine. Using the event-based model outlined above, that is no problem; a new line of dialogue can be opened and the user's new utterance can be answered. Eventually the PerformedEvent in (5) above will arrive, and Cloddy Hans can then be made to switch back to the original line of dialogue.

7 Discussion and related work

The NICE fairy-tale game addresses the problem of managing conversational speech with animated characters that reside in a 3D-world. Few such systems have been built; the one is which most resembles the fairy-tale game is the Mission Rehearsal Exercise (MRE) system from the USC Institute of Creative Technologies (Swartout et al. 2004). The MRE system have more complex interaction between animated characters than the current version of the fairy-tale game, and uses a more sophisticated model for emotion (Traum et al. 2004).

Another problems addressed by the fairy-tale game is handling asynchronous, multimodal input. Here "asynchronous" means that a strict turn-taking scheme, where speakers proceed in alternation, need not be upheld. In particular this means that the user can make several dialogue contributions in sequence, without needing to wait for the system's reply. It also means that not only user utterances can trigger reactions from the system, but a fairy-tale character can also be triggered to speak as a reaction to events in the environment (e.g. that some other character says or does something). Existing asynchronous dialogue systems mostly work in robot domains (see e.g. Rayner et al. 2000, Lemon et al. 2001), where stimuli from the sensors of the robot trigger utterances from the spoken dialogue interface.

Over the recent years interactive story-telling research systems have been developed that in some cases allow linguistic input. It has been argued that interactive storytelling will change computer entertainment by introducing better narrative content and allowing users to interfere with the progression of the storyline (Cavazza et al. 2002). Most interactive games developed so far allow users to intervene in the storytelling by acting on physical objects on the screen using direct maniputation (Young 2001, Cavazza et al. 2002). Moreover, some systems allow users to interact with characters by means of written text input (Mateas and Stern 2002). In addition, Cavazza et al. (2002) explored using a speech interface that handled isolated utterances from the user.

Acknowledgements

This work was carried out within the EU-funded project NICE (IST-2001-3529, http://www.niceproject.com).

References

Boye, J, Hockey, B.A. and Rayner M. (2000) Asynchronous dialogue management. *Proc. Götalog, 4th workshop on the semantics and pragmatics fo dialogue,* Göteborg.

Boye, J, Gustafson, J. & Wirén, M. (2006) Robust spoken language understanding in a computer game. *J. of Speech Communication*, forthcoming special issue on spoken language understanding.

Cavazza, M., Charles, F. and Mead S. J. (2002). Character-based interactive storytelling. *IEEE Intelligent Systems*, Special issue on AI in Interactive Entertainment,, pp. 17-24.

Charles, F. and Cavazza, M. (2004) Exploring the scalability of character-based storytelling. *Proc. ACM Joint conference on autonomous agents and multi-agent systems*, New York, USA.

Fikes, R. E. and Nilsson, N. (1971) STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2 (3-4), pp. 189-208.

Gustafson, J., Boye, J., Fredriksson, M., Johannesson, L., and Königsmann, J., "Providing computer game characters with conversational abilities," in *Proc.of Intelligent Virtual Agent (IVA05)*. Greece, forthcoming.

Hindley, R. and Seldin, J. (1986) *Introduction to combinators and λ-calculus.* Cambridge University Press.

Lemon, O., Bracy, A., Gruenstein, A. and Peters, S. (2001) Information states in a multi-modal dialogue system for human-robot conversation. *Proc. Bi-Dialog, 5th workshop on the semantics and pragmatics of dialogue*, pp 57 – 67.

Mateas, M. and A. Stern (2002). Architecture, authorial idioms and early observations of the interactive drama Facade. Technical report CM-CS-02-198.

Rayner M., Hockey B.A. and James, F. (2000) A compact architecture for dialogue management based on scripts and meta-outputs. *Proc. Applied Natural Language Processing (ANLP).*

Swartout, W., Gratch, J., Hill, R., Hovy, E., Marsella, S., Rickel, J. and Traum D. (2004). Toward virtual humans. *AAAI Fall symposium on Achieving human-level intelligence through integrated systems and research*.

Traum D., Marsella, S. and Gratch J. (2004) Emotion and dialogue in the MRE virtual humans. *Proc. Workshop on affective systems*, Kloster Irsee.

Varanese A. and LaMothe, A. (2003) *Game scripting mastery.* Premier Press.

Young, R. M. (2001). An overview of the Mimesis architecture: Integrating intelligent narrative control into an existing gaming environment. *Working notes of the AAAI spring symposium on Artificial intelligence and interactive entertainment.*