

DYNAMIC RELATIONSHIP MANAGEMENT FOR PERSONALITY RICH CHARACTER PRESENTATIONS IN INTERACTIVE GAMES

Manish Mehta*, Kinshuk Mishra* and Andrea Corradini**

***Georgia Institute of Technology, College of Computing, Atlanta, GA 30332, USA
{mehtama1,kinshuk}@cc.gatech.edu**

****University of Southern Denmark, Institute of Business Communication and Information
Science, 6000 Kolding, Denmark
andrea@sitkom.sdu.dk**

Introduction

Generation of realistic behaviors is paramount to the believability of characters within an interactive computer game

Endow characters populating the virtual graphical world with capability of generating human-like behaviors and shape them according to their own personality

Non-player characters (NPC) capable of responding appropriately to game activities

Problem Definition

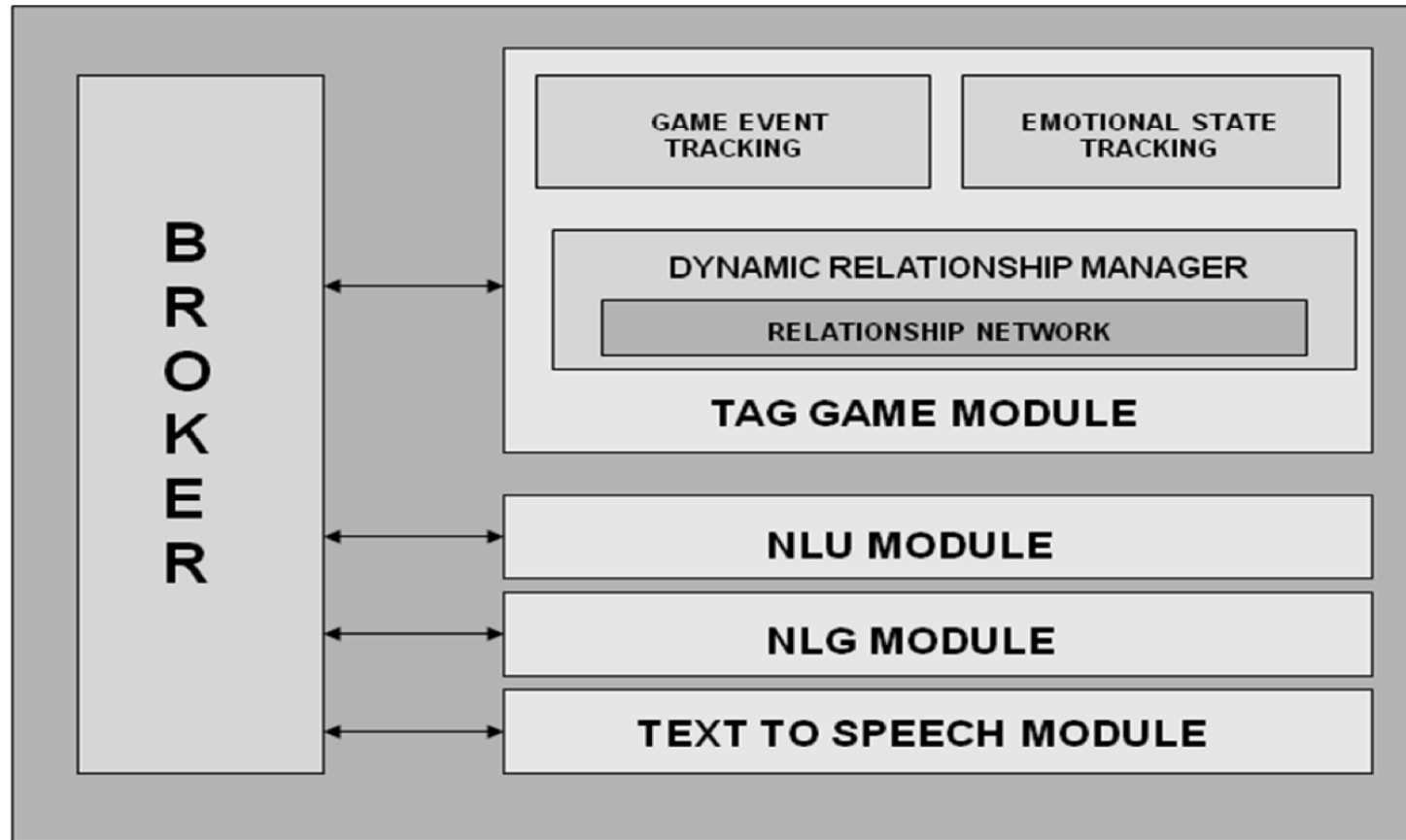
- Generate dialog for characters in a real time domain using various factors
 - Character Personality (touchy, reserved etc)
 - Their emotion state (happy, sad, angry)
 - Various important game events during the game
 - Interpersonal relationship between the speaker and hearer

Problem Definition

- Involved in a game of Tag where they chase the player or get chased
- Each character has its own personality and affects the way he approaches the game



Communication architecture



- Different software agents communicate via the broker
- Broker module freely available

(Lewin, E.: 1998, <http://www.speech.kth.se/broker>)

Dynamic Relationship Management

- Dynamically modify relationship between characters, based on
 - Current game event
 - Existing relationship
 - Verbal exchanges between the characters
- To understand verbal exchanges, we use NLP approach based on previous work in [Mehta & Corradini 2006]
- The parsed NLU input could be used by NLG to select templates and fill in their corresponding slots

Tracking Game Events

- Use A Behavior Language (ABL) [Mateas & Stern, 2003] to track different events that characters should respond to
- Different events being tracked
 - ◆ Character gets bumped by someone
 - ◆ Tags someone
 - ◆ Tagged by someone
 - ◆ Character hears something

Checks to make sure that the character you are bumping is not IT

```
with (success_test {      (ItWME itPlayerName :: itName )
                          (!itName.equalsIgnoreCase(MY_REALNAME))
                          (BumpWME bumpPlayerName :: bumpName)
                          (!bumpName.equals(itName))    })wait;

  mental_act{
    nlg_object.ablgameevent("Bumpsomeone");
    if(thetrace!= null)
      nlg_object.putemotion(thetrace.getEmEmotions(),
                            thetrace.getEmValues());
  }
```

Call the NLG system to generate sentence

Keeping Track of Different Emotions

- Emotion Tracking based on previous work (Zhang et. al 2007)
- Emotion model based on a simple OCC model of emotions (Reilly, 1996)
- Currently Four emotions (Angry, Sad, Happy, Stress)
- Each Behavior is annotated with an emotional increment
- As behaviors finish, the emotional state gets incremented with emotion value
- Current emotional state combination of values for the different emotions

Understanding verbal exchanges

- Detect the phrase type (whether critical)
- Rule based approach to understanding
-

Convert all the auxiliaries at the beginning to question of type yes/no

```
<aux:all> <character> :- <question:yes/no>  
<character>  
apply_at_position :- [beginning]  
Number_of_conditions :- 0
```

There is no condition that effects the application
Conditions can be based on presence of other categories

Character input : I dont really like you

NLU output : <dialogact:user_opinion<dialogacttype:negative>
<property:like>

Phrase Type : <phrase_type:user_criticism>

Represent user input in terms of dialog act, concepts, properties

Natural Language Generation

- Uses a set of author defined templates that can be reused across different characters and emotional states
- Templates as a set of sentence structures that require words and phrases from a lexicon to be filled in
- Words and phrases that are used are dependent upon the emotions, and personality of the speaker
- Generate dialog for characters in a real time domain using various factors
 - ◻ Character Personality (touchy, reserved, etc)
 - ◻ Their emotion state (happy, sad, angry)
 - ◻ Various important game events during the game
 - ◻ Interpersonal relationship between the speaker and hearer

Example Illustration

1. initial relationship between Jack and the player is a neutral
2. During the game, when Jack is tagged, the player utters “now you bummer should run after me”
3. Jack’s current emotional state changes to sad

Example Illustration

- 4. NLU detects it as an “insulting” conversational move on the part of the player
- 5. DRM modifies the relationship between Jack and the player to unfriendly using 1, 2, 3, 4
- 5. Using 4 changes of 1) the emotional state of Jack becoming stressed and sad, 2) the relationship between Jack and the player changing to unfriendly, 3) the game event of Jack getting tagged and 4) his personality of an introvert who gets stressed a lot

NLG produces “I don’t like playing this game ”

System Evaluation

- . Group of seven participants (six male and one female) to consider whether the NLG system was producing verbal output in accordance with the emotional state of the character
- . Categorization of the produced verbal output on a boolean scale (i.e. correct or incorrect emotion corresponding to the verbal output), the results indicated a 71% correct categorization rate
- . Plan to carry out further experiments on the DRM module in the near future

That was it, thank you!

Relationship Modification & Text to Speech

- Three kinds of relationships that have been modeled between the characters: friendly, neutral and unfriendly
- DRM takes 3 factors into account while modifying the relationships: game events, conversational move and existing relationship that the actor and the subject have between them

• MS Speech Synthesis to have different voices for Jack and Jill

1
Some character tags the other characters

```
<category: self_self_neutral_all_neutral>  
<gameevent:tagsomeone> :- <postrelationwithactor:friendly>
```

```
number_of_conditions : 1  
condition : <conversational_move:user_praise>
```

3
Relationship changes to friendly

2
And makes a conversational move praise