1 The Situated Language Interpreter

1.1 Introduction

This chapter introduces the SLI system, which implements the interpretive framework developed in this thesis. In this framework the interpretive modules of an NLVR system models the linguistic context <u>and</u> the user's perceptual context. This approach differs from previous NL systems that attempted to interpret language grounded in a visual domain, (i.e., SHRDLU (Winograd 1973), CITYTOUR (Andre, Herzog et al. 1986; Andre, Bosch et al. 1987), CSR-3-D (Gapp 1994), SPRINT (Yamada 1993), WIP (Olivier, Maeda et al. 1994; Olivier and Tsuji 1994), Situated Artificial Communicator (Socher and Naeve 1996; Socher, Sagerer et al. 1996; Vorwerg, Socher et al. 1997; Fuhr, Socher et al. 1998), Virtual Director (Mukerjee, Gupta et al. 2000), and CommandTalk (Dowding, Bratt et al. 1999; Stent, Dowding et al. 1999; Goldwater, Bratt et al. 2000)) as these systems – rather than modelling the user's knowledge of the simulation – gave their interpretive module complete access to all the objects in the simulated world. This is phenomenologically unrealistic and impractical in large environments with many objects where such an approach can result in a multiplication of the possible referents.

This chapter gives an overview of the SLI system's architecture and provides an example user-system interaction scenario that illustrates some of the system's functionality. The scope is restricted to a general introduction and a high-level description of how the systems functions.

1.2 The SLI System Architecture

There are six components in the SLI system architecture: the parser, the rendering engine, the world model, the visual saliency module, the context model, and the interpretive module.

1.2.1 The SLI Parser

The SLI parser is simple and was developed by the author for the system. It is written in Lingo, an object-oriented programming language similar to C++ or Java. The only aspects of the SLI parser specific to this project are the categorising of an input into different types of commands and explicit checking for linguistic cues that impact on the frame of reference selection; in particular the use of a genitive nouns phrase to describe the landmark of a locative. Essentially there are two types of commands that the system handles:

1. Avatar Commands: user commands to their avatar; For example, *walk forward* or *turn right*.
2. Artefact Commands: user commands which change the state of the world by altering attributes of any world artefact apart from the user's avatar. For example, *make the house taller* or *move the tree to the right of the blue house forward*.

Categorising the input is based on the verb used and the syntactic structure of the command. For example, if the verb is either *run* or *walk,* it is categorised as an avatar command. However, if the verb is *move* it could be either; e.g., *move forward* (avatar command) or *move the house back* (artefact command). Here, *move* + *adverb* is categorised as an avatar command and *move* + *np* + ... is categorised as an artefact command. Once the parser has processed the input string, categorised the type of command, and checked for frame of reference cues, it passes its results on to the interpretive module, described later.

1.2.2 The SLI World Model

The world model is a list of the objects to be drawn on the screen by the 3-D rendering engine. A similar form of data structure is present in all 3-D applications. Each element in this list describes the geometry, surface appearance, world position, and frames of reference associated with each world artefact. The 3-D geometry of each object is represented by a mesh, the world position by a vector, and the frames of reference orientations by a set of unit vectors. A mesh is an interconnected network of points that defines a 3-D model's surface. The only part of this model specific to this thesis is the inclusion of the frame of reference information. This, however, is not novel, since several other systems (e.g., (Olivier and Tsuji 1994)) use similar approaches.

1.2.3 The Rendering Engine

The rendering engine used by the SLI system is the Shockwave 3-D Viewer. A **rendering** engine takes a geometric model of a scene and camera viewpoint and draws the scene on the screen. There are several rendering engines available at present, all of which provide similar functionality. Two of the best known are DirectX and OpenGL. The motivation for using the Shockwave 3-D rendering engine in developing the SLI system was to deploy the system over the Internet.

The novelty of the system resides in the visual saliency module, the context module, the interpretive module, and how these components interact. A high-level overview of their role in the SLI framework is given here.

The purpose of the SLI visual saliency module is to capture the flow of visual information from the simulated environment to the user. This module extends the false colouring technique (Renault, Magnenat-Thalmann et al. 1990; Noser, Renault et al. 1995; Kuffner and Latombe 1999; Peters and O'Sullivan 2002). The output of this module feeds into the context module.

The role of the SLI context model is to provide a context for the interpretation of referring expressions, integrating and structuring the perceptual information captured by the SLI visual saliency algorithm and the linguistic discourse history. Because the SLI context module extends the scope of context beyond the linguistic discourse, the SLI system can resolve references to visually accessible objects which have not been previously mentioned in the discourse.

The SLI interpretive module takes user commands from the parser and uses information from the context model to interpret these commands. This process results in the updating of the world model and the context model. It is important to note that, unlike previous systems that have interpreted language which is grounded in a visual domain, SHRDLU (Winograd 1973), CITYTOUR (Andre, Herzog et al. 1986; Andre, Bosch et al. 1987), CSR-3-D (Gapp 1994), SPRINT (Yamada 1993), WIP (Olivier, Maeda et al. 1994; Olivier and Tsuji 1994), Situated Artificial Communicator (Socher and Naeve 1996; Socher, Sagerer et al. 1996; Vorwerg, Socher et al. 1997; Fuhr, Socher et al. 1998), Virtual Director (Mukerjee, Gupta et al. 2000), and CommandTalk (Dowding, Bratt et al. 1999; Stent, Dowding et al. 1999; Goldwater, Bratt et al. 2000), the SLI interpretation module does not directly access the world model. Rather, the interpretive module's knowledge of the world is mediated through the SLI context model and, consequently, is founded on the information captured by the SLI visual saliency module.

Figure 1-1 is a schematic overview of the system. Each of the boxes in this figure represents and names a component in the system. The arrows between boxes represent the flow of information between the components: arrow (1) represents the user inputting commands to the system; arrow (2) represents the parsed input being passed to the interpretive module; arrow (3) represents the flow of contextual information from the context model to the interpretive module (this information provides a context for the interpretation of user input received from the parser); arrow (4) represents the updating of the context model after the interpretation module has processed a user input; arrow (5) represents the updating of the world module after the interpretive module has processed a user input; arrow (6) represents the rendering engine's use of the world model during the rendering process, represented by arrow (7); arrow (8) represents the interrogation of the rendering engine by the visual saliency algorithm after each frame has been rendered; arrow (9) represents the visual saliency algorithm's use of the false colouring information stored in the world model and its interrogation of the world model; and arrow (10) represents the updating of the context model by the visual saliency module after a frame has been rendered.

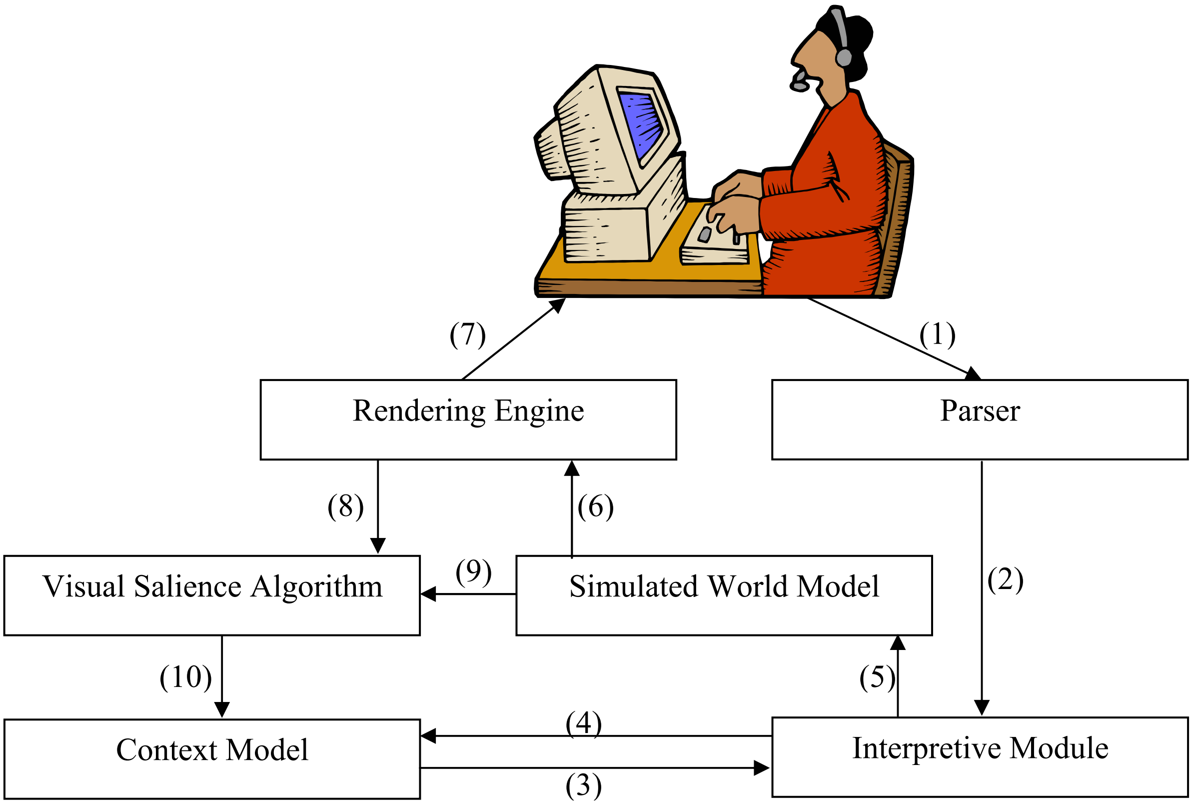

Figure 1-1: Schematic of the SLI system architecture and the data flow between the system components.

1.3 Example User-System Interaction Dialogue

This section contains a sample user-system interaction dialogue. The setting of the dialogue is a simple scene containing houses and trees. The numbered, italic and lower case text below was input by a user. The system's response to these inputs was displayed on the screen either as changes to the visual environment or text output in the SLI message window. All figures here are SLI screen shots which update the visual scene in response to an input. The dialogue was carried out in real-time.

Figure 1-2 shows the start-up screen. The major components of the SLI interface are named. Note the house and trees in the 3-D simulation window.

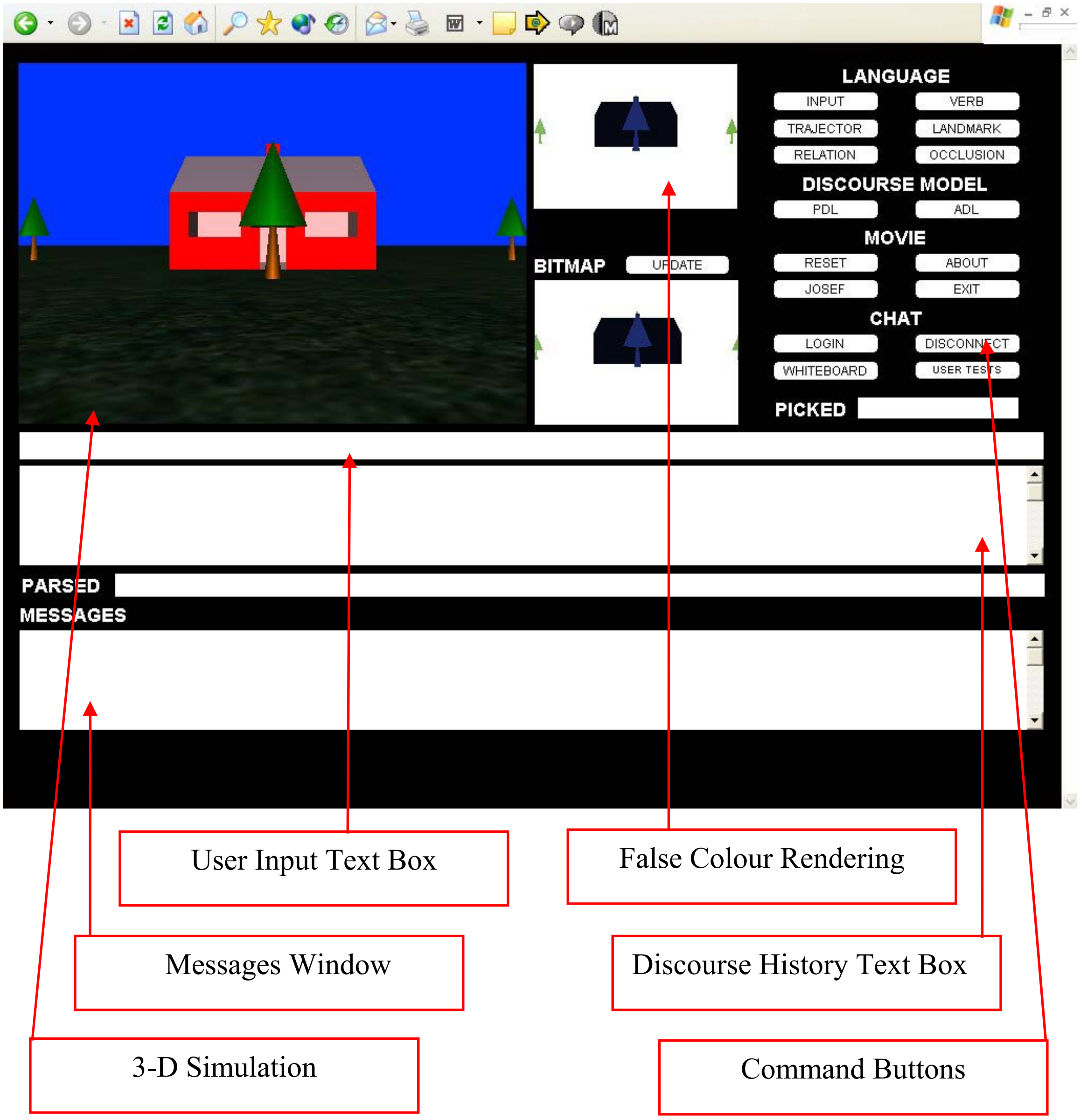

Figure 1-2: The SLI Interface.

The following is a sample input:

Input 1-1: *make the tree to the right of the house red*

In order to interpret Input 1-1, *make the tree to the right of the house red*, the system first resolves the landmark nominal *the house*. It should be noted, that although there is more than one object in the world that fulfils the description *the house* (the complete 3-D simulation contains a number of houses currently not in the view, the SLI system is able to resolve this reference because its interpretive module does not directly access the world model. Rather, its knowledge of the world, which is stored in the context model, is mediated by the visual saliency algorithm. Consequently, despite the fact that there are multiple houses in the world, the system can use the knowledge that the user is currently only aware of one of these houses to resolve this definite description. Importantly, systems that allow their interpretive module complete access to the world model would not be able to resolve a definite description if there was more than one object in the simulated world that fulfilled the description. Furthermore, discourse models that are only updated in response to linguistic input would not have recorded the perceptual event of the user seeing the house. Consequently, these discourse models would be of limited use in resolving this reference.

Having resolved the landmark reference, the system constructs a model of the projective preposition's semantics using a potential field. The system uses this potential field model to adjudicate between the elements of the set of candidate trajectors. The set of candidate trajectors is the set of objects in the context model that fulfils the description of the subject noun phrase in the input. Again, because the SLI's interpretive module does not have full knowledge of the world, the system can restrict the set of candidate objects to those that fulfil the linguistic description of the input and, crucially, that the user is currently aware of. In this instance, there are three trees in the context model. The system uses its model of the preposition's semantics to select between these candidates. Once this selection is complete, the system updates the world model to reflect its interpretation of the user's input.

Figure 1-3 illustrates the state of the simulated world after this input has been processed.

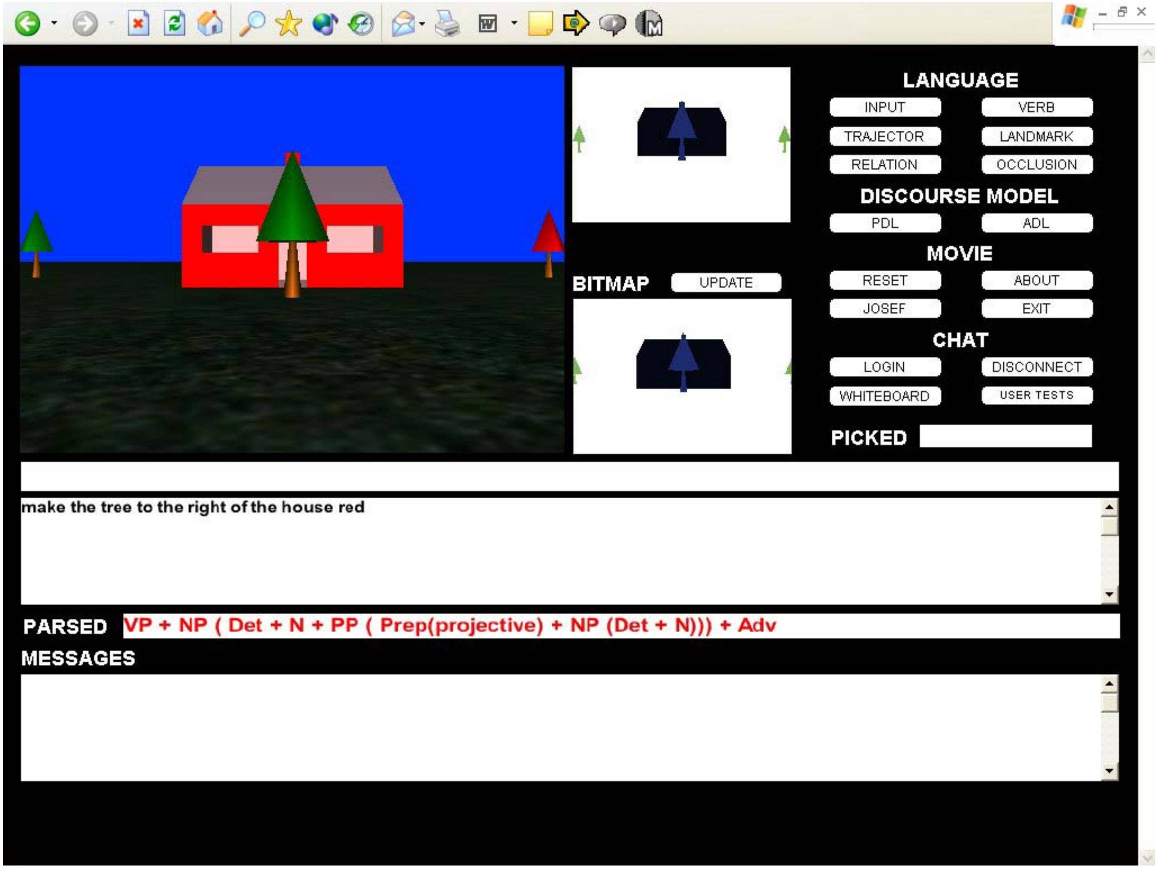

Figure 1-3: The state of the system after the input *make the tree to the right of the house red.*

Input 1-2: *make the blue house taller*

Although there are blue houses in the world, the user has not seen any of them during this interaction dialogue. As a result, there are no references to a blue house in the context model. Consequently, the system responds by outputting a message to the user in the message window:

<u>No object of that type in memory!</u>

Figure 1-4 illustrates the state of the system after this message has been output.

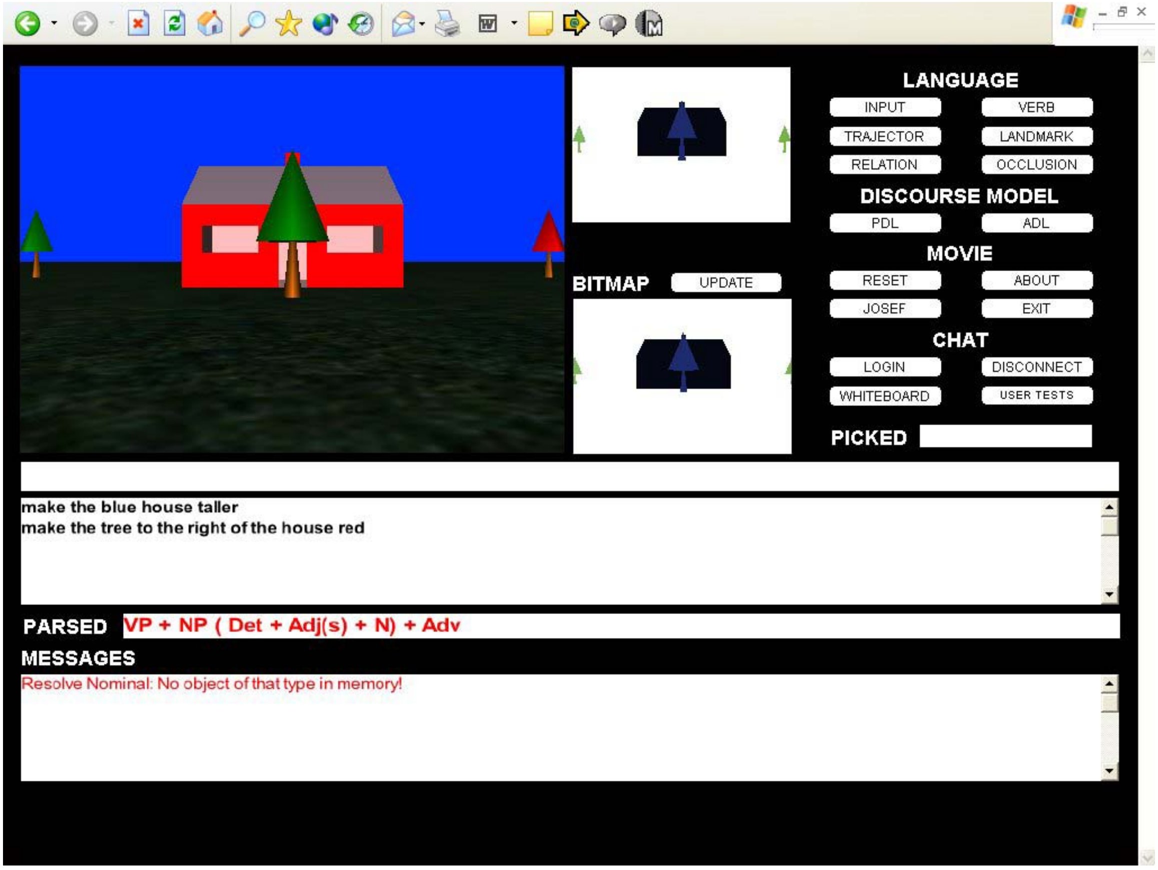

Figure 1-4: The SLI system after it has output a message to the user.

Input 1-3: *turn left*

Input 1-3 is an avatar command; i.e., it modifies the user's viewpoint. The system recognises these types of commands by examining the syntactic structure of the input (Verb + Adverb). Once the system determines that an input is an avatar command, it adjusts the user's viewpoint accordingly.

Figure 1-5 illustrates the user's view of the simulation after the input *turn left* has been processed.

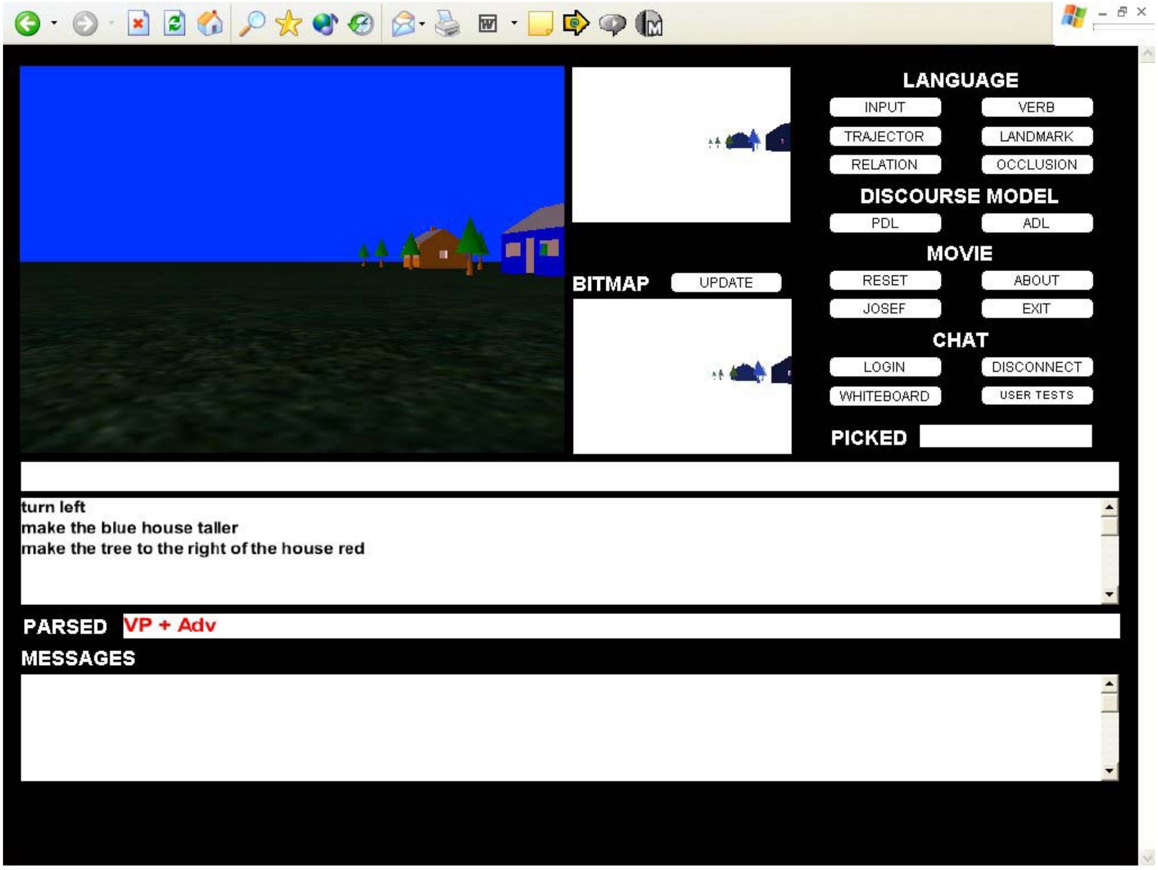

Figure 1-5: The user's view of the simulation after the input *turn left* has been processed.

Input 1-4: *make the blue house taller*

Input 1-4 is a repeat of Input 1-2. Earlier when this command was input, the system responded with: No object of that type in memory! However, now there is a blue house in the user's view volume. Recall that the system's context model is updated by the visual saliency module as well as the interpretive module (see Section 1.2), and as such, there is a blue house in the context model at this point. Therefore, the system can resolve the definite description *the blue house* in the input.

Figure 1-6 illustrates the visual context after this input has been processed. Comparing Figure 1-5 and Figure 1-6 highlights the change in the height of the blue building as a result of this input. More importantly, when this input was being processed there were three different houses in the context model: the brown and blue houses in the current scene and the red house seen previously. However, the system was able to resolve the nominal *the blue house* by using the supplied adjectival description as a selectional restriction on the referent of the expression. The processing of this input also illustrates the system's ability to manipulate the properties of world objects in response to user input. The magnitude of the scaling of an object's dimensions is calculated by the system using simple heuristics. Note that changes to object properties in the main 3-D environment are mirrored in the false colouring rendering, evident when the false colour renderings in Figure 1-5 and Figure 1-6 are compared.

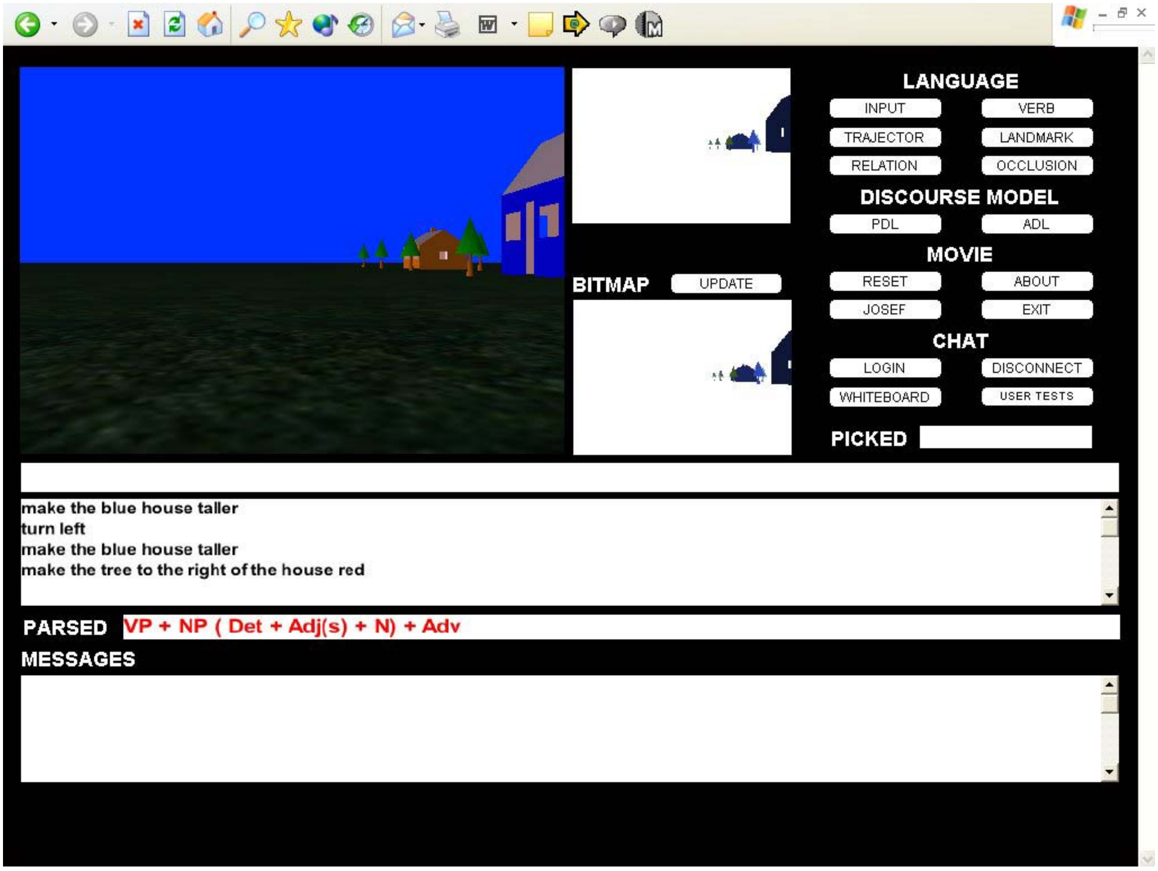

Figure 1-6: The SLI simulation after the input *make the blue house taller* has been processed.

Input 1-5: *make the other one yellow*

Interpreting Input 1-5 involves figuring out what it meant by *other* and *one*. It is important to note that, although Input 1-5 uses a one-anaphoric expression, modified by *other* which is also anaphoric, the most likely referent of this expression (i.e., the brown house that is currently in the view volume) has not been previously mentioned in discourse. Again, however, because the SLI's discourse model is updated by the visual saliency algorithm, the object in question is in the context model.

The system interprets one-anaphora as intending on an object of the same type as the currently profiled object: an object is profiled when it is selected as the referent for an expression. The anaphoric modifier *other* is understood by the system to specify that the referent of this expression should not be currently profiled. Following this, the SLI system interprets *the other one* to refer to the most salient object in the current context that is of the same type as the currently profiled object, but is not itself currently profiled.

The currently profiled object is the blue house that was selected as the referent for the previous input, Input 1-4. There are two other (i.e., not currently profiled) houses in the context model: the brown house in the view volume and the red house the user saw earlier. Neither of these are currently profiled, hence they both fulfil the linguistic descriptions of *the other one* as understood by the SLI system. The system uses the current visual salience associated with each of these objects to select one as the referent. In this example, the brown house is in the view volume while the red house is not. Consequently, the brown house has a higher visual salience than the red house and is selected as the referent.

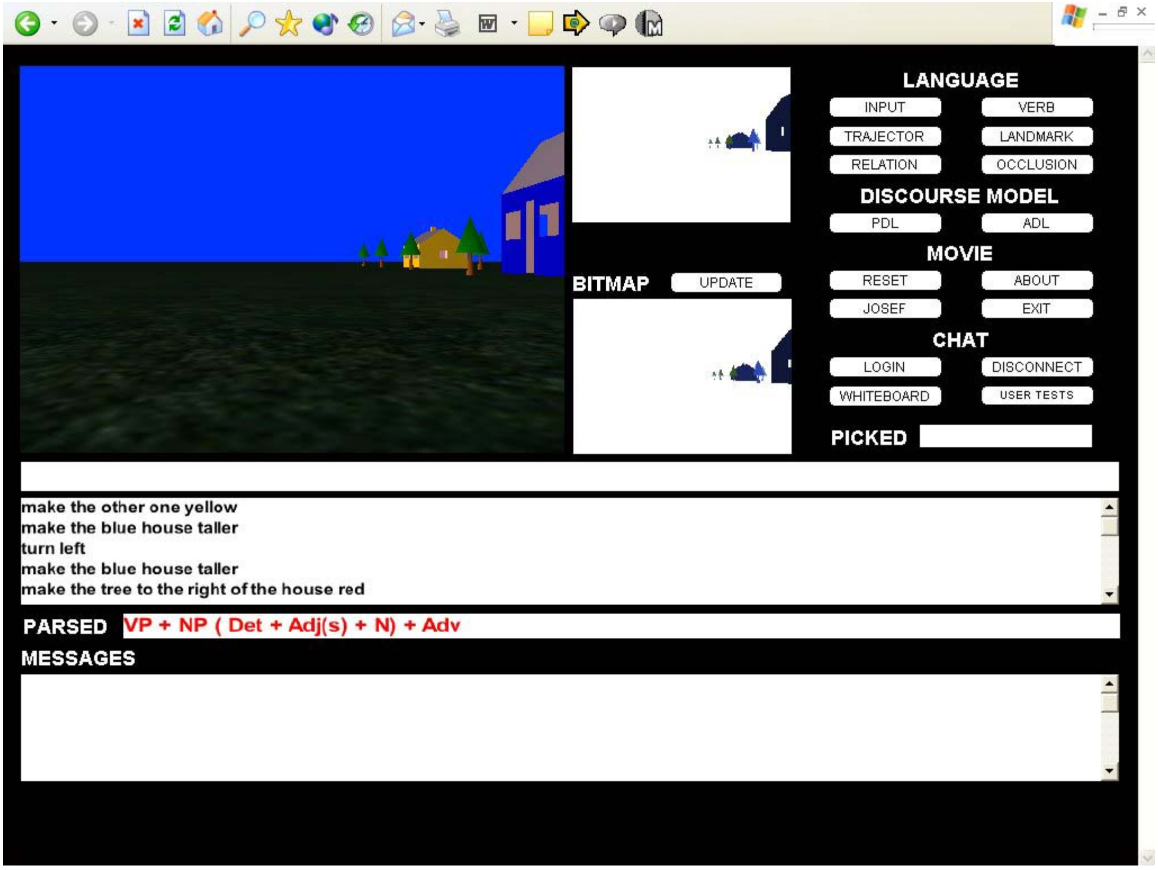

Figure 1-7: The state of the SLI simulation after processing the input: *make the other one yellow.*

Input 1-6: *look at the red house*

Although there is no red house in the user's current view, the system remembers that the user saw a red house earlier in the session. Crucially, this is recorded in the system's context model, even though no reference to a <u>red</u> house has been made in the discourse. This is because the context model is updated by the visual saliency algorithm after each frame has been rendered. As a result, the system can resolve the reference *the red house* to the red house previously seen by the user and rotates the user's avatar so that the user's view is centred on this house.

Figure 1-8 illustrates the user's view of the world after the input *look at the red house* has been processed.

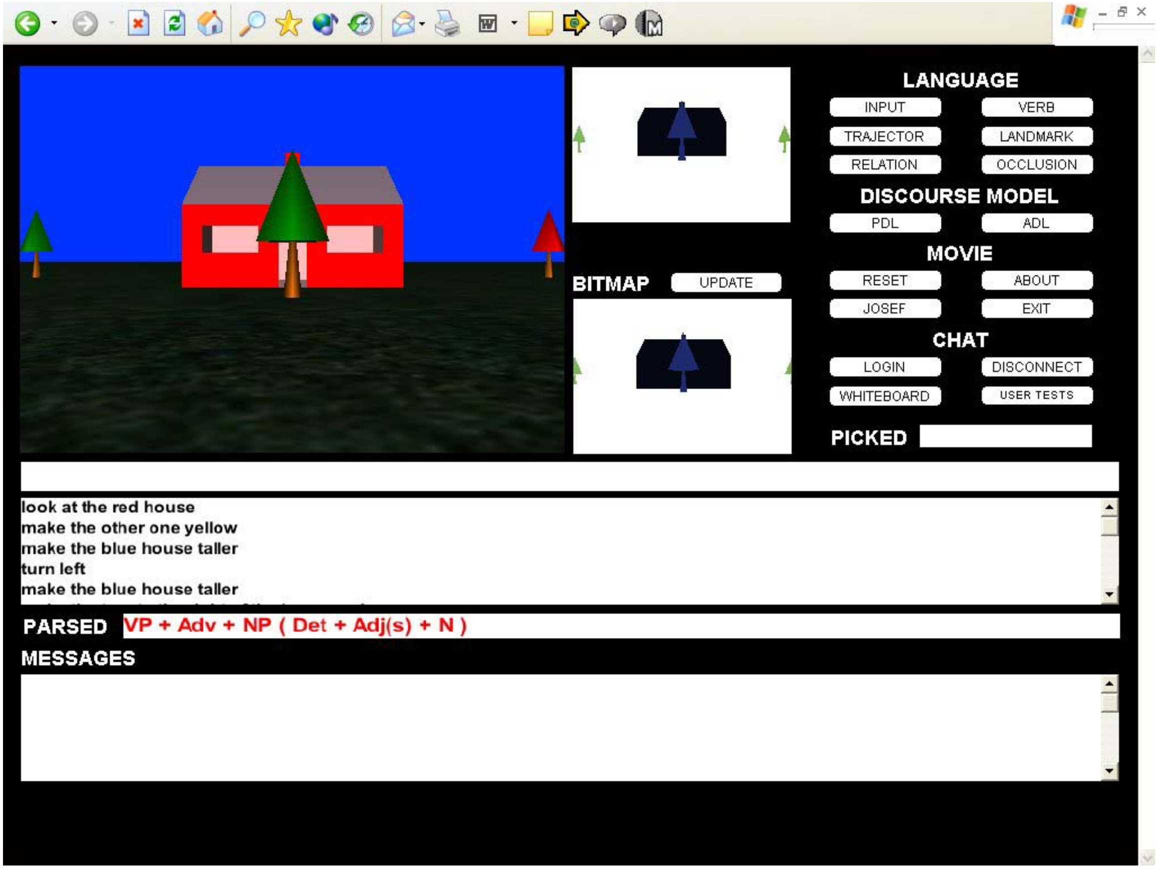

Figure 1-8: The user's view of the simulation after the input *look at the red house* has been interpreted.

Input 1-7: *make it taller*

This input uses the pronoun *it* to intend on its referent. In contrast to Input 1-5, whose referent had not previously been mentioned in the discourse, this input uses an anaphoric expression that links back to a previously mentioned object. The ability of the SLI system to resolve anaphoric expressions is based on the updating of the context model by the interpretive module after each user input has been processed.

Figure 1-9 shows the state of the world after the input *make it taller*, has been interpreted.

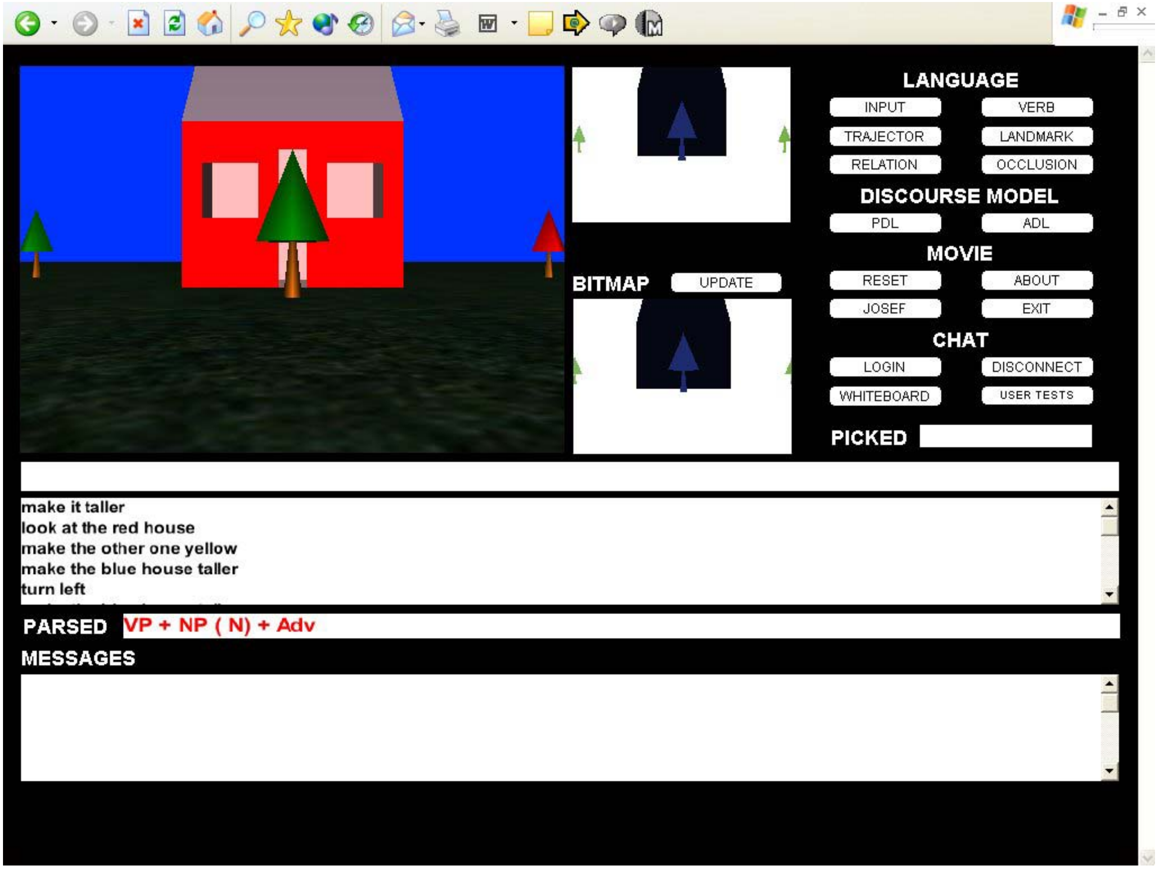

Figure 1-9: The SLI simulation after the input *make it taller* has been processed.

Input 1-8: *move backwards*

Input 1-9: *stop*

Input 1-8 and Input 1-9 are both avatar commands. These commands allow the user to navigate through the simulated world. Input 1-8 causes the user's viewpoint to move backwards from its current position. Input 1-9 stops the movement of the user's viewpoint.

Figure 1-10 illustrates the user's view of the simulation after these commands have been processed.

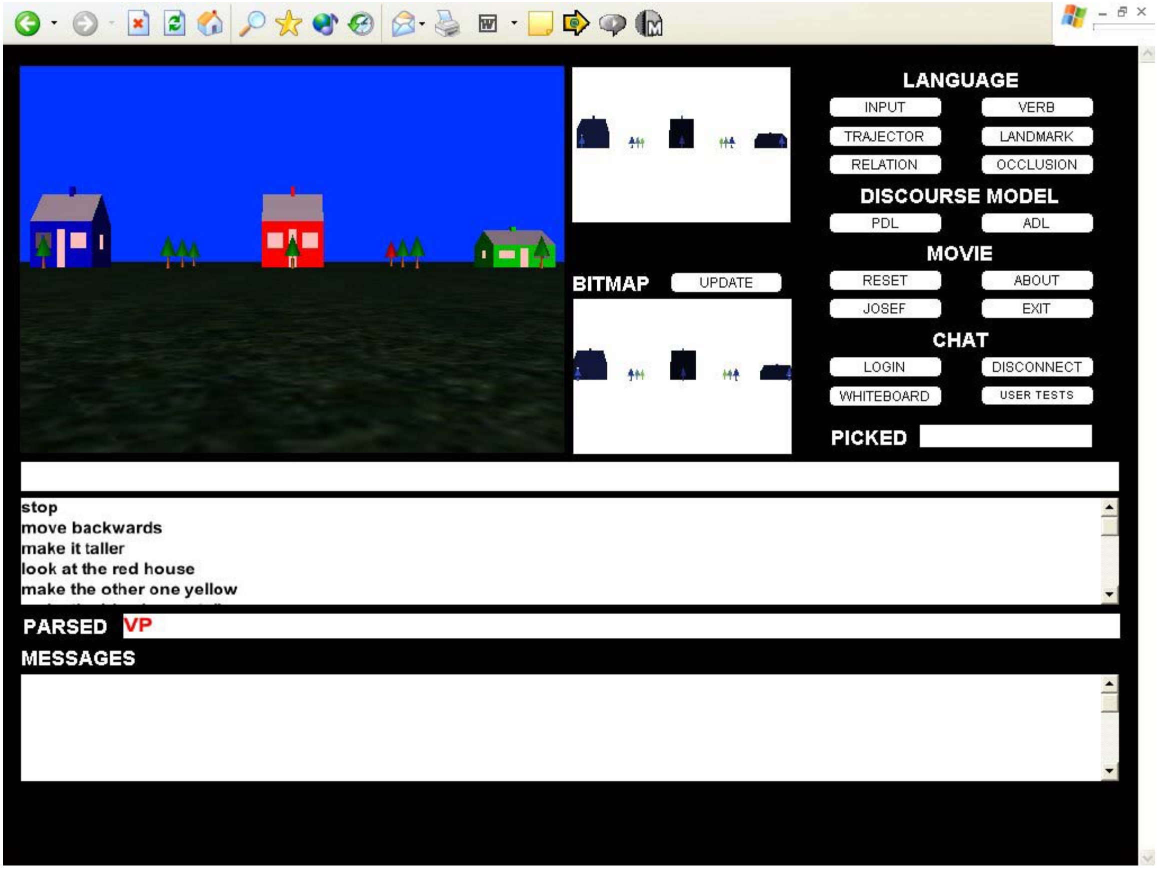

Figure 1-10: The user's view of the simulation after the avatar commands *move backwards* and *stop* have been processed.

It is worth noting that Figure 1-10 also illustrates that there are trees in the world which are more *to the right of the red house* than the tree which was selected as the referent to Input 1-1 *make the tree to the right of the house red*. However, at the time Input 1-1 was being processed, the user was not aware that these other trees existed. Because the SLI system used its model of the user's world knowledge (created by the visual saliency module and stored in the context model), rather than the world model itself, as the context for interpreting Input 1-1, it was able to exclude the trees that the user was not aware of from the interpretation process, even though they were more to the right of the house than the tree selected as the referent for Input 1-1.

Input 1-10: *make the red tree and the short house taller*

Input 1-10 illustrates the system's ability to handle coordinating expressions. It also illustrates the system's ability to resolve definite description using both qualitative and quantitative object properties. In resolving the expression *the red tree* the system uses a supplied qualitative adjectival description, *red*, as a restriction on an object being considered as a candidate referent. In this instance, there is only one red tree in the context model and it is extracted as the referent for this expression. In order to resolve *the short house*, the system must compare the dimensions of candidate referents to select the candidate that fulfils the supplied adjectival description to the greatest extent. It first creates a list of all the objects in the current context that are off the right type, *house*. At this point in the dialogue, there are four houses in the context model: the three houses currently in the user's view (the red, blue, and the green houses) and the yellow house (currently no in view) to the left of the blue house that was selected as the referent to Input 1-5. Once the system has created the list of candidate referents, it then searches the list for the shortest object and selects this object as the referent for the expression. In this instance, the green house and the yellow house have an equal score with respect to the shortness criteria. In order to adjudicate between these two candidates, the system uses the visual saliency scores associated with each object. Here, the green house is in the view volume and the yellow house is not; consequently, the green house has a higher visual saliency score and is selected as the referent for the nominal *the short house*. Again, it should be noted that the referent of this expression has not previously been mentioned in the discourse.

Figure 1-11 illustrates the state of the simulation after this coordinating expression has been resolved.

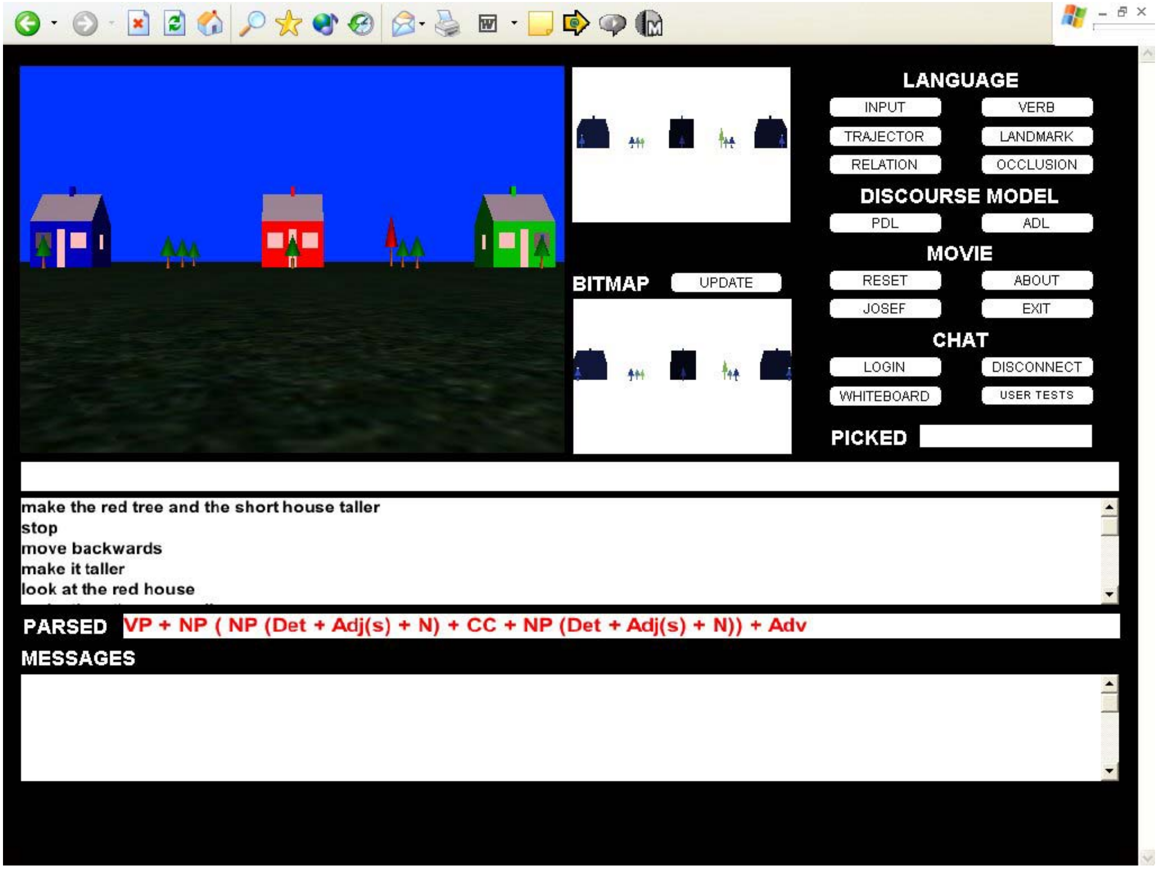

Figure 1-11: The state of the SLI simulation after the input *make the red tree and the short house taller* has been processed.

Input 1-11: *make a house yellow*

Input 1-11 illustrates the system's ability to resolve indefinite descriptions. In the context of the SLI system, an indefinite expression, *a N,* may be used to arbitrarily refer to one of the elements of type *N* in the spatio-temporal context, or to refer to the generic type *N* in commands that create new objects in the world. The system decides which type of *N* an indefinite expression refers to based on syntactic cues. If the indefinite expression is followed by an adjective functioning as an adverb, as in Input 1-11, the system treats the input as referring to one of the elements of type *N* in the spatio-temporal context. Otherwise, the system assumes that the indefinite expression refers to the generic type *N*. Consequently, Input 1-11, is understood to refer to an object in the spatio-temporal context, and is interpreted as the user wanting the system to arbitrarily select object of type *N* that is already in the simulation and manipulate it. Accordingly, the system randomly selects a referent from the set of objects in the view that match the type restriction and any supplied adjectival restrictions in the input. In this instance, the system selected the red house and changed its colour to yellow.

Figure 1-12 illustrates the system after this input has been processed.

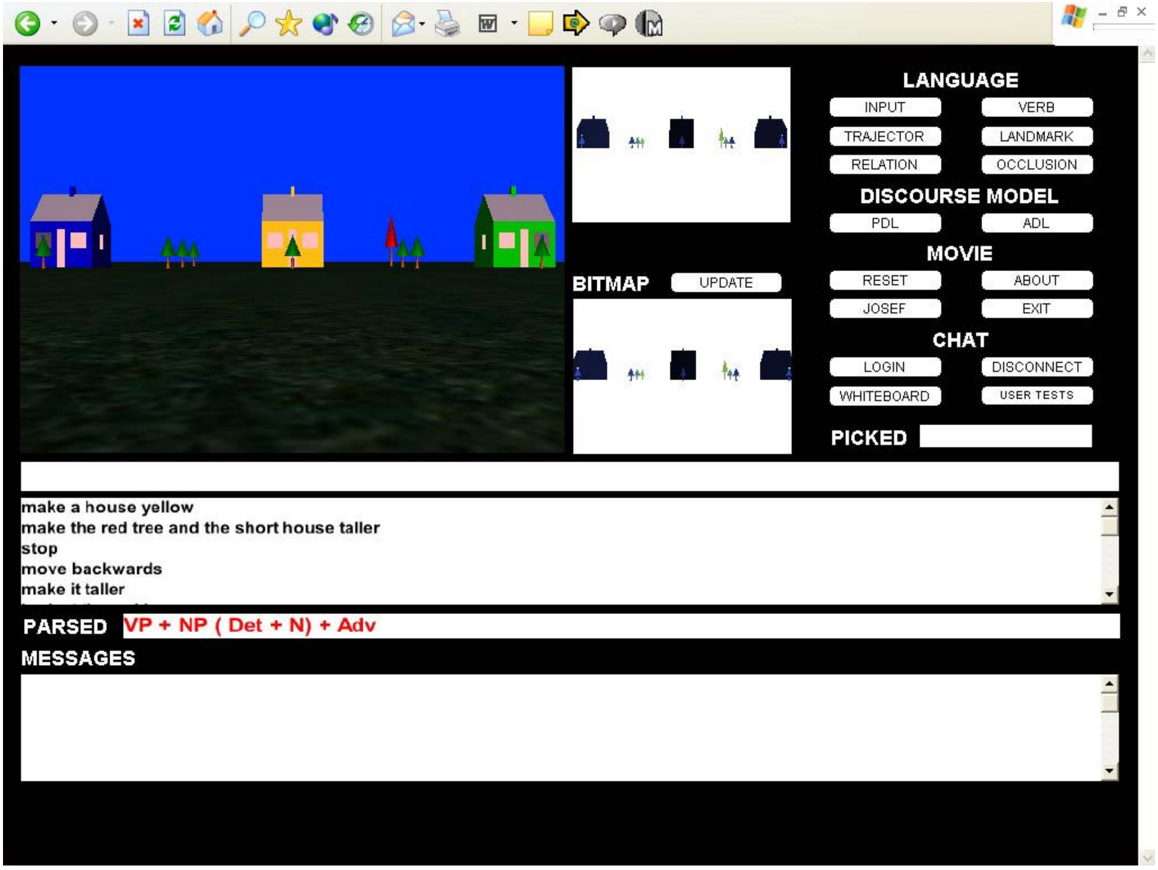

Figure 1-12: The state of the simulation after the input *make a house yellow* has been processed.

Input 1-12: *add a red house*

Input 1-12 illustrates how the system handles indefinites that refer to a generic object, rather than an arbitrarily selected object already in the context. It also illustrates the system's ability to extend the world by adding new objects. Note that it is possible to parameterise the creation of an object by using adjectives in the input. The system uses simple heuristic rules to calculate where the newly created object should be added.

Figure 1-13 illustrates the user's view of simulation after a new red house has been added to the world.

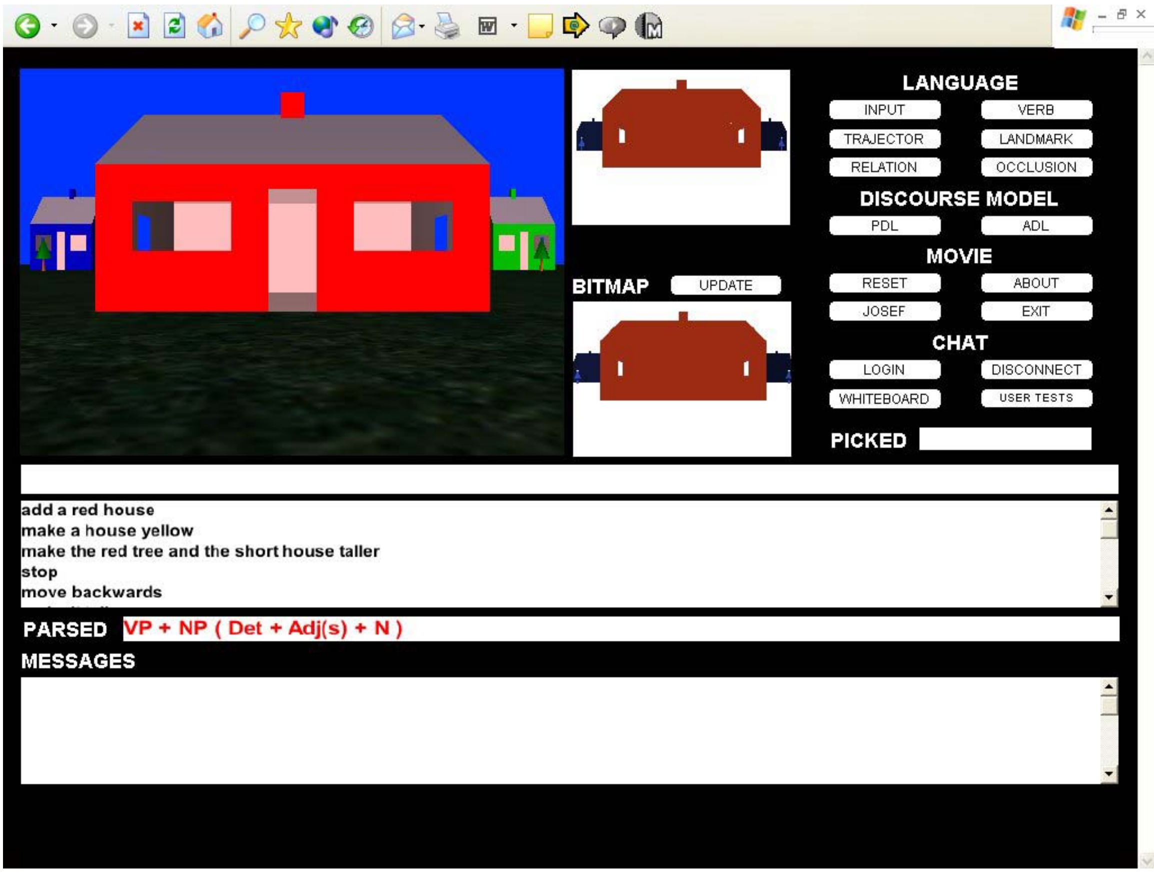

Figure 1-13: The state of the 3-D world after a new object has been added.

Input 1-13: *make this => brown*

The final input in this example dialogue, Input 1-13, illustrates how the SLI system resolves demonstratives that are accompanied by a pointing gesture. The arrow in the input (=>) symbolises the pointing gesture. In the SLI system, deictic gestures are simulated by mouse clicks on the intended object in the view volume.

Figure 1-14 illustrates the state of the SLI system after it has processed Input 1-13. The position of the user's mouse click is depicted in this image by the mouse cursor. Furthermore, the text box labelled PICKED lists the name of the object clicked on by the users.

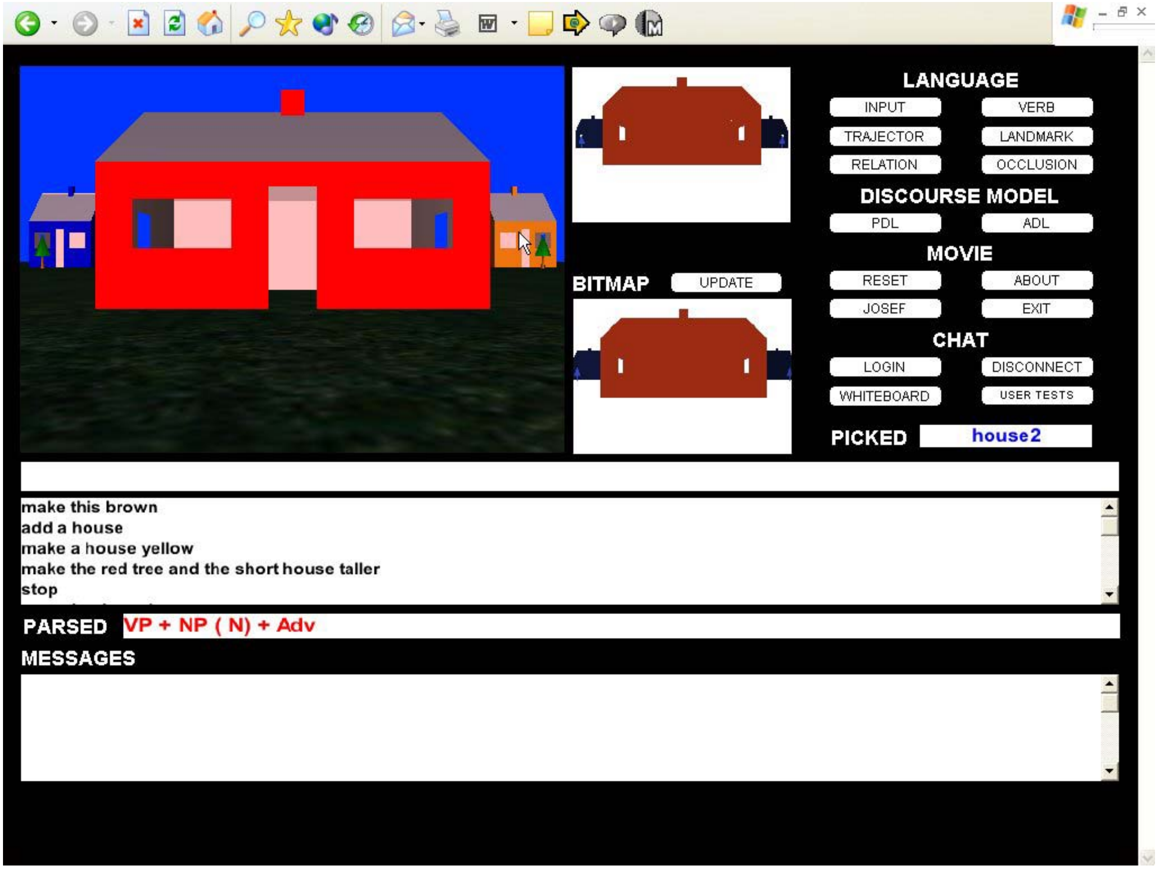

Figure 1-14: The state of the SLI system after the input *make this* => *brown* has been processed. The demonstrative *this* was accompanied by a pointing gesture (simulated by a mouse click). The position of the mouse is shown in the image. Also, the text box in the SLI interface labelled PICKED lists the name of the world object that was clicked on.

1.4 Chapter Summary

The SLI system was implemented as a test bed for the framework developed in this thesis. Illustrating the functionality of the system in this chapter before continuing with more details has hopefully provided the reader with an overview of framework. Moreover, describing the framework in the context of an implemented system also illustrates the tractability of the approach.

Bibliography

Andre, E., G. Bosch, et al. (1987). Coping with the Intrinsic and Deictic Uses of Spatial Prepositions. Originally appeared in: Proc. of 7th ECAI, volume 2, pp. 1-8, Brighton, UK, 1986 Available: http://citeseer.nj.nec.com/andre87coping.html. Accessed: February 15 2000.

Andre, E., G. Herzog, et al. (1986). Natural Language Access to Visual Data: Dealing with Space and Movement. Available: http://www.dfki.uni-sb.de/vitra/index/node7.html#SECTION00250000000000000000. Accessed: February 15 2000.

Dowding, J., E. O. Bratt, et al. (1999). Interpreting Language in the Context of CommandTalk. In "Communicative Agents: The Use of Natural Language in Embodied Systems", pp. 63-67, Association of Computing Machingery (ACM) Special Interest Group on Artificial Intelligence (SIGART), Seattle, WA. Available: http://www.ai.sri.com/~lesaf/commandtalk.html. Accessed: 24th September 2002.

Fuhr, T., G. Socher, et al. (1998). "A Three-Dimensional Spatial Model for the Interpretation of Image Data." in Representation and Processing of Spatial Expressions. P. Olivier and K.-P. Gapp, Lawrence Erlbaum Associates. **2001:** 103-118.

Gapp, K.-P. (1994). Basic meanings of spatial relations. Computation and evaluation in 3D space. Originally appeared in Twelfth National Conference on Artifical Intelligence (AAI-94),Seattle-Washington, Menlo Park, California: AAAI Press 1994 Available: http://citeseer.nj.nec.com/gapp94basic.html. Accessed: Mark 12 1998.

Goldwater, S. J., E. O. Bratt, et al. (2000). Building a Robust Dialogue System with Limited Data. Presented at the Workshop on Conversational Systems at the 1st Meeting of the North American Chapter of the Association of Computational Linguistics, Seattle, WA. Available: http://www.ai.sri.com/~lesaf/commandtalk.html. Accessed: 24 September 2002.

Kuffner, J. and J. C. Latombe (1999). Fast synthetic vision, memory, and learning models for virtual humans. Computer Animation IEEE.

Mukerjee, A., K. Gupta, et al. (2000). Conceptual Description of Visual Scenes From Linguistic Models. Originally appeared in Journal of Image and Vision Computing, Special issue on Conceptual Descriptions, v.18 (March 2000) Available: http://www.cse.iitk.ac.in/~amit/research/theat.html. Accessed: March 11 2002.

Noser, H., O. Renault, et al. (1995). "Navigation for Digital Actors Based on Synthetic Vision, Memory, and Learning." in Computer Graphics **19**(1): 7-9.

Olivier, P., T. Maeda, et al. (1994). Automatic Depiction of Spatial Descriptions. Twelfth National Conference on Artificial Integigence AAAI '94, Seattle, Washington.

Olivier, P. and J.-I. Tsuji (1994). "Quantitative Perceptual Representation of Prepositional Semantics." in Artificial Intelligence Review **8**(Special Issue on Integration of Natural Language and Vision Processing): 147-158.

Peters, C. and C. O'Sullivan (2002). A Memory Model for Autonomous Virtual Humans. EGIreland: Eurographics Irish Chapter Workshop, Dublin.

Renault, O., N. Magnenat-Thalmann, et al. (1990). A Vision-Based Approach to Behavioural Animation. Originally Appeared in Visualization and Computer Animation, 1: 18-21, 1990 Available: http://ligwww.epfl.ch/~thalmann/papers.dir/JCVA.vision.pdf. Accessed: 20th June 2002.

Socher, G. and U. Naeve (1996). A Knowledge-based System Integrating Speech and Image Understanding. Available: www.techfak.uni-bielefeld.de. Accessed: Novemebr 10th 2001.

Socher, G., G. Sagerer, et al. (1996). Integration of Image and Speech Understanding in a Hybrid Distributed System. In proceeding of ICIP 1996 Volume 2 pp. 809-812 Lausanne Available: www.techfak.uni-bielefeld.de. Accessed.

Stent, A., J. Dowding, et al. (1999). The CommandTalk Spoken Dialogue System. In Proceedings of Thirty-Seventh Annual Meeting of the ACL pp. 183-190, University of Maryland, College Park, MD Available: http://www.ai.sri.com/~lesaf/commandtalk.html. Accessed: 24th September 2002.

Vorwerg, C., G. Socher, et al. (1997). Projective relations for 3D space: Computational model, applications, and psychological evaluation. Originally presented at: AAAI'97, Providence Rhode Island, pp. 159-164, 1997 Available: www.techfak.uni-bielefeld.de. Accessed: August 5 2001.

Winograd, T. (1973). "Procedural Model of Language Understanding" in Computer Models of Thought and Language. R. C. Schank and K. M. Colby. San Francisco, W. H. Freeman and Company**:** 152-186.

Yamada, A. (1993). Studies in Spatial Descriptions Understanding based on Geometric Constraints Satisfaction., Kyoto**:** 160.