Populating Semantic Virtual Environments

Abstract

The capacity for knowledge representation within simulated environments is a growing field of research geared towards the inclusion of detailed descriptors within the environment through which to provide a virtual agent the knowledge necessary to afford decision-making and interaction. One such layer of representation is that concerning the natural-language, or semantic, depiction of the environment and the objects within it. However, even as technology grows more capable of representing such information-rich environments the process of authoring and injecting this knowledge has largely remained a manual effort. Not only is this effort arduous for the author in both time and energy expenditure required, but the process also lends itself to limiting the ontology of the simulation further constraining the extent to which such knowledge may be used.

Here we offer an affordable method for semi-automating the generation and injection of semantic properties into a virtual environment for the purpose of producing more natural agent-object interaction.

Keywords: Smart Objects, Autonomous Actors, Virtual Humans, Semantics and Ontologies for Virtual Environments

1 Introduction

The utilizaion of semantics representation as a key component in the construction of virtual environments is a growing practice both in academia and industry, particularly as a means by which to afford virtual agents a greater form of autonomy [1]. Regardless, the current process by which semantics and their respective ontologies are constructed is unable to adequately keep up with what is required of them.

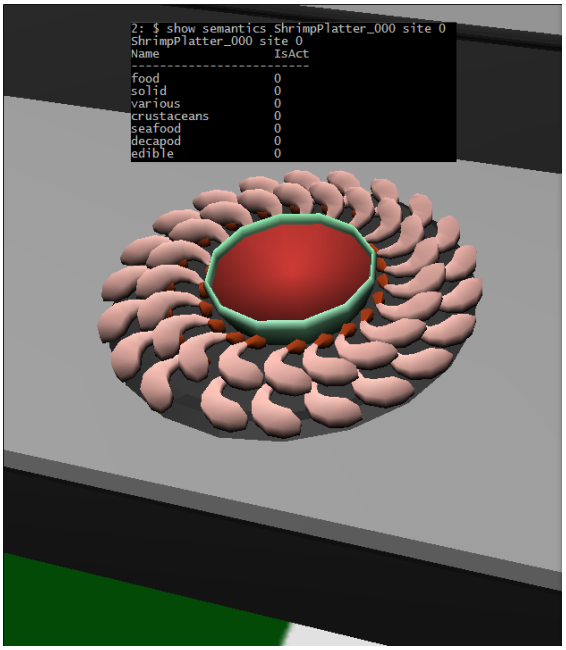

Figure 1: A `ShrimpPlatter` virtual object with embedded semantic properties.

We present in this paper a method and associated set of tools to facilitate the construction of semantic ontologies for virtual environments, as well as the generation of semantics themselves. Section 2 discusses related work in the field of semantic ontologies and their application. Section 3 further explores the issues surrounding current efforts at incorporating increasingly large semantic ontologies into virtual environmnets. Section 4 presents our tools authored in response to these efforts and details regarding their function and capability. The paper concludes with a discussion of potential applications of this approach, as well future work.

2 Related Work

The concept of object *affordance* offers us a framework by which to characterize an agent's ability to perceive and interact with the environment in which it exists. While the notion of an affordance varies in its use within the field of human-computer interation [2], the definition as originally put forth by psychologist James J. Gibson [3] forms the basis on which this research is motivated.

Gibson's affordances are highlighted through three defining characteristics [2]:

- Affordances present to an agent potential interactions it may perform with respect to that object's current environment.
- The existance of an affordance is independent of the agent's ability to perceive it.
- The existance of an affordance is binary.

With particular respect to the first point, affordances not only encapsulate the specific actions themselves but descriptors which may in turn be used to facilitate the performance of action.

Smart objects, as introduced by Kallmann and Thalmann [4], apply the concept of affordances to objects within a virtual environment in a more generic manner. A smart object illustrates that object's functionality, possible interactions with, and low-level manipulation descriptors in an effort to enable real-time interactions between the agent and itself.

While the notion of smart objects is more concerned with providing the abstracted behavioral representations and functional aspects of an object in application [5, 6, 1], this method for representing affordances in a direct manner to the agent provides a base through which to provide most high-level descriptions.

The development of natural language ontologies for the representation of objects and their affordances has been well-explored within the field of AI. Bindiganavale el. al. [7] pioneered the conceptualization of actions into an ontology to support the execution of actions using natural language instructions. Pellins et. al. [8] have introduced the use of semantics for the construction of virtual enviromemts at a conceptual level. Kalogerakis et. al. [9] proposed the use of ontologies for the structuring of content within objects of a virtual environment using OWL graphs, while Pittarello et. al. [10] similarly explored the use of X3D for annotating interactive environments with textual information in a hierarchal form. Balint et. al. [11] likewise studied the construction of hierarchies of ontologies using semantic information to facilitate agent behavior.

Our implementation does not attempt to create its own ontology but rather combines existing ontologies into a single framework upon which virtual environments may be enriched and maintained.

3 Incorporating Semantics

When handling the incorporation of semantics into the virtual environment there are two primary issues that need to be addressed:

- The quality of the data being included.
- The inherent trade-off between the quantity of information within a smart object and that object's extensibility.

In order to ensure the quality of the semantic data within an object we utilize pre-fabricated, open source lexical corpa. To allow for the inclusion of incresing amounts of object-specific data we explore the further abstraction of smart objects through modularization and runtime-specific attributes.

3.1 Lexical Databases

As stated prior, the hand-crafting of object hierarchies and semantic ontologies in the production of virtual environments suffers from numerous pitfalls inherent in the manual approach itself. Such systems are often manufactured around specific applications in simulation, resulting in inflexibility with regards to scalability and increased overhead in the time and effort required to maintain the model [1]. Additionally, leaving the simulation author to construct their own ontology not only detracts from time better spent concerned with application of said ontology, but also may result in models that are, while functional, fundamentally incorrect in their construction, resulting in problems later on.

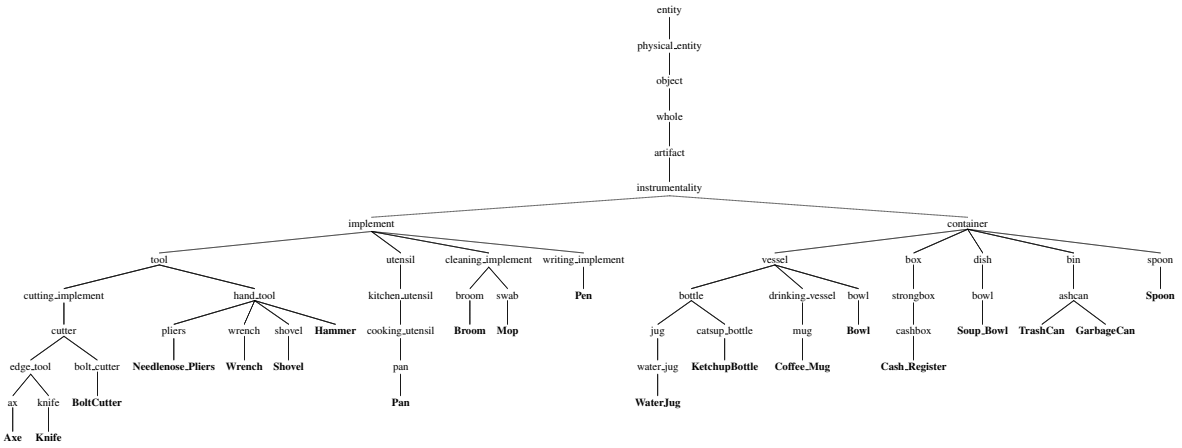

Figure 2: A Sample object hierarchy as generated through WordNet.

Many research endeavours have attempted to assist in the production of relational hierarchies through such means as crowdsourcing [12, 13]. Crowdsourcing by nature operates under an open world model, under the formal logic assumption that the truth of a statement is independent of whether or not it is known by any one individual to be true [14]. Irrespective of preventative steps taken to ensure the integrity of collected input, the gathering of information from such a diverse population results in data that may be misleading, irrevelant, or otherwiase incorrect. This approach is fundamentally in conflict with the closed-world model under which we construct ontologies for game and simulation, wherein what is known is fact and what is not is assumed false. While it should not be denied that crowdsourced data has its application, there exist numerous drawbacks which make it highly impractical for use within the context of a closed-world system.

In order to provide for a more consistent means of reliable data collection we approach the use instead of established lexical corpa such as WordNet [15], VerbNet [16], and PropBank [17]. The primary focus on the use of these corpa is their freedom of availability and ease of accessibility. Additionally, for each corpus there exist a variety of developent tools, facilitating not only the manipulation of corpus data but also the incorporation of this data into a development cycle. Finally, the corpa are set up such that the information contained within any one corpus may be used in conjunction with the other two corpa for the purposes of creating more complex, complete environments.

3.2 Modularized Smart Objects

As originally shown by Kallmann and Thallmann [4], there exists a disjunction between the specificity of information that may be contained within a smart object, and the extensibility of that object with respect to its applicability beyond the given scenario. While the overloading of smart objects ostensibly results in the production of entities which sit in direct contrast to the core principles of the theory, strict adherence to the prescribed level of abstraction effectively limits the extent to which an implementation may otherwise allow for the enhancement of a scene.

Therefore, in order allow for the inclusion of increasingly detailed, domain-specific information in the construction of smart objects, while maintaining the concept's fundamental notions of generality and extensibility, it is necessary to alter the approach itself by which smart objects are traditionally constructed by introducing further abstraction in the form of smart object modularization. Through further separating of the knowledge representaion of objects from its graphical representation, this approach enables us to treat smart objects as more capable, general entities than previously permitted.

4 Semantic Generation

The system is built in two parts. The *pre-runtime* component encompasses the two tools

for semi-automating the generation of both an object hierarchy as well as the requisite semantic modules for the produced hierarchy. The *runtime* component is composed of the *Semantic Module Handler* (*SMH*) which acts as abstracted knowledge layer between simulation agent and object for managing the loading, dissemination, and interaction with all semantics in an environment.

4.1 Hierarchy Generation

All pre-runtime components have been built using *Python*, an interpreted object-oriented language [18]. *Python* scripts are easily testable and capable of cross-platform operation. The process by which object hierarchies are generated makes heavy use of the WordNet corpus, as controlled through the use of the *Natural Language Processing Toolkit* (NLTK) [19].

The WordNet database itself is organized into a hieracrhy of *synsets*, or synonym sets, of the form `word.PartOfSpeech.synsetNumber` (e.g. `rabbit.n.01`). Each synset is representative of a single meaning of a given word, for which there may exist multiple definitions. The construction of the WordNet hierarchy itself is that of a directed acyclic graph with all nodes extending from the most general form, `entity.n.01`. Therefore the challenge in constructing hierarchies from this data comes from first associating each object in a scene to its proper synset definition, then properly trimming the resulting hierarchy into a tree structure more suitable for use in simulation.

To assist the simulation author in the construction of an object hierarchy, all that is required is a list of all the unique objects to be represented within a scene, labeled in the form

```
object[ :  keywords ]
```

The keyword arguments constitute assistive text used in determining the proper synset to be associated with the given object. While optional, if none are provided the author may be prompted to select a specific definition upon hierarchy generation.

For example, if an environment is to include a number of common household pets, the author may include the following :

```
rabbit : fur ears lagomorph
cat
dog : canine mammal friend
```

An example hierarchy generated using this method is presented in Figure 2.

Once a hierarchy is generated, a MySQL table is injected into the simulation database ready for immediate use by the author. Additionally, at this stage all appropriate semantic modules are generated and the object-module associations stored within the database.

Objects	Found	Mismatch	Accuracy (%)	Time (s)
50	49	0	98.00	12.97
100	100	5	95.00	18.70
150	150	5	96.67	24.83
200	198	4	97.00	33.02
250	249	3	98.40	37.06
300	299	8	97.00	42.07
350	349	5	98.29	55.16
400	400	11	97.25	62.19
450	447	11	96.89	67.98
500	498	6	98.40	69.96

Figure 4: Testing of the hierarchy generation process was conducted over an increasing number of randomly-generated objects, measuring the degree to which synsets were correctly mapped back.

Tests were performed by creating object groupings of various size, consisting of objects extracted from randomly-selected WordNet synsets. For each hierarchy, the items were organized into a list in the previously described `object : keywords` format and the resulting list used to generate a hierarchy using our tool. The output list of paired synsets was then compared with the original list to determine overall accuracy.

This method for generation has been shown to be exceedingly capable in its ability to accurately construct object hierarchies, with an average 97.29% success rate, as demonstrated in Figure 4. The measurement of time shown relates not only to the construction of the hierarchy, but also the generation of all related semantics modules, as well as the writing of all necessary information to the database. For every 50 objects there appears an average 6.33 second increase in overhead (Figure 5).

The process iteself is a one-time operation overhead; once a hierarchy is produced there is no need to re-run the tool except for when modifications to the hierarchy itself are made.

4.2 Semantic Modularization

The underlying focus of modularizing smart object components to facilitate the construction of better, more accessible virtual environments relies not only on the ability of the system to manage this information during runtime, but also on the capacity to organize and manage this information during the design process. For these reasons, an XML-like schema was used utilized, as demonstrated in Figure 6.

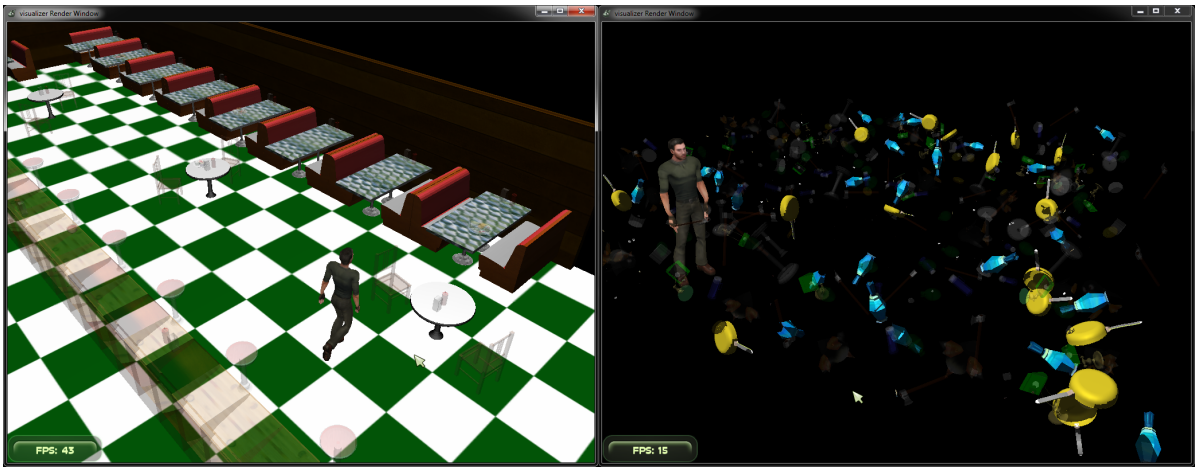

Figure 3: The agent interacts with the virtual environment with the assistance of semantic affordances.

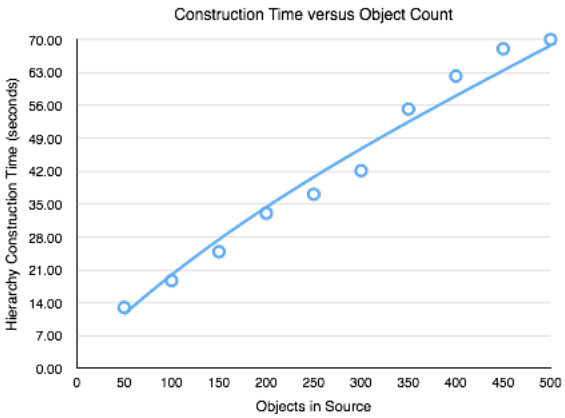

Figure 5: Increasing the number of unique objects in a scene has minimal effect on the time taken to compute a new hierarchy and generate the appropriate semantic modules.

bowl_01.mod

```
<name>bowl_01</name>
<modType>object</modType>
<class>PhysicalObject</class>
<state>entity</state>
<qualitative>
    <prop val="round"/>
    <prop val="vessel"/>
    <prop val="holding"/>
    <prop val="top"/>
    <prop val="food"/>
    <prop val="container"/>
    <act val="hold"/>
    <act val="contain"/>
</qualitative>
```

Figure 6: A sample semantic module containg relational semantics for object `bowl`.

The structure of each file is provided to be easy to maintain and extend by the simulation author. The name of the file corresponds to the specific WordNet synset from which it was produced to avoid conflict with similarly-defined objects. Semantic affordances are stored as `qualitative` properties of the form `<type value>`. This allows us to further delineate our semantics into textual descriptors relating to higher-level illustrative **prop**erties and those relating more directly to specific **act**ions which may be afforded by the object.

Modules themselves are not limited to establishing attributes of specific objects as the example demonstrates, but may be defined instead as collections of related affordances of any general type. As a result, it is possible to have modules such as `Dry`, `Mammal`, or `John`.

To specify the relation of modules to their respective virtual objects at runtime, the existence of these modules are stored in an indexed list within the provided runtime database. Correspondences between `object` and `module` are then stored in a separate table which is then referred to when loading objects into an environment.

4.3 Runtime Performance

The management of semantic modules at runtime is controled by a separate *handler*, the *SMH*, in an implementation similar to the actionary as developed by Bindiganavale el. al. [7]. A hierarchical approach was necessary to allow for a greater level of precision and control over the dissemination of semantics accross an increasingly complex virtual environment.

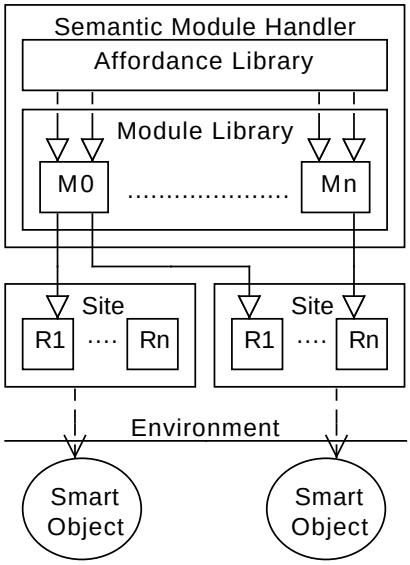

Figure 7: An overview of the semantic injection system hierarchy.

Modules are loaded into the simulation once on an per-need basis, utilizing the open source tool RapidXML [20] to limit the overhead cost to fractions of a second per file. As each file is loaded a corresponding container, an *SMod*, is created into which the appropriate semantics are stored, to be distributed to further instances of the object without necessitating subsequent rereadings. Instances of *smart objects* themselves are appended with *site* containers which act as intermediaries between the object and handler. The *site* containers themselves store *semantic regions*, which are capable of representing individual segments or the whole of an object. It is these *regions* into which semantics from the *handler* are transferred and subsequently interacted with. The complete system hierarchy is illustrated in Figure 7.

Given a hierarchy constructed of 50 objects chosen at random, with a resulting 300 generated semantic values, the scalability of the system was tested over a set of randomly-generated environments. The two primary areas of concern over the applicability of the given implementation were the initial overhead of injecting semantics into the environment (Figure 8) and the processing time required for semantic-based agent-object interaction (Figure 10).

The processes of loading an environment and injecting it with the necessary semantics occur in parallel. Therefore, in order to measure the speed at which semantic affordances were both loaded into the system as well as disseminated across all appropriate virtual objects, it was necessary to first calculate the average time required to load each scene without the given affordances. From this we were able to approximate the injection speed of semantics within each environment, as shown in Figure 8. Overall, the speed at which semantics were injected increased at a rate proportional to two seconds per 100 objects, or 600 semantic inclusions (Figure 9).

Objects	Approx. Semantic Count	Injection Speed (s)	Total Load Time (s)
100	600	13	36
200	1200	14	37
300	1800	17	40
400	2400	18	41
500	3000	21	44

Figure 8: The measured performance of the distribution of semantic affordances across a suite of increasingly large randomly-generated environments.

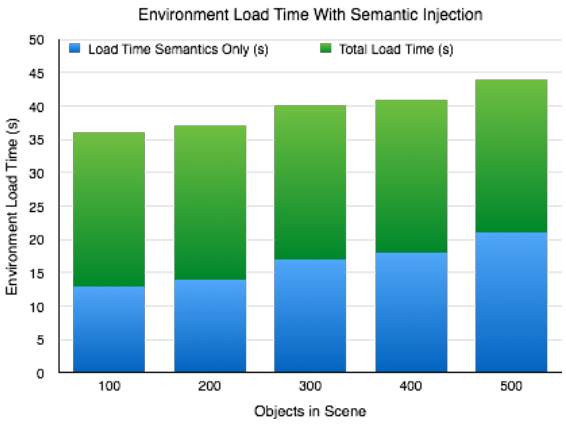

Figure 9: The time necessary to inject a virtual environment with semantic affordances proportional to the total load time of the scene.

While the preprocess overhead time required for injection is demonstratably linear with respect to the size of the environment, it must also be computationally feasible to utilize these semantics during runtime, particularly through the carrying out of agent tasks. To test this, the agent was placed within the generated environments and directed to walk from one end to the other, such that all objects were caused to enter its field of perception. Figure 10 presents the results of these tests. The complexity of agent search through its environment is roughly $O(log(n))$, with respect to the number of searchable semantics.

Objects	Approx. Semantic Count	Average Search Time (ms)
100	600	1
200	1200	2
300	1800	2
400	2400	3
500	3000	3

Figure 10: The process of searching through the semantics of objects within an agent's field of perception has a negligible impact on the overall performance of the simulation.

5 Conclusion

In this paper we have proposed a method geared towards facilitating the construction of semantic ontologies for virtual environments, as well as an approach for the generation of semantic affordances and their inclusion within these environments. The method has been shown to be computationally efficient in its implenentation and scalable across diverse environments of varying size and semantic inclusion.

A future extension of this work would be to more fully integrate the framework with high-level agent planning systems, using the embedded semantics in a more direct manner in order to assist behaviour and decision making processes.

References

[1] Tim Tutenel, Rafael Bidarra, Ruben M Smelik, and Klaas Jan De Kraker. The role of semantics in games and simulations. *Computers in Entertainment (CIE)*, 6(4):57, 2008.

[2] Joanna McGrenere and Wayne Ho. Affordances: Clarifying and evolving a concept. In *Graphics Interface*, volume 2000, pages 179–186, 2000.

[3] James Jerome Gibson. *The ecological approach to visual perception*. Routledge, 1986.

[4] Marcelo Kallmann and Daniel Thalmann. Modeling behaviors of interactive objects for real-time virtual environments. *Journal of Visual Languages & Computing*, 13(2):177–195, 2002.

[5] Jean-Luc Lugrin and Marc Cavazza. Making sense of virtual environments: action representation, grounding and common sense. In *Proceedings of the 12th international conference on Intelligent user interfaces*, pages 225–234. ACM, 2007.

[6] Marc Cavazza, Simon Hartley, Jean-Luc Lugrin, and Mikael Le Bras. Qualitative physics in virtual environments. In *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 54–61. ACM, 2004.

[7] Rama Bindiganavale, William Schuler, Jan M. Allbeck, Norman I. Badler, Aravind K. Joshi, and Martha Palmer. Dynamically altering agent behaviors using natural language instructions. In *Proceedings of the Fourth International Conference on Autonomous Agents*, AGENTS '00, pages 293–300, New York, NY, USA, 2000. ACM.

[8] Bram Pellens, Olga De Troyer, Wesley Bille, Frederic Kleinermann, and Raul Romero. An ontology-driven approach for modeling behavior in virtual environments. In *Proceedings of the 2005 OTM Confederated International Conference on On the Move to Meaningful Internet Systems*, OTM'05, pages 1215–1224, Berlin, Heidelberg, 2005. Springer-Verlag.

[9] Evangelos Kalogerakis, Stavros Christodoulakis, and Nektarios Moumoutzis. Coupling ontologies with graphics content for knowledge driven visualization. In *Virtual Reality Conference, 2006*, pages 43–50. IEEE, 2006.

[10] Fabio Pittarello and Alessandro De Faveri. Semantic description of 3d environments: a proposal based on web standards. In *Proceedings of the eleventh international conference on 3D web technology*, pages 85–95. ACM, 2006.

[11] John T Balint and Jan M Allbeck. Macgyver virtual agents: using ontologies and hierarchies for resourceful virtual human decision-making. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1153–1154. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[12] Massachusetts Institute of Technology. ConceptNet 5, 2013. http://conceptnet5.media.mit.edu.

[13] Kai Eckert, Mathias Niepert, Christof Niemann, Cameron Buckner, Colin Allen, and Heiner Stuckenschmidt. Crowdsourcing the assembly of concept hierarchies. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 139–148. ACM, 2010.

[14] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Englewood Cliffs, 1995.

[15] Princeton University. About WordNet, 2010. http://wordnet.princeton.edu.

[16] Martha Palmer and Karin Kipper. VerbNet: A Class-Based Verb Lexicon, 2013. http://verbs.colorado.edu.

[17] Martha Palmer and Mitch Marcus. Proposition Bank, 2013. http://verbs.colorado.edu.

[18] Guido Van Rossum et al. Python programming language. In *USENIX Annual Technical Conference*, 2007.

[19] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[20] Marcin Kalicinski. RapidXml. http://rapidxml.sourceforge.net/, 2009.

[21] Marcelo Kallmann and Daniel Thalmann. Modeling objects for interaction tasks. In *Computer Animation and Simulation98*, pages 73–86. Springer, 1999.

[22] Tolga Abaci, Jan Ciger, and Daniel Thalmann. Planning with smart objects. In *WSCG (Short Papers)*, pages 25–28. Citeseer, 2005.

[23] Isaac Dart and Mark J Nelson. Smart terrain causality chains for adventure-game puzzle generation. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 328–334. IEEE, 2012.

[24] Tolga Abaci and Daniel Thalmann. Planning with smart objects. In *In The 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision: WSCG*, pages 25–28, 2005.

[25] Henry Lieberman, Hugo Liu, Push Singh, and Barbara Barry. Beating common sense into interactive applications. *AI Magazine*, 25(4):63, 2004.