Services in Game Worlds: a Semantic Approach to Improve Object Interaction

Jassin Kessing, Tim Tutenel, and Rafael Bidarra

Computer Graphics Group
Delft University of Technology, The Netherlands
JassinKessing@gmail.com, T.Tutenel@tudelft.nl, R.Bidarra@ewi.tudelft.nl

Abstract. To increase a player's immersion in the game world, its objects should behave as one would reasonably expect. For this, it is now becoming increasingly clear that what game objects really miss is richer semantics, not eye-catching visuals. Current games' lack of semantics is mostly due to the difficulty of game designers to realize such complex objects. This paper proposes a solution to this problem in the form of services, characterizing classes of game objects. An example of this is the service of a vending machine, which exchanges a coin for a soda. A three-phased methodology is presented to incrementally specify and add services to game objects. This approach has been implemented and validated by means of a prototype system, which enables a simple and intuitive definition of services in an integrated environment. It is concluded that game objects aware of their services facilitate more and better object interaction, therefore improving gameplay as well.

Key words: game worlds, services, semantics, object interaction

1 Introduction

Look around and you will probably see objects scattered all around the place. If the same room would be used as the virtual environment of a game, one would probably want to see the same objects - or at least some objects - because empty environments are unnatural to walk through. Game environments that are filled with objects will therefore help immerse the player into the game world. By using graphics, animations, and physics, virtual objects could appear as players expect. However, that only accounts for their visual aspect, because most objects in games are still useless, being there for decoration purposes only. Only few objects, which are crucial for the game progress, are made functional.

An example is the role-playing game *The Elder Scrolls IV: Oblivion* [1], where objects can be picked up and used on specific locations to trigger an event, like opening a door with a key. Some objects have an effect on the player's avatar, for instance eating bread to increase the health, or wearing armor to increase the level of defense. In the game *Alone in the Dark* [2], several objects, which look useless at first sight, can be combined to create a functional object: a full battery and an empty flashlight will provide a light in the dark when combined.

In the examples above, the functionality of each object (including its meaning, roles, etc.) was thought up by the game designer and implemented by the programmer. In the real world, a particular object may assume other functions or roles never anticipated by its designer; with a game object in a virtual world, this is definitely not (yet) the case, and certainly not automatically. This limitation makes it impossible for a player to (make his avatar) interact with a game object in many reasonable ways.

In the fields of linguistics, computer science and psychology, *semantics* is the study of meaning in communication. When focusing on virtual environments for computer games, semantics is the information conveying the meaning of (an object in) a virtual world [3]. A serious problem in current game development is a lack of tools to easily specify and add semantics to objects, resulting in a lack of object semantics in games. With a semantically rich object representation, virtual objects assume behaviors like in the real world, instead of consisting of a geometric model only. This can be illustrated with a few examples. When eaten by a character, an apple will reduce the hunger level of that character; in other words, an apple provides the *service* of satisfying someone's hunger. A coat serves its wearer for warmth. A fire, however, will provide warmth to everyone in the area. And a vending machine has the service to supply cans of soda, but only after it has received money.

The role of semantics in virtual environments is receiving increasing attention [3], but so far not much research has been done on adding semantics to game objects, let alone with the purpose of making them more functional or improving the overall gameplay. This paper focuses on our research efforts to improve the semantics of objects placed within game worlds. In particular, the notion of services is proposed, by which virtual objects get to 'know' about their roles in the world, how they can affect other entities (including the player's avatar or artificial agents), and how others can interact with them. Empowered with the notion of services, objects acquire their own behavior, instead of a purely predefined behavior; they can behave as one expects, and correspondingly one is able to interact with them as one expects, regardless of whether the virtual object is completely imaginary or mimicking some real world object. We believe that this can significantly change and improve the gameplay, as players will be able to express their creativity and find more paths to achieve the same goal. Enabling game developer teams to easily declare services and assign them to object classes, as described in this paper, is a major step towards the ultimate goal of achieving more and better object interaction.

2 Related Work

Smart objects [4] were a successful proposal for adding semantics to virtual objects, dealing with many of the possible user interactions in a virtual environment. Noticeably, smart objects were primarily devised for manipulation, animation, and planning purposes, like grasping, pulling, or rotating (individual parts of) objects. An example is an artificial agent that can open a door by

moving its hand to the door knob, using the correct hand posture, and turning the knob. Although smart objects are powerful for these purposes, they lack the information of which services they provide to their users.

Research in artificial intelligence (AI) proposed the notion of ontologies, due to the lack of shareable and reusable knowledge bases. An ontology is an explicit specification of a conceptualization: a representational vocabulary for a shared domain of knowledge, in the form of human-readable and machine-enforceable definitions of classes, relations, functions, and other objects [5]. When placing ontologies in the context of this research, they define the meaning of objects and the relations between them. In ontologies, important relationships are *generalization and inheritance*, where classes are connected, and each subclass inherits the features of its superclass [6]. The class *Car*, for example, has the class *Vehicle* as its parent. Another important relation is *instantiation*, which relates a class with each of the individuals that constitute it. A *Ferrari*, for instance, is a kind of *Car*. In Section 4, the usefulness of these two relations for the creation of objects with services will become apparent.

3 Designing Services

In order to design services for game objects, it can be very useful to analyze how real world objects can be structured and classified. In particular, we can identify the notions of *class* and *attribute*. Each object in the real world can be said to belong to some class, defined as 'a generic description of a collection of entities based on their essential common attributes'. An attribute, in turn, is defined as 'a characteristic of an entity'. Classes, therefore, describe entities, varying from physical objects like 'apples' and 'people', to substances like 'water'. For attributes, one can think of abstract attributes like 'edibility', or physical attributes like 'mass'. Units and states express the values of attributes. The 'mass' attribute could be expressed in the units 'kilograms' or 'ounces', while 'edibility' could be expressed in the states 'edible' or 'inedible'. Units are also required to express substances, because they are not quantifiable in integer values only, unlike physical objects.

The notions introduced above give us a foundation for the definition of services. In the real world, entities have particular functions, and provide services, and this should also be the case for entities in a virtual world; for example, a coat provides the service of supplying warmth, but only when it is worn. We define a *service* as 'the capacity of an entity to perform an action, possibly subject to some requirements'.

An *action* can then be described as 'a process performed by an entity, yielding some attribute value changes or (new) entities'. Actions are best illustrated by some examples. A service of a heater is to heat the entities in the surrounding area. This means that the values of their temperature attribute rise; see Fig. 1. Attribute value changes do not have to occur to target entities only; they can also affect the actor. Consider an avatar punching an enemy, which lowers the enemy's health, but also increases the avatar's fatigue. The service of a vending

machine, supplying a soda, is a good example of an action yielding an entity; see again Fig. 1. This soda is an entity that is supplied from the inventory of the vending machine. However, it is not necessary that the actor always has a stock of existing entities that can be supplied, as an entity's action can also yield new entities. An example is a saw machine that requires trunks, and processes them into wooden planks, which are new entities. This process leads us to the notion of *service requirements*: they can be either actions (e.g. the coat should be worn, and the saw machine should be given trunks) or some attribute constraints, as for example a range of values/states (e.g. electrical devices should be powered on before performing an action, and the fatigue level of the avatar should not be too high before being able to fight).

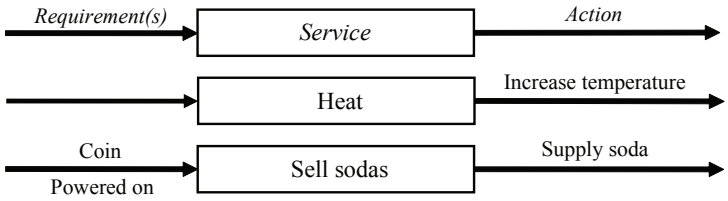

Fig. 1. A generic service, a service of a heater, and a service of a vending machine.

From the examples above, there are four important elements that should also be taken into account when designing services. First, quantities are essential to indicate how many entities are exchanged during an action, or in which amount an entity is exchanged, in case of substances. Second, temporal properties are relevant, because they indicate the duration of a service, which could be a one-time event, or last for some amount of time. Third, spatial properties indicate who or what is affected by a service, e.g. the consumer, or all entities within a certain radius. Finally, a sequence of interaction steps indicates the order in which requirements should be met before performing an action.

4 Services Put to Work: a Three-phased Approach

The concepts developed in the previous sections have been implemented in a prototype system which supports the definition of services for game objects step by step. This system covers the three main phases that were identified in the object design process: (i) a *specification phase*, in which generic classes are specified in a library, (ii) a *customization phase*, where a selection of classes from that library is customized into concrete game-specific classes, and (iii) an *instantiation phase*, where object instances of these game-specific classes are placed in a game world. Figure 2 gives an overview of these three phases.

The key idea of the specification phase is to create a library of generic semantic classes. By designing generic classes that can be used in all kinds of virtual

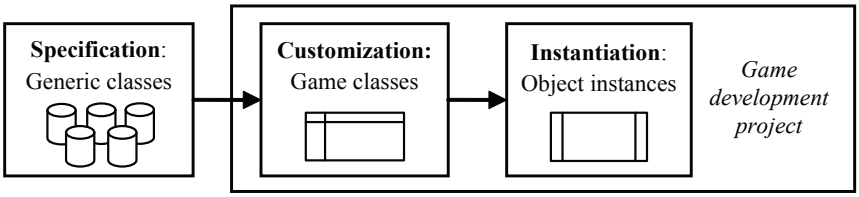

Fig. 2. A phased approach with generic classes in the specification phase, game-specific classes in the customization phase, and object instances in the instantiation phase.

worlds, consistency and reusability are stimulated, and thus development time reduced. In this phase, libraries of classes, attributes, units, states, and actions are created. Relations can be established among these components, and for each class, services can be defined in order to specify its semantics. By applying inheritance, a class hierarchy is developed, with attributes and services that have been assigned to a class being inherited by all its children. In this way, for example, an attribute like 'mass' does not have to be defined for each single class, but to the 'physical object' class only. To populate these libraries, the WordNet database [7] was used, as it contains many nouns and verbs in the English language, being therefore useful for many possible classes, attributes, actions, etc.

In contrast with the specification phase, the other two phases are not generic. Instead, customization and instantiation play a central role during each particular game project, which typically has its own unique environment, object style and desired behavior, etc. In the customization phase, specific game classes are derived from the generic classes from the first phase, thus automatically inheriting all their generic semantics, including their set of attributes, and also the services they provide. It suffices then to customize the specific behavior desired for this particular project, including their specific attribute values, e.g. quantities and temporal properties. The customization phase is also the right time to assign the project-specific 3D models to the relevant game classes. Finally, in the instantiation phase, instances of the customized game classes can be created and placed inside a game world, which is done by means of a level editor specifically created for this purpose.

5 Conclusions

Despite exuberant visuals, most current games considerably lack proper semantics in the objects populating their virtual worlds. This is partly because designing semantic objects poses especially difficult challenges, including the inherent complexity of maintaining and scaling all interactions among such objects. This paper presented a solution to that problem in the form of services, specified as characteristics of classes of objects. A three-phased methodology has been presented that enables a game development team to incrementally specify and add services to game objects. This approach has been implemented and validated by

means of a prototype system, providing an integrated environment which effectively supports a simple and intuitive definition of services. Among the numerous advantages of this approach, among them (i) it promotes reusability of previously specified object semantics, (ii) it easily supports behavior customization as required by each specific game, and (iii) it seamlessly blends with our semantics engine, charged with all service handling during the game (analogously to what a physics engine does with in-game physics).

We believe that enabling designers to create game objects that are aware of each other's services will be instrumental to achieve more and better object interaction. This in turn is considered one of the key conditions to significantly improve gameplay. However, it should also be stressed that object semantics isn't but a (powerful) means to serve the gameplay. In particular, it will never automatically make dispensable the creative work of designers. On the contrary, care should be taken to avoid overloading objects with superfluous semantics, as semantics make virtual objects not only more realistic, but more complex as well, which could end up undermining the gameplay. Therefore, it is the task of the game designer to seek a balance between achieving realism and good gameplay. The approach presented here, giving designers the possibility to include realistic semantics by means of services, while keeping much control on the fine-tuning of the behavior of their objects, is a valuable aid in that direction.

In the future, we would like to experiment with coupling our semantics engine with a game AI system, so that artificial agents can make use of services as well, in addition to players' avatars.

Acknowledgement. This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

References

1. Bethesda Game Studios: The Elder Scrolls IV: Oblivion. Bethesda Softworks (2006)
2. Eden Studios: Alone in the Dark. Atari (2008)
3. Tutenel, T., Bidarra, R., Smelik, R. M., de Kraker, K. J.: The Role of Semantics in Games and Simulations. In: Computers in Entertainment, vol. 6 (4). ACM, New York (2008)
4. Kallmann, M., Thalmann, D.: Modeling Objects for Interaction Tasks. In: Proceedings of the 9th Eurographics Workshop on Animation and Simulation, pp. 73–86. Lisbon (1998)
5. Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition, vol. 5 (2), pp. 199–220. Academic Press Ltd., London (1993)
6. Huhns, M. N., Singh, M. P.: Ontologies for Agents. In: IEEE Internet Computing, vol. 1 (6), pp. 81–83. IEEE, Piscataway (1997)
7. Miller, G.: WordNet: A Lexical Database for English. In: Communications of the ACM, vol. 38 (11), pp. 39–41. ACM, New York (1995)