Action semantics in Smart Objects Workshop Paper

Tolga Abacı
tolga.abaci@epfl.ch
http://vrlab.epfl.ch/˜ tabaci

Ján Cíger
jan.ciger@epfl.ch
http://vrlab.epfl.ch/˜ janoc

Daniel Thalmann
École Polytechnique Fédérale de Lausanne
Virtual Reality Laboratory
daniel.thalmann@epfl.ch
http://vrlab.epfl.ch/˜ thalmann

Abstract

We present a method of formal description of the action semantics in smart objects. Smart objects were primarily used for behavioral animation in the past. We demonstrate, how a formally described semantics can be used for action planning purposes by intelligent agents trying to achieve a goal. The described approach also reduces the complexity of common planning approaches by reducing the amount of information the agent has to process.

Keywords: Smart objects, artificial intelligence, virtual reality, planning

1. Introduction

There are significant AI and animation challenges to be overcome in contemporary virtual reality systems. Realism and believability of virtual characters plays important role in the immersion of the user, but from an engineering point of view, it has to stay simple enough to be feasible with the available computing resources. The only way how to achieve these two contradictory goals is to employ some engineering and mathematical tricks allowing the realistic-enough simulation but being much simpler to compute – e.g. inverse kinematics, simplified or completely absent dynamics and smart objects.

Virtual reality applications often require the virtual characters to be able to manipulate the objects in their environment. Such interactions can be arbitrarily complex and their accuracy requirements vary as well (i.e. ranging from simple, single-shot motions to sequences of numerous motions that require high accuracy). Traditional solution are pre-designed or pre-recorded (e.g. by motion capture) animations. Another, more general solution is to shift the responsibility for the animation at least partially to the object, leading to the smart object concept [1].

The basic animation and behavior problems are addressed by smart objects, however there are still major limitations. In order for the intelligent agents to be able to reason about the smart objects and the behavior they facilitate, the semantics of that behavior has to be described in a formal way. Typically, such reasoning is performed by action planners, a technique common in artificial intelligence. Planners require a formal description of the virtual environment, usually in the form of logic formulas describing the state and the actions possible in the virtual world (e.g. STRIPS operators and predicate calculus).

What we would like to explore in this paper

is a natural extension of the smart objects by describing the semantics of the actions which enables the use of the planners for complex interactions. Embedding the high-level information together with animation data in the smart object allows for more efficient planning, because only relevant operations and data are considered. Another advantage is that the embedding allows integration of the creation of the logic data into the design pipeline, together with other semantic information. This ensures that the smart object animations and the corresponding high-level information are created at the same time and in a consistent way.

2. Related work

There are few virtual environments supporting interactions between virtual humans and the virtual objects. Those that do usually follow a very limited, "hardwired" approach of small set of pre-defined animations. Nevertheless, the functionality to introduce interaction possibilities into a virtual environment is of great importance. There are a number of works in the literature that have addressed this issue. Parametrized action representation [2] describes an action by specifying conditions and execution steps, and supports chaining of actions. The Improv system [3] consists of an Animation Engine, used for the motion generation aspects and a Behavior Engine, used for describing the decision-making process through rules.

On the behavior front, virtual human – object interaction techniques were first specifically addressed in the object specific reasoner (OSR) [4]. The primary aim of this work is to bridge the gap between high-level AI planners and the low-level actions for objects, based on the observation that objects can be categorized with respect to how they are to be manipulated. However, the OSR is different from the work presented in this paper, because the author takes a "top-down" approach – generic actions are gradually refined with the use of object taxonomies into agent-executable actions.

Recently, Vosinakis and Panayiotopoulos have introduced the Task Definition Language [5], aimed at filling the gap between higher-level decision processes and an agent's interaction with the environment. This language supports complex high-level task descriptions through combination of parallel, sequential and conditionally executed built-in functions.

The smart objects paradigm has been introduced for interactions of virtual humans with virtual objects [1]. It considers objects as agents where for each object interaction features and plans are defined. We extended this model in the sense that we also augment the geometric description of the objects with additional information; our approach arranges this information in an extended scene graph hierarchy together with the geometry. Even though smart objects are more flexible than other approaches when it comes to animation and behaviors, the fact that interaction plans are typically fixed imposes a severe limitation from the interaction point of view and also reduces the capability to adapt to new situations.

To overcome the rigid constraints of pre-defined interaction plans, the intelligent agent has to be able to reason about the objects it has to interact with. One of such reasoning techniques is planning. It is one of the oldest topics in both artificial intelligence and robotics. Planning in robotics is usually concerned with synthesizing collision-free motion. On the other hand, AI understands planning as a search for a sequence of logic operators/actions that transform the initial state of the world into the desired goal state.

One of the first published works about AI planning is the STRIPS planner from 1971 [6]. This planner introduced the concept of operators, with preconditions and effects. The state of the world is expressed using predicate calculus. This method of describing the planning problem is still popular and was used in many planners - e.g. UCPOP [7], Prodigy [8]. One of the most popular planners using the STRIPS representation is Graphplan [9] and its many derivatives, such as Blackbox, Sensory Graphplan [10], Temporal Graphplan [11] and many others.

3. Action semantics in smart objects

Smart objects provide not only the geometric information necessary for displaying them on the

screen, but also semantic information useful for animation purposes. We store this information in the form of sets of attributes attached to the scene graph nodes of the object.

The attributes convey various kinds of information – e.g. important places on or around the object (e.g. where and how to position the hands of the virtual character in order to grasp it), animation sequences (e.g. a door opening) and general, non-geometric information associated with the object (e.g. weight or material properties) The semantic information in the smart object is used by the virtual characters to perform actions on/with the object, e.g. grasping, moving it, operating it (e.g. a machine or an elevator).

However, this simple geometric information does not provide enough data to the agent, if it has to reason about the possible interactions with the object. In order to be able to do this, every meaningful interaction has to be described in terms of its preconditions and its effects on the state of the agent/object, if the action is performed. Fortunately, such information can be easily encoded using the first order predicate calculus[1]. A detailed description of it and its use to describe possible actions by an intelligent agent can be found e.g. in [12].

In our case each possible interaction can be formally expressed in a form of a rule (assuming left-to-right evaluation) shown in figure 1. The formula describes a "preparepush"operation, where agent X prepares itself to move an object Y, if the given conditions are satisfied in some state of the world.

$$\forall X \forall Y (\exists P\ place(P) \wedge at(X,P) \wedge at(Y,P) \wedge agent(X) \wedge pushing(X,Y)) \Rightarrow preparepush(X,Y)$$

Figure 1: Predicate calculus expression for preparepush interaction

Such expression can be interpreted as describing a whole class of actions which can be obtained by substituting for the variables X, Y, P.

[1]To be exact, this is not completely correct – predicate calculus does not allow expression of actions because there is no notion of time. We are abusing the notation a bit here.

There are two main practical issues with this approach:

1. The amount of potential interactions rules for a realistic virtual reality scenario is huge and would put an unreasonable burden on the intelligent agent implementation. A "localized" implementation is desirable, where the agent knows only about relevant interactions, not all possible ones. The amount of information available to the agents has to be limited.

2. The mathematical notation used is unwieldy for computer processing. It is desirable to have a subset which is easier to parse and to interpret.

As mentioned above, the smart objects provide geometry-related semantic information. Kallmann described also the idea of "interaction plans" in [1]. Interaction plans are essentially scripts containing the animation of the action itself. They coordinate the animation of the virtual character and the object to create the intended result, which could be a complex animation of a virtual human pushing a crate, opening a door, etc.

```
(:action preparepush
  :params (?X ?Y)
  :precond (and (at ?X ?P)
                (at ?Y ?P)
                (agent ?X)
                (place ?P))
  :effect (pushing ?X ?Y))
```

Figure 2: STRIPS version of the preparepush interaction (operator)

The first issue can be easily addressed by an extension of the smart object paradigm. In order to provide the notion of locality, in addition to the generic behaviors defined by the agent's author, the interactions defined in the manipulated smart object will be made available to it as well. Furthermore, the interaction plans (scripts) contained in the smart object will be augmented with the formal description of the interaction.

To address the second issue, we will augment the interaction plans not with the rules expressed as predicate calculus but with an equivalent simpler notation instead – the STRIPS notation. The interaction described in figure 1 can

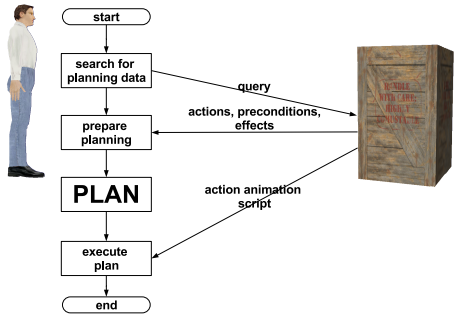

Figure 3: Planning process with the extended smart objects

be written also as in figure 2. This particular format is known as PDDL notation, used by many STRIPS-like planners for the problem description.

The STRIPS notation allows us to describe formally, when the interaction is possible – by means of preconditions and what exactly will be its effect on the state of the world. In the STRIPS notation, interaction described in this way is called "operator", to signify, that it described a whole class of possible interactions depending on the substitutions for the variables. An operator with all variables substituted (an instantiated operator) is called "action" and describes one single interaction.

In the typical usage scenario, the instantiated operators (actions) are matched against the current state of the system using unification. Unification also ensures consistent substitution for unbound variables (P in the examples), in effect playing the role of the $\exists$ quantification. The exact value of the variable is not important, however there has to be at least one such value which satisfies the given propositions (conditions).

The augmented smart objects can now be used in the action planning process of the intelligent agent. The outline of the process is described in figure 3.

The agent will use the semantic information from the smart object to supplement its own set of operators with object-specific knowledge, permitting it to correctly interact with it. In essence, the agent "learns" the object-specific information by querying the smart object at runtime. The full set of operators is used in the planning process, allowing the planner to schedule object-specific actions in the plan. Finally, during the execution of the plan, the object-specific actions are mapped to the interaction plans (scripts) stored in the smart object, executed and the state of the system is updated according the specified effects of the action.

4. Results

To verify the functionality of the extended smart objects, we have implemented a test case using our VHD++ framework (described in [13]). The goal was for a virtual human to move a crate in the virtual environment, while using the object-specific knowledge to properly animate the process.

An overview of the smart object structure is shown in figure 4. An abbreviated smart object definition is in appendix A. The smart object definition consists of several parts, apart of the transform defining the original position and attitude of the crate, there are several sets of attributes defining various important points on and around the smart object – e.g. proper hand position for moving the object and initial position for approaching it. The format is easily extensible, in order to formally define the semantics of the possible interactions, several new attribute sets were introduced:

- The "properties" attribute set – defines in a symbolic way the properties of the object. In our case, it establishes "small_box" as an object and declares it as not heavy.

- The operators. Operators are defined in separate attribute sets each, named with the name of the operator. Each consists of

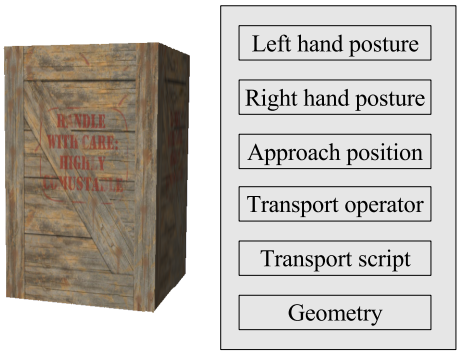

Figure 4: The "small box" smart object

Figure 5: Virtual human pushing a crate

three sections containing parameters, preconditions and effects of the operator. For each operator attribute set there is a corresponding interaction plan/script.

In our test case, the implementation of the "transport" operator described in the appendix A involves a complex script using inverse kinematics to for the agent to grasp the object properly and a walking engine to move the virtual character and, consequently, the grasped box. Figure 5 shows a snapshot from the resulting animation.

Our action planner implementation employs a modified version of Sensory Graphplan (SGP), originally developed at the University of Washington. Sensory Graphplan builds upon the standard Graphplan and adds sensing actions and conditional effects. It builds contingency plans - plans where the initial truth value of some predicate may be unknown (uncertain in the SGP terminology) and the planner plans for both eventualities indicating which actions have to be taken in each case (planning worlds - full description in [9]). As such, it is more suitable for virtual reality simulations because the input language is much more expressive compared to the standard STRIPS-like planners.

The advantage of putting the object-specific animation and formal semantic information into the smart object becomes obvious when we consider that in a real VR simulation the agent has to interact with large amount of objects having different properties and different animation needs. With the described extension of the smart objects, we can keep the amount of object-specific information in the intelligent agent minimal – e.g. it is enough for the agent to know that it has to perform a "transport" operation, but the object-specific details are resolved at run-time with the help of the smart object. In another experiment, we have defined a second smart object "big_box", which requires two virtual humans to transport it because it is heavy. The agent learns this property from the smart object, along with the modified "transport" operator which requires two agent to work together. Without having to change the high-level action plan, the system adapts to the changed conditions, resulting in action depicted in figure 6.

5. Conclusions

Our proposed extension of the smart object concept by defining a formal semantics of the possible interactions addresses the need of action planners for object-specific information usable in the reasoning process. Knowledge of the formal semantics of the interaction allows the intelligent agent to not only perform the animation but also to reason about the consequences of its actions. This reasoning is essential for any action planning process.

Furthermore, such extended smart objects enable the intelligent agent to "learn" how to interact with new, unknown objects on the fly, by exploiting the stored semantic information. It enables simpler, more generic design of the agents, which do not need to "know" about every possible object in the virtual world in advance. The agents can be kept very generic, allowing very good reusability and adaptability of the simulation system.

Figure 6: Virtual humans moving a large crate

Finally, storing the extended semantic information in the smart object has the advantage of simplifying the design process as well. The op-

erators have to be designed at the same time as the interaction scripts, this helps to ensure consistency between the formal semantics of the action and its real implementation.

In the future, we would like to focus on the integration of interaction semantics into the design process of smart object. At the moment all information has to be entered manually by writing XML code, which is error-prone. We envision a semi-automated tool where the smart object designer could create the semantic information for the provided interaction scripts by filling a template or having it generated automatically from manually entered information. Ideally, the semantics of the interaction should be generated automatically from the interaction script, however this is a very difficult problem, because the scripts can contain arbitrary code.

We have demonstrated, how smart objects could be extended from a purely geometric and animation technique into a valuable tool for reasoning of intelligent agents. The formal representation of the interaction contained in them allows the agents to meaningfully interact and create action plans even with objects never encountered before and not anticipated by the agent developer. The coupling between the animation and its formal semantics enables the virtual characters to perform complex actions which are very complex to achieve otherwise.

Acknowledgments

The work was sponsored by the Federal Office for Science and Education in the Framework of the EU Network of Excellence AIM@SHAPE.

References

[1] Marcelo Kallmann. *Object Interaction in Real-Time Virtual Environments*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

[2] Norman Badler, Rama Bindiganavale, Juliet Bourne, Martha Palmer, Jianping Shi, and William Schuler. A parameterized action representation for virtual human agents. In *Embodied Conversational Agents*, pages 256–284, Cambridge, MA, 2000. MIT Press.

[3] Ken Perlin and Athomas Goldberg. Improv: a system for scripting interactive actors in virtual worlds. *Computer Graphics*, 30:205–216, 1996.

[4] Libby Levison. *Connecting planning and acting via object–specific reasoning*. PhD thesis, CIS, University of Pennsylvania, 1996.

[5] Spyros Vosinakis and Themis Panayiotopoulos. A task definition language for virtual agents. *Journal of WSCG*, 11:512–519, 2003.

[6] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[7] J. Scott Penberthy and Daniel S. Weld. UCPOP: A sound, complete, partial-order planner for ADL. In *Third International Conference on Knowledge Representation and Reasoning (KR-92)*, Cambridge, MA, October 1992.

[8] Manuela Veloso, Jaime Carbonell, Alicia Perez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), 1995.

[9] Avrim L. Blum and Merrick L. Furst. Fast plannig through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[10] Daniel S. Weld, Corin R. Anderson, and David E. Smith. Extending Graphplan to handle uncertainty & sensing actions. In *Proceedings of AAAI '98*, 1998.

[11] David E. Smith and Daniel S. Weld. Temporal planning with mutual exclusion reasoning. In *IJCAI*, pages 326–337, 1999.

[12] George F. Luger. *Artificial Intelligence*. Addison Wesley, 4. edition, 2002.

[13] Michal Ponder, Tom Molet, George Papagiannakis, Nadia Magnenat-Thalmann, and Daniel Thalmann. VHD++ development framework: Towards extendible, component based VR/AR simulation engine featuring advanced virtual character technologies. In *Computer Graphics International 2003*, pages 96–104, 2003.

A. Smart object definition

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vhdHObjectProperty name = "small_box">
    <hobjObjectoid name = "small_box">
        <hobjMatrix for = "worldTransform">
            1.000000 0.000000 0.000000 0.000000
            0.000000 1.000000 0.000000 0.000000
            0.000000 0.000000 1.000000 0.000000
            0.000000 -1.20000 -3.000000 1.000000
        </hobjMatrix>

        <hobjGroup name = "Box01">
            <hobjVisualGeometry name = "smallboxgeom">
                <hobjFile>smallbox.osg</hobjFile>
            </hobjVisualGeometry>

            <hobjAttributeSet name = "lefthand1" annotation = "h
                <hobjMatrix for = "transform">
                    0.212211 -0.085698 -0.973459 0.000000
                    -0.018253 -0.996321 0.083732 0.000000
                    -0.977053 -0.000000 -0.212995 0.000000
                    0.248064 1.292640 -0.505559 1.000000
                </hobjMatrix>
            </hobjAttributeSet>

            <hobjAttributeSet name = "righthand1" annotation = '
                <hobjMatrix for = "transform">
                    -0.219345 -0.085698 0.971876 0.000000
                    0.018867 -0.996321 -0.083596 0.000000
                    0.975465 -0.000000 0.220155 0.000000
                    -0.206275 1.292640 -0.532656 1.000000
                </hobjMatrix>
            </hobjAttributeSet>

            <hobjAttributeSet name = "approach1" annotation = "p
                <hobjVector for = "position">
                    -0.001913 0.000000 -0.907755
                </hobjVector>
            </hobjAttributeSet>

            <hobjAttributeSet name = "operators_transport" annot
                <hobjText for = "lisp">
                    (:action transport
                     :parameters (?who ?what ?from ?to)
                     :precondition (and (at ?who ?from)
                                        (at ?what ?from)
                                        (not (= ?from ?to))
                                        (object ?what)
                                        (place ?to)
                                        (agent ?who)
                                        (or
                                           (connected ?from ?to)
                                           (connected ?to ?from)
                                        (not (heavy ?what)))
                    :effect (and (at ?what ?to)
                                 (at ?who ?to)
                                 (not (at ?who ?from))
                                 (not (at ?what ?from))))
                </hobjText>
            </hobjAttributeSet>

            <hobjAttributeSet name = "script_transport" annotati
                <hobjText for = "script">
                    ... omitted for brevity
                </hobjText>
            </hobjAttributeSet>
        </hobjGroup>
    </hobjObjectoid>
</vhdHObjectProperty>
```