Behavior Design of Game Character's Agent

Tony Dujardin, Jean-Christophe Routier
Université Lille 1
LIFL, CNRS, UMR 8022
F-59650 Villeneuve d'Ascq, France
Email: tony.dujardin@lifl.fr, jean-christophe.routier@lifl.fr

***Abstract*—In human-level simulations, like video games can be, the design of character's behaviors has an important impact on simulation realism. We propose to divide it into a *reasoning* part, dedicated to a planner, and an *individuality* part, assigned to an action selection mechanism. Applying the separation of declarative and procedural aspects, the principle is to provide every character's agent with the same procedural mechanisms: the planner and the action selection mechanism. Declarative knowledge is then used at the agent level to individualize the behavior.**

The contribution of this paper consists in a motivation-based action selection mechanism that allows individualization in behavior. The modularity provided by the motivations enables a large variety of behaviors for which the designer has to choose parameters. If the simulation of characters are our first motivation, the principles involved in the proposed motivation-based action mechanism are general enough to be used in other contexts.

***Index Terms*—behavior, simulations, games, action selection**

I. Introduction

The simulation of human behavior is an application field of AI that is complex and ambitious. Reaching such an AI is not yet at hand. However it is possible to consider simplified instances of this problem and to try to tackle them. Games, since they provide a well defined and bounded context, have always been a good AI target. We agree with authors in [1] to consider that video games are the good target, from experimental or application point of views, for research on believable behaviors. In particular the management of so called "Non Player Characters" (NPC) has to be considered. The construction of believable NPC behavior enhances the playability of video games, as well as the interest of the players since immersion is increased. It has become a real challenge for the video games industry. This paper considers this problem and makes a proposition that allows to achieve the design of various behaviors.

In ethology, a behavior is defined as the continuous agent-environment interactions. In other words, a behavior results from the sequence of actions that an agent undertakes in its environment. In this case, the environment must be considered broadly: it includes the surrounding "topological" environment and every factor influencing the agent. Thus we adopt the following definition:

Definition 1 (Behavior): The behavior of an agent is the result of the sequence of actions an agent performs in its environment.

In [2], following this approach, the author considers a behavior to be a joint product of the agent, the environment, and the observer interactions. For the author, the classification of a behavior depends on the observer point of view. Hence, to give a name to a behavior is a personal interpretation from the observer on actions that the agent has performed. The mention of an observer is important. It establishes that the notion of agent behavior must not be confused with that of agent architecture. Moreover, the introduction of an observer underlines the problem of the meaning of "behavior design". Since the classification of a behavior is subjective, to make a proposition for behavior design whose result receives unanimity is hopeless. So, the work of behavior design must be tackled considering the desired behavior in its broad lines rather than in detailed precise behavior.

An issue is that, actually, with the notion of agent behavior, comes not only the observation of the actions that the agent undertakes but also an appreciation of the personality of the agent is made. That is, through the observation of the undertaken actions, the observer builds its own agent's appreciation and classifies it according to his/her evaluation: rough, fickle, friendly, etc. Then, the problem for agent's behavior designers is not only to provide the agent with the ability to perform actions, but to design the agent such that it expresses some personality. This is the case in computer games. Involved non-player characters (NPC) must act but it increases the game's realism if these characters "behave differently" from each other. Another problem for the designer is the design itself. This task must not be too complex and it should be possible to easily obtain distinct behaviors for characters. The purpose of this paper is to make a proposition in this way. Since we have stated that aiming at creating a given precise behavior is meaningless, our principle is to consider that a character suffers influences that direct its choice of action and to play on these elements. This proposition favors the reuse of behaviors or elements of behavior.

We discuss the design of behavior in section 2 where we split it into reasoning and individuality. Section 3 gives definitions concerning the reasoning part. Section 4 presents the core of our proposition, which corresponds to the individuality part. Section 5 illustrates how this proposition can be implemented. Some related works are presented before the conclusion.

II. Behavior modeling

Definition 1 establishes that the behavior results from all actions that an agent performs in its environment. Obviously, these actions are selected among the agent's abilities in order to solve its goals. In most of cases, there exists more than one way to solve them. The choice of a solution rather than another depends on several constraints. Indeed, the behavior is influenced (positively or negatively) by **tendencies** as stated in a behavioral psychological theory developed by Albert Burloud [3]. A tendency can represent:

- neutrality, if it has no influence on the behavior,
- attraction, if it drives the behavior to do something,
- repulsion, if it tends to divert the behavior from doing something,
- inhibition, if it prevents absolutely the behavior to do something, this is a particular, extreme case, of the repulsion.

In [4], these tendencies are defined by cognitones (i.e. elementary particles defining the mental state of an agent according to the author) urging or coercing an agent to act. They arise from combination of **motivations** that are more basic cognitones. This approach is explained through a "*conative*" system proposed by the author. In this system, the motivations are at the basis of the tendencies that are forcing the agent's decision. The author proposes a classification of motivations into four categories according to their origin: *personal*, *environmental*, *relational* and *social motivations*. Each motivation may influence the agent behavior according to the tendencies it provides.

Obviously, the observable behavior of an agent depends on the actions it can perform, i.e. its abilities. For example, if an agent, to enter rooms, can only break the doors, it could be perceived by the observer as more brutal than an agent who has the ability to open them "cleanly". Therefore, for a behavior designer, assigning an ability or another to an agent is already a way to build its behavior.

To solve a goal, an agent builds a plan based on its abilities (and its knowledge). From this plan an action is selected and performed by the agent. This action is an element of the agent behavior. Hence, the behavior results actually from a sequence of choices. Then, two elements that impact on the behavior appear: first, the way the plan is built and, second, the way this particular action is chosen among others. Therefore the behavior building, and by way of consequence the work of a behavior designer, can be split in two distinct sequential parts. We call these parts **reasoning** and **individuality**.

The **reasoning** corresponds to the part that consists in computing the possible solutions for solving goals according to agent's abilities and knowledge. This task is usually assigned to a *planner*. Our proposition is that this reasoning be the same for all the agents. As said above, simply because agents have different abilities and knowledge, will lead this part to engender different behaviors.

The **individuality** part follows the reasoning. Its task is to select an action from the possible solutions computed by the reasoning. It is assigned to an *action selection mechanism* which selects the action that is going to be actually performed by the agent. The proposition we developed in the following is an action selection mechanism based on motivations, the same mechanism is used for every agent. The evaluation of the motivations being different from one agent to another, they are the way to distinguish agent's behaviors and to add personality traits to the agent's behavior. In our proposition, two agents with the same abilities and knowledge would produce the same reasoning but can nevertheless behave differently thanks to this action selection mechanism which enables to express their individualities.

The reasonning and individuality procedural mechanisms are then the same for every agent. The abilities and motivations can be declared for every agent. They can differ from one agent to the other achieving the separation of procedural and declarative aspects.

Responsibility of each part is clear. Actually, the reasoning part is dedicated to the solving of agent's goals, and individuality part is dedicated to the problem of choosing the next action to be performed.

III. Reasoning

We have presented the reasoning as the part that build plans. The aim of these plans is to solve the agent's goals, and in this purpose the agent uses its abilities.

The agent's abilities are the actions the agent disposes of to interact with its environment. Performing an action leads the state of the environment to change. An action can then classically be seen as an operator that enables to go from one (input) state to another. A planning problem is then to find a sequence of actions that lead from an initial state to a final one. Among the actions (or abilities) we want to emphasize those appliable in the current environment (i.e. for which the environment corresponds to an admissible input state):

Definition 2 (Runnable action): For a given environment $\mathcal{E}$, a **runnable action** is an action such that $\mathcal{E}$ is in a valid input state. The agent can perform such an action immediately.

For an agent, a planning problem is then to find a sequence of actions that leads the environment from its current state to another in which the goal is achieved. Usually, to achieve a given goal in a given environment, there exist several ways, which correspond to disjunctions in a goal resolution. To distinguish these possibilities, we use the term of **alternative**, which corresponds for one goal to a possible plan without disjunction. We use the following definition:

Definition 3 (Alternative): An **alternative** is a triple $(g, \alpha, (a_i)_{i\in[1,n]})$ where

- g is a goal,
- α is a runnable action
- $\forall i \in [1, n], a_i$ is an action and $(\alpha, a_1, \ldots, a_n)$ is a sequence of actions to achieve g.

The three parts of an alternative can be identified as follows.The runnable action α is what the agent **can do**, now. The sequence of actions (a_i) is what the agent **expects to do**: the actions the agent plans to do to achieve its goal once

α would have been performed. The goal g is what the agent **wants**: it is a state the agent should have reached once he has performed all the planned actions.

For a given goal and a runnable action, there are several related alternatives. These alternatives are differentiated by what the **agent expects to do**, i.e. they are differentiated by how the agent plans to solve the goal according to the current state.

Agents must behave rationally. It is classically accepted that an agent "*is rational if it does the "right thing" given what it knows*". As said before, the reasoning part processor is dedicated to a planner:

Definition 4 (Planner): Given an agent σ, its set of goals $\mathcal{G}$, its set of abilities $\mathcal{A}$ and its memory (knowledge base) $\mathcal{KB}$, a **planner** is a process that, according to information in $\mathcal{KB}$, computes all the alternatives $\mathcal{A}lt_\sigma$ solving the goals in $\mathcal{G}$ using actions in $\mathcal{A}$.

Then, if $\mathcal{A}gents$ denotes the set of agents and $\mathcal{A}lt$ the set of all possible alternatives, a planner can be seen as the application[1]:

$$\begin{array}{rccc} Planner: & \mathcal{A}gents & \rightarrow & \mathcal{P}(\mathcal{A}lt) \\ & \sigma = (\mathcal{G}, \mathcal{A}, \mathcal{KB}) & \mapsto & \mathcal{A}lt_\sigma \end{array}$$

It is not the purpose of this paper to make new proposition for the planner. A lot of works have been done in this area.Any solution can be chosen as soon as it fulfills Definition 4: to provide the set of alternatives. In the following of the paper, we will more focus on the action selection mechanism.

IV. Individuality: Motivation-Based ASM

While the reasoning part tries to solve the goals, the role of the individuality part is to select the next action to be performed by the agent. This task is dedicated to an action selection mechanism (ASM for short). Our proposition consists in a motivation-based ASM.

An ASM is responsible for selecting an action among all the runnable actions identified by the reasoning engine. In this purpose, an ASM assigns to each runnable action a numeric value and select action with the greatest value.

Definition 5 (Action Selection Mechanism): Let $\mathcal{A}$ be the set of actions, and ϕ a function:

$$\begin{array}{rccc} \phi: & \mathcal{A} & \rightarrow & \mathbb{R} \\ & a & \mapsto & value \end{array}$$

then the **action selection mechanism** is defined as the application:

$$\begin{array}{rccc} ASM: & \mathcal{P}(\mathcal{A}) & \rightarrow & \mathcal{A} \\ & \mathcal{A}^R & \mapsto & \arg\max_{a \in \mathcal{A}^R}(\phi(a)) \end{array}$$

Let $\mathcal{E}$ be an environment, $\mathcal{A}$ be the set of available actions in $\mathcal{E}$, σ be an agent situated in $\mathcal{E}$, and $\mathcal{A}^R(\subset \mathcal{A})$ be the set of runnable actions for σ in $\mathcal{E}$ at a given time t, then the ASM provides σ with its next action α to be run in $\mathcal{E}$ at t.

[1]where, if $\mathcal{S}$ is a set, $\mathcal{P}(\mathcal{S})$ denotes the set of the parts of $\mathcal{S}$: $\mathcal{P}(\mathcal{S}) = \{s \mid s \subseteq \mathcal{S}\}$.

Our contribution consists in the definition of the function ϕ of Definition 5. We claim that the behavior is influenced by tendencies computed from motivations. Then, we propose to define ϕ as the function that combines these influences, each motivation being expressed through a function that we called **evaluator**. Thus, an evaluator provides a rating for each runnable action, expressing influence of the motivation on this runnable action. We need to be more specific about the last assertion. The ASM selects the best action, which is the action with the highest evaluation resulting from the combination of all the ratings it receives from evaluators. However, it should not be discarded that these runnable actions belongs to one, or more, alternatives provided by the planner. The other actions in the alternative correspond to "what the agent expects to do". These actions contain the prediction (not the anticipation) on actions that the agent will run. Therefore, not forgetting that we want some rationality in the agent's behavior, it is necessary to take these other actions into consideration. Indeed, the *action* selection mechanism should rather be considered as an *alternative* selection mechanism[2], since, with its choice of a runnable action, α, it determines the next goal and, in this purpose, the next plan[3] considered by the agents. Plausible observed behaviors require that the agent does not begin a plan that will later be discarded nor switch continuously from one plan to another. To achieve this, an evaluator, or at least some of the evaluators, must consider not only α but the whole alternative to compute the value assigned to α.

As an illustration, let us consider two alternatives (or plans) that share the same goal but have different runnable actions, $p_1 = (g, \alpha_1, (a_i^1)_{i\in[1,n_1]})$ and $p_2 = (g, \alpha_2, (a_i^2)_{i\in[1,n_2]})$. Let us assume α_1 is chosen by the ASM, it means that the agent engages in p_1 and runs the a_i^1. It should not happen that, after a few resolution steps of p_1, the ASM discards an action a_k^1, and then turns back toward p_2, while this refusal could have been foreseen from the beginning, because of some motivation's inhibition due to this action for example. In such a situation, the ASM has to promote immediately α_2, and then p_2. This can be achieved only by considering the whole alternative while evaluating the runnable action. This corresponds to consider a plan in the medium term and not only in the short-term that results in selecting α_1 in spite of a_k^1.

According to the previous remarks, we define a motivation and its evaluator as a function that evaluates alternatives:

Definition 6 (Motivation, Evaluator): Let $\mathcal{A}lt$ be the set of all alternatives. A **motivation** is defined by a function γ, called **evaluator**:

$$\begin{array}{rccc} \gamma: & \mathcal{A}lt & \rightarrow & \mathbb{R} \\ & alt = (g, \alpha, (a_i)_{i\in[1,p]}) & \mapsto & r \end{array}$$

Then, a motivation based ASM, for action/alternative selection mechanism, is defined as:

[2]luckily, it makes the acronym *ASM* still work.

[3]Of course, there can be several goals, and then plans, for the same runnable actions but this does not change the comment.

Definition 7 (Motivation based ASM): Let $\mathcal{A}$ be the set of actions, $\mathcal{A}lt$ be the set of all possible alternatives. A **motivation based action selection mechanism** is a pair $(Comb, \Gamma)$ where $Comb$ is a function from $\mathbb{R}^n$ to $\mathbb{R}$ and $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ is a set of motivations.

Then, function ϕ can be defined by:

$$\begin{array}{rccl} \phi: & \mathcal{A}lt & \rightarrow & \mathbb{R} \\ & alt & \mapsto & Comb(\gamma_1(alt), \cdots, \gamma_n(alt)) \end{array}$$

and a motivation based ASM is defined as the application:

$$\begin{array}{rccl} ASM: & \mathcal{P}(\mathcal{A}lt) & \rightarrow & \mathcal{A} \\ & \mathcal{A}lt_i & \mapsto & \alpha \end{array}$$

where α is the runnable action of an alternative alt such that, $alt = (g, \alpha, (a_i)_{[1,p]}) = \arg\max_{alt \in \mathcal{A}^R}(\phi(alt))$. $Comb$ is called the combination function.

Thus, let σ be an agent, if $Planner(\sigma) = \mathcal{A}lt_\sigma$ then $ASM(\mathcal{A}lt_\sigma) = \alpha$ is the next action to be performed by σ.

Our point of view is to use the same action selection mechanism for every agent, i.e., all the agents use the same function $Comb$ and set of motivations Γ. It implies that the issue results now in the definition of the motivations, theirs evaluators, and the $Comb$ function as well. It represents the first effective task of the behavior designer, but it can be done just once for every agents, possibly for different simulations or games.

Indeed, what must change from one agent to another is how the agent, actually its behavior, is influenced by each motivation. We assert they are the differences in these influences that constitute the differences between the agent's behaviors. Then, from one agent to another, besides the abilities as stated before, the change is how the agent appreciates the motivations. To enable this, evaluators, γ_i, are defined as parametrized functions whose parameter values are defined by each agent. These parameters influence how a given motivation impacts on the agent behavior. For each γ_i, we denote π_{γ_i} this set of parameters. Since they use the same planner (reasoning) and action selection mechanism, it is a distinct assignment of these parameters that makes two agents with the same abilities in the same environment behave differently. So, we define the individuality profile of an agent as follows:

Definition 8 (Individuality profile): Let σ be an agent, $(Comb, \Gamma)$ be a motivation based ASM, Π be the set $\cup_{i \in [1,|\Gamma|]} \pi_{\gamma_i}$, the **individuality profile** of σ is the set of values assigned by σ to elements in Π.

We can then consider some individuality factory function that applies from the set of agents to $\mathbb{R}^{|\Pi|}$ and maps its individuality to an agent. The definition of this function is the second task of the behavior designer.

V. Designing a motivation-based ASM

The behavior designer has first to define the motivation based ASM. It can be done once for all and this task can be divided in three stages. First, the identification of all the desired and relevant motivations, i.e. the set Γ, must be done. Second, the combination function $Comb$ to be used must be chosen. Third, for each motivation in Γ an evaluator γ_i must be defined. First and second steps are independent from each other. In third step, the chosen combination function must be considered.

The first step is rather conceptual. It is the problem of determining, and possibly naming, the general motivations that should drive the agent behavior. A good principle to follow is to separate motivations such that the role of each can be easily and clearly expressed. A good expression of the motivation role eases the evaluator definition at third step.

In the second step the combination function is chosen. Each motivation gives its "advice" on runnable actions. The role of the combining function is to aggregate these evaluations in order to obtain a general evaluation. The combination function plays a role similar to the "arbitrator" in DAMN [5] that "votes" for the action to be performed. There are several methods for combining motivations, such as the social choice inclusion [6] to represent the "opinion" of motivations. Numerous mathematical functions can be used as combining function candidates, each having properties that may influence the action selection. However, the combination function has to respect the two criteria. First, it must enable motivations to express neutrality, repulsion, attraction and inhibition, as presented in section II. Second, it must enable the adding and removal of motivations while keeping the consistency of aggregation with respect to the individuality. The respect of this second criteria implies that it is possible to incrementally build the ASM. In this case, a posteriori required new motivations can be added, without questioning what has already be done, even and especially concerning the agent individuality level. This is an important property that increases the ASM robustness. Let us note that the choice of the combination function constraints and influences the evaluators domain and range. These must be precise at this step as well.

The main difficulty lies probably in the third step where the chosen motivations must be translated into a function. However, one must not forget several points that allows, hopefully, to make this difficulty a bit more relative. First, the evaluation will depend on various observers and unanimity is impossible. We have already said how the behavior naming is subjective, and then the respect of the initial (first step) naming remains unsure. Second, there will be several concurrent motivations, therefore the notion of "tendency" is important, the evaluator has to make the action selection to tip in the wished tendency, and not to define precisely the chosen action. Because of the large variety of possible motivations, it has no meaning to sketch some generic evaluator. But, as it has been discussed in section IV, the designer has to be watchful to build motivations that take into account the three parts of the alternatives: the goals, the runnable action and the other-actions sequence. Every motivation does not need to tackle the three parts, even if some can, but in any case, the three parts must be considered at the moment or the other. During this step, the designer determines the set of parameters π_{γ_i} for each evaluator γ_i. Merged, the parameters in these sets have to be instanciated to establish the agent individuality.

Following these steps we have designed a specific ASM for character's agent acting in simulated environments, like in role-playing video games. The simulation's designer provides the agents with abilities. The agents have goals and use their abilities to solve them, and thus they act in the environment and interact with other entities of the simulation. The ASM we designed considers five *personal* motivations:

- **The goal influence** takes into account the different goals and their priorities. An agent has several concurrent goals to manage. The higher the prioritary, the more favored a goal is.
- **The agent preferences** motivation enables the agent to express some personality traits. Since the behavior expresses through the performed actions, one can consider that to a personality trait correspond some actions preferred to others. The idea is to promote or to penalize an alternative depending on the actions it contains. A preference value, that corresponds to the parameters of $\pi_{\gamma_{pref}}$, is assigned to each action that the agent can perform. This value expresses how much the agent likes or dislikes to perform this interaction, according to its personality. Since some actions are favored, the agent leans to execute them and then to express the associated trait.
- **The achievement in time** favors alternatives whose achievement requires the less time. The durations of the actions in the alternative are considered.
- **Momentum** purpose is to prevent too many changes, or oscillations, in agent's behavior. It favors the actions in the same alternative than the last previous selected best action. It leads the agent to be inclined to achieve a goal once it has begun to treat a corresponding alternative.
- **The multi-goal revaluation** promotes runnable actions that contribute to several goals.

and two *environmental* motivations:

- **The opportunism** promotes a runnable actions if it involves a target that is close enough. This motivation is detailed below.
- **The achievement in space** favors alternatives that requires less move steps to be achieve. The move steps required to achieve all the actions in the alternative are considered.

The limited number of pages forbids us to detail here all these motivations. So, we are going to focus on the design of only one of them: the *opportunism*. The purpose here is to illustrate the approach that the behavior designer could adopt.

At step 1, we decide that the behavior of our agents must be influenced by a ("conceptual") motivation that we name *opportunism*. An opportunist is defined as "*a person who adapts his/her actions, responses, etc., to take advantage of opportunities, circumstances, etc.*". We decide that this motivation expresses through the fact that the agent will favor actions whose target is close to it. Then an observer should notice that while moving near some environment elements, the agent seems to be attracted by an element, whether it is of some interest for the agent. For example an agent that comes to move close to an apple could be inclined to take this apple to eat it, if it feels hungry, even if eating was not the prioritary goal at that moment. Thus, opportunism could lead the agent to be temporarily diverted from a "main" goal and gives a feeling of reactive behavior.

At step 2, we define $Comb$. For the need of this example, let us assume that the chosen function is such that a value of 0 means inhibition, a value in $]0, 1[$ means repulsion, 1 means neutrality and a value greater than 1 means attraction and the greater the value is the stronger the attraction is.

At step 3, the evaluator function must be defined. The opportunism can be seen as an attraction from the simulation elements. This represents a short term and very contextual tendency and then only the runnable action of an alternative is considered. We decide that the element influence is restricted within some range and has no effect outside, i.e. it is neutral. Inside, the closer is the target element, the higher the value will be. Then, let $alt = (g, \alpha, (a_i)_{i\in[1,n]})$ be an alternative, we define the opportunism evaluator γ_{opp}:

$$\gamma_{opp}(alt) = \begin{cases} 1 \text{ if } \theta_{opp} \leq 1 \\ 2 \text{ if } dist(target(\alpha)) = 0 \\ \max\left(1, 1 + \log_{\theta_{opp}}\left(\frac{\theta_{opp}}{dist(target(\alpha))}\right)\right) \text{ otherwise} \end{cases}$$

where *target* is a function that returns the target of the action α, *dist* is a function that gives the distance in number of move steps between the actor agent and an element and θ_{opp} is the influence range.

We introduce a $\log$ to limit the strength of a close target attraction, here $\gamma_{opp}(\alpha) \in [1, 2]$.$\theta_{opp}$ is the single parameter for this evaluator, i.e. $\pi_{\gamma_{opp}} = \{\theta_{opp}\}$. This value can be changed from one agent to another to build the individuality profiles. The higher it is, the more opportunist an agent will be perceived.

Once it has defined the ASM that will be shared by all the agents, the second task of the designer is to define agent individuality profiles. Again the variety in the possible evaluators prohibits to propose an universal methodology. However, to prevent the task to become too tedious, the designer can define several instances of individuality profiles, i.e. different sets of instanciation values for the same subsets of evaluator parameters. Then, (s)he can build the various agent individualities by merging such different instances. Such instances can be considered as kind of individuality profile prototypes. The aim is to avoid to have to repeatedly define parameters one by one for every agent. The instances ranges do not necessarily map to the π_{γ_i} and can recover several of these sets or correspond to subsets of them.

We propose to label (name) these instances in order to add some semantics to the values through a choice of labels that express personality traits. The designer can then pick the various instance to build the agent individuality profile. Actually, the designer can consider to characterize and to label intervals of parameter values, and it is not necessary that these

intervals cover all the possible value range. For a given agent the assigned value can then be chosen in this interval, the diversity of obtained profiles is then increased.

By example, considering the above opportunism evaluator, we chose, arbitrarily, to define the following instance prototypes for θ_{opp}:

- $\theta_{opp} = 1$ (or $\theta_{opp} \in [1, 1]$) corresponds to *not opportunist*
- $\theta_{opp} \in [2, 3]$ corresponds to *weakly opportunist*
- $\theta_{opp} \in [5, 8]$ corresponds to *averagely opportunist*
- $\theta_{opp} \in [12, 20]$ corresponds to *very opportunist*

Then, depending of the personality trait the designer wants the agent to express, he can choose between *not*, *weakly*, *averagely* or *very opportunist*. To be more adaptive, the interval bounds can be fixed to a value relative to some agent properties. In the θ_{opp} case, this could be the agent perception range (the vision radius for example). Thus, *weakly opportunist* could correspond to θ_{opp} lesser than the quarter of this range, *averagely* from the quarter to the half and *very* from the half to the full perception range, or any other choice made by the designer.

Shifting from numerical to symbolic, this labelling should make the individuality profile design more accessible and more understandable. It is then possible to consider two designer levels. The first corresponds to some kinds of designer administrator: its tasks correspond to what we have described so far, i.e. to design the ASM following the three presented steps and to produce the individuality profiles prototypes. This role requires some expertise in programming and ASM understanding. The second level is the individuality profile designer who picks from proposed prototypes to build individuality profiles for each agent. Basing upon the labels, a simulation designer, without a priori programming knowledge, should be able to achieve this task.

The other motivations and their evaluators are defined in a similar way. As mentionned previously, altogether these motivations considers the three parts of the alternatives. Thus, if *opportunism* considers only the runnable action part, the *agent preferences* motivation, for instance, considers the whole alternative.

We led several experiments that validates this ASM, and present here one of them as an illustration. In this simulation, we consider an NPC-agent named c with a limited vision radius and an *energy* property. The value of this energy decreases at each time step. c is able to perform the interactions: *move*, *take*, *eat*, *break*, *unlock*, *open* and *explore*. We decide to define an agent who could seem to be "brutal". This is achieved through the coice of a preference value higher for the *break* than for the *unlock* action.

c is situated in the environment, shown in Figure 1. Two apples and one ax are in the environment, eating an apple re-energizes the agent and the ax can be used to break the door, which can undergo several interactions: *break*, *open* and *unlock* interactions. The door in the right side is only closed whereas the door in the left side is closed and locked.

Goals are then given to c. In our case, its first goal g_1 is

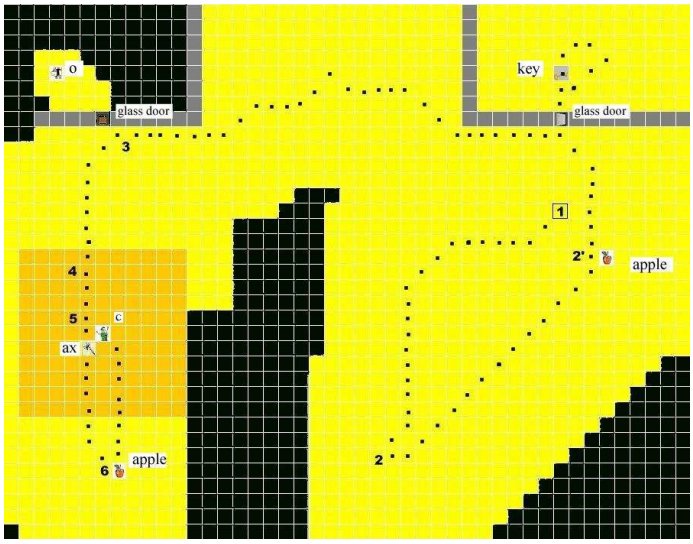

Fig. 1. The environment and the agent route after some simulation steps.

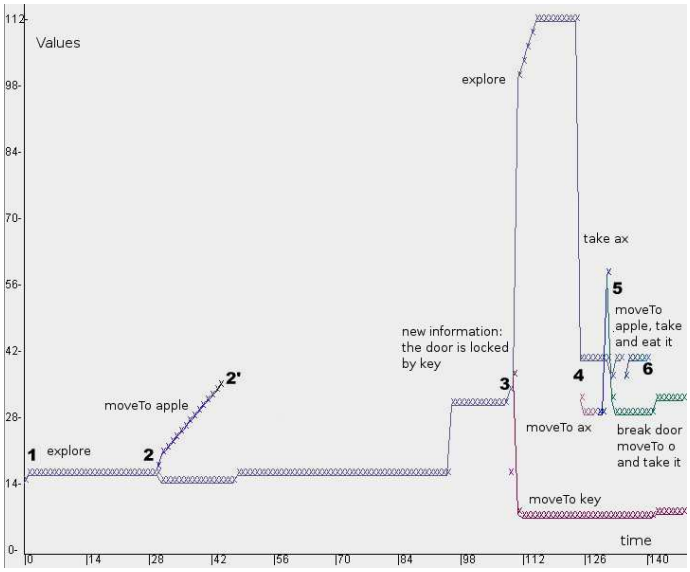

Fig. 2. ASM values according to time, each curve corresponds to an alternative.

to *take the object* o in the upper left room closed by a glass door whose the key is in the top right room. The priority of g_1 is a constant function. Its second goal, g_2 is to maintain its *energy level above a given value* v. The influence of g_2 is the function depending on the energy level of the agent.

c has no a priori knowledge about its environment and has to discover (explore) it. The unknown (never seen) places are in black in Figure 1 where we can also guess the agent radius vision. For each step, c performs the action selected by the ASM in order to solve its goals. The agent route during the simulation is shown by the black dots. Figure 2 shows the different values delivered by the ASM to alternatives. Initially, the only runnable action is *explore*, thus it is selected.

At the beginning, c is located at point **1**, it only perceives the apple at the right side. As the agent starts with an energy

level above v, the only active goal is g_1 (i.e. g_2 is satisfied, the priority is 0). As c did not know its environment, he must explore it to find the target o. During its exploration, c loses energy. When reaching the point **2**, its energy falls below v, the goal g_2 is then no longer satisfied, the goal influence promotes the alternative related to g_2 according to the current value of v. Moreover the *action to move toward the apple* (to eat it) becomes runnable. As shown in Figure 2, the value of this action is the highest, then c chooses to perform this action. As its energy continues to decrease during move, goal priority is growing hence the evaluation of the alternative too. In point **2'**, c eats the apple, it receives energy and the goal g_2 becomes inactive again. c resumes in exploring the environment to find o. We can notice the small up and down for the curve corresponding to g_1 alternative: this is due to momentum, with respectively loss, when g_2 alternative becomes selected, and gain, when this alternative is selected again.

Reaching the point **3**, c has perceived o. Having tried to open the door, c learns it is locked. The plan proposes two options for clearing the door: *unlock* it or *break* it. Two runnable actions appear in the graph corresponding to two alternatives: first, *move* toward the key (which has been seen previously); second, *explore* to find an object to break the door. Insofar break was favored, the agent prefers to break rather than to unlock the door. That's why the alternative with the action *explore* is favored over the alternative with the action *move* that corresponds to the lowest curve starting at **3**. The other curve is especially high, because coincidentally, at the same time, the goal g_2 has become active again. Now, *explore* action occurs in both alternatives, thus, the *multi-goal revaluation* promotes it.

By exploring, c goes "down" and perceives an ax in point **4**. Then in the corresponding alternative the runnable action is no more *explore*, but *take* the ax. This corresponds to the new curve in the middle. *Explore* loses favor from the *multi-goal revaluation* evaluator, which explains the drop in its curve. However, it is still the highest priority action, because of g_2's priority. At point **5**, the *opportunism* about the ax makes the action *take the ax* to become the highest priority. The peak at the point **5** is due to this **opportunism**, in the same time the decrease of action *explore* is due to the loss of *momentum*. Once the ax taken, the influence of *opportunism* disappears and action *explore* becomes again that with the highest assessment. Later, c found an apple and eats it the point **6**. *To break* the door becomes the action selected by the ASM. c moves to the door, breaks it and takes the object o (not shown in Figure 1).

This simulation illustrates the "competition" between the motivations and how the ASM works. It shows that the proposed motivation-based ASM enables to produce complex behaviors that considers several factors and can express personality traits.

VI. Related works

Concerning the primary application field targeted by this work, like the authors in [7], we notice that, even if things seem to evolve slowly, in video games "*most characters AI still uses only very basic techniques that are decades old*". Hard-coded scripts remains of regular use. Moreover it is often difficult to obtain precise information on how behaviors are implemented into commercial games.

Among interesting works, Jeff Orkin's can be mentioned. In [8], he presents the build of NPC's A.I. in a First Person Shooter video game named F.E.A.R. Its purpose is to delegate some of the workload of designer to a planning system. The A.I. is composed by a Finite State Machines (FSM) and the use of A^*. Let us notice that Orkin shows the difficulty and the complexity of using FSM approaches although they are commonly used in game design. In this proposal, the logic determining transitions from one state to another is moved from procedural FSM into a declarative planning system which took inspiration from STRIPS. One of the benefits presented by the author is that an agent with different actions behaves differently from another one in the exact same level (environment) and with the same goals. The author adds a cost per action, that permits him to use an A^* to select the best action performed. In this A^*, edges are actions, nodes are world states and cost metric is the cost per action. The calculation of the cost per action depends on many factors of the situations in F.E.A.R. and this task, which corresponds to the individual part of behavior was not facilitated. In our approach, we separate declarative and procedural knowledge not only in the reasoning part of the behavior but also in the individual part of behavior. Thus factors (i.e. motivations) can be added or removed easily.

Another intersesting approach is the one presented in [7]. The authors propose a subsumption-based graphical user interface intended to AI novices for building interactive characters. Their initial interface was based on Final State Machine approaches, but most users thought them very complex and unintuitive and so FSM and HFSM (Hierarchical FSM) approaches were abandoned. Although subsumption architecture is easily handled by novices in AI, it is a static hierarchical structure. Their studies show that, with BehaviorShop, the behavior building seems to become easier, but it does not seem to be easy to define reusable behaviors.

Action selection mechanisms are often associated in literature to animats. Animats are artificial robot (or computer simulated) animals which behave in the real (or in a simulated physical) world in a realistic way. In his PhD thesis [9], T. Tyrrell compares several (concrete or only models of) action selection mechanisms. He also implemented models of action selection, tested and compared them, in order to extract constraints that must be considered in assessing a "good" action selection mechanism. In our proposition, we focus on virtual agents which have no physical actuators or sensors. The implemented ASM of section V fulfills the Tyrrell's criterias that can be used for virtual agent and do not involve the sensori-motor part of animats.

B. Schmidt proposes in PECS (*Physical conditions Emotional state Cognitive capabilities Social status*) a model to represent human behaviors [10]. Behaviors are determined by physical, emotional, cognitive and social factors, and their

interactions. This model is based on internal variables like temperature, hunger, tiredness and other hysteresis variables that drive the agent to perform a specific action. All of theses influences are called motives which compete each other and the strongest one determines the agent's behavior. The author defines four kinds of motives which are distinguished by their constructs and origins (drives, emotional intensity, will power and social desire). His model works in four steps. It is possible using a motivation-based ASM to reproduce it.

VII. Conclusion

The character's behavior design is a complex task. A part of this complexity can not be reduced. In particular since in simulations (like video games) a large variety of rich behaviors is expected. However, the need to provide numerous individual character behaviors should not lead to repeat the work for each agent. Moreover, the impact of changes in the simulation design must be limited: the adding of new abilities to agentss, of new elements in the game, etc. must not question previous behavior design work. Simultaneously, it should be possible to incrementally adapt the behaviors by taking into consideration new factors that must influence these behaviors.

Applying the separation of declarative and procedural aspects, the approach proposed in this article aims to bring an answer to this problem. A principle is to provide every agent with the same procedural mechanisms: the planner and the action selection mechanism. The core of our proposition is based on our motivation-based action selection mechanism that enables behaviors' individualization. The modularity provided by the motivations enables a large variety of behaviors for which the designer has to instanciate parameters. But (s)he can rely on previously defined parameter sets and combine them.

References

[1] J. E. Laird and M. van Lent, "Human-level AI's Killer Application: Interactive Computer Games," in *the 7th Nat. Conf. on Art. Intelligence*. AAAI, 2000, pp. 1171 – 1178.

[2] A. K. Seth, "Evolving action selection and selective attention without actions, attention, or selection," in *the 5th Int. Conf. on Simulation of Adaptive Behavior*. MIT Press, 1998, pp. 139–147.

[3] A. Burloud, *Principes d'une psychologie des tendances*. Librairie Félix Alcan, Paris (France), 1936.

[4] J. Ferber, *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.

[5] J. K. Rosenblatt, "DAMN: A distributed architecture for mobile navigation," in *Journal of Experimental and Theoretical Artificial Intelligence*. AAAI Press, 1997, pp. 339–360.

[6] K. Arrow, *Social choice and individual values*. New York: J. Wiley, 1951.

[7] F. W. P. Heckel, G. M. Youngbloo, , and D. H. Hale, "Behaviorshop: An intuitive interface for interactive character design," in *proceedings of the 5th AIIDE*, 2009.

[8] J. Orkin, "Three states and a plan : The a.i. of f.e.a.r." in *Proceedings of the Game Developers Conference*, 2006.

[9] T. Tyrrell, "Computational mechanisms for action selection," Ph.D. dissertation, University of Edinburgh, 1993.

[10] B. Schmidt, "Human factors in complex systems : The modelling of human behaviour," in *ECMS'2005*, pp. 5–14.