A semantic generation framework for enabling adaptive game worlds

Ricardo Lopes
r.lopes@tudelft.nl

Rafael Bidarra
r.bidarra@tudelft.nl

Computer Graphics Group
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

ABSTRACT

Adaptive games are expected to improve on the pre-scripted and rigid nature of traditional games. Current research uses player and experience modeling techniques to successfully predict some gameplay adjustments players desire. These are typically deployed to adapt AI behavior or to evolve content for simple game levels. In this paper we propose a generation framework aimed at creating personalized content for complex and immersive game worlds. This framework, currently under development, captures which content provided the context for a given personal gameplay experience. This model is then used to generate content for the next predicted experience, through retrieval and recombination of *semantic gameplay descriptions*, i.e. case-based mappings between content and player experience. Through its integration with existing player and experience modeling techniques, this framework aims at generating, in an emergent way, game worlds that better suit players. Dynamic game content, which responds to the player performance, has the ability to personalize player experience, potentially making games even more unpredictable and fun.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Modeling packages, Object hierarchies*

General Terms

Algorithms, Design

Keywords

adaptive game worlds, procedural content generation, semantics

1. INTRODUCTION

Typically, most computer games feature a pre-scripted gameplay. Their content, rules, narratives and environments are all created during development, as finished *static* elements. Later, *dynamic* players will interact with these, learning and adjusting themselves to one of the many gameplay experiences that were predicted and pre-designed. Static pre-scripted game components have become a standard solution so that games can remain robust, testable and controllable.

However, partly as a result of such rigidness, the expected gameplay might not be the most appropriate for pleasing a large number of potential players. This can be apparent, for example, in two statistical facts: the low average rate of players who buy and complete games (20 - 25%) [7] and the male-dominated nature of games. Static game content typically is one of the many explanations for such a lack of gameplay appeal. Even for those players who suit the game, problems can occur. For example, such pre-scripted nature can make game outcomes more easily anticipated, since all possible interactions can be explored. Even worse, if players can predict certain outcomes, their progress can be often achieved by repeatedly exploiting a successful strategy.

In an attempt to account for player individuality, games often include minor variations that depend on players profiling themselves. For example, by customizing the difficulty level, players are classifying themselves as one of the available pre-defined low-resolution stereotypes, *e.g.* beginners or experts. However, this discrete approach implies that such games might fail in appealing to players who do not know how to profile themselves or who do not identify themselves with any of the available classifications.

All issues above indicate that, by contrast, a more dynamic gameplay could simultaneously: (i) encourage a more flexible, personal and unpredictable player experience, and (ii) widen the appeal of games to a larger audience. Recently, several researchers [5, 2, 13] have proposed adaptive games as a solution to achieve this dynamic gameplay and both of these goals. The main idea is to cater the game experience to the individual user and, ultimately, adapt game components, in a dynamic fashion, to better suit players.

Although a recent field, significant contributions already allow to accurately model players' skills, preferences and styles in a variety of game genres. Moreover, research in experience modeling is already enabling assessment and prediction of gameplay experiences. This sort of technique is also successful in adapting game narratives, non-player characters (NPC) behavior, scenarios for serious games or evolving game content for simple 2D level structures.

In this field, we are especially interested in the adaptation of more complex and immersive game worlds. This focus is driven by three motivations: (i) adaptive game worlds can offer new unexplored possibilities for affecting player experience, (ii) the research coupling between adaptivity and game world generation, which seems natural to us, is mostly lacking, and (iii) this coupling can allow designers to better author adaptive games.

In this paper, we describe the conceptual scheme of a procedural generation framework for adaptive game worlds. Its implementation is currently underway and preliminary results can be considered

Short presentation, ACE'2011 - Lisbon, Portugal

very promising. Our aim is to generati game world content that can provide the context for personalized dynamic gameplay. Such contextualized content can be made independent of the game narrative, although compatible and coherent with it. Firstly, as will be discussed later (Sections 3 and 4), this solution applies naturally to games that either have simple narratives or are tightly bound to learning aims. Secondly, for more intricate plots, generation can be specified to occur within self-contained plot events, focusing on the way to achieve an outcome and not the (narrative) outcome itself.

In order to represent the personal gameplay value of game worlds, we use semantics, *i.e.* a declarative modeling approach that embeds the world and its objects with all information beyond their geometry [29]. Deploying gameplay semantics will not only allow us to steer procedural mechanisms, but also enable an interactive design of this type of adaptivity.

This paper is structured as follows: in Section 2 we briefly survey adaptivity in games and position our research. In Section 3 our semantic generation framework is described in detail. In Section 4, we discuss possible scenarios where we anticipate this framework will successfully apply. We finalize with our conclusions and future work in Section 5.

2. ADAPTIVITY IN GAMES

The definition of an adaptive game can be extracted from the early proposals from Charles [2], Magerko [13] and many others: adaptive games recognize and comprehend their players' interaction, and intelligently alter themselves to adapt to the in-game needs or goals of their players, improving their gameplay experience.

The idea of creating adaptive games is not new. Several attempts have been made in recent years, both in the industry and in academia. Some examples in commercial games are Remedy's *Max Payne* player-centred dynamic difficulty adjustment mechanism, Nintendo's *Mario Kart* rubber-band AI or Valve's *Left 4 Dead* adaptive procedural narrative.

Academic research typically tries to focus on more global and abstracted approaches, far from the *ad-hoc* nature of commercial games. The architectural principles that should be behind generic game adaptivity have already been discussed by several researchers [2, 13, 33]. We have outlined them in Fig. 1. In essence, game logs, recording the players' performance, are used to create models of players' skills, preferences or style. Given a game state, these models can also be used to assess and predict the players desired experience of the next game state. Depending on the approach, both player and experience models can be used, in conjunction or not, to steer an adaptation and generation engine. This engine adjusts or generates the appropriate game components to better suit the player, *i.e.* adapted to the models' data.

Adaptive games typically require this two-step methodology: player modeling and content generation. This means that a coupling between both of them is not only natural, but required for adaptivity to work. As stated in Section 1, we are investigating the architecture of a semantic-based generation engine of virtual worlds, for adaptive games. One of our goals is to enable a loose coupling to existing player and experience modeling methods, so that our generation framework can be integrated and applied in several domains. It is therefore highly relevant to survey not only adaptation and generation techniques, as done in our recent and more in-depth survey [12], but also what steers them, *i.e.* player and experience modeling, as well as what we propose to support them, *i.e.* semantics.

2.1 Related work on player modeling

With player modeling, gameplay information and metrics are processed to create knowledge about the behavior of the player.

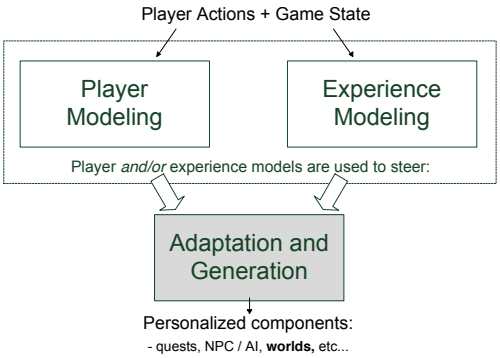

Figure 1: Overview of game adaptivity architectural principles: player and experience modeling steer adaptation and generation of personalized game components

Several techniques, typically used in adaptive games, have already been proposed to model some aspects of player behavior. One of them is supervised machine learning. Through analysis of a training data set, consisting of correctly modeled players, a classifier function is inferred by a learning algorithm (*e.g.* artificial neural networks, decision trees). This classifier function can then be used to model players from real game data sets. Machine learning has been used to model player skills in platform games [8] and shooting games [16], and preferences in strategy games [25].

Unsupervised machine learning has also been proposed in this domain. Player clustering, *i.e.* identifying and aggregating correlated gameplay data, has been applied to classify player styles and preferences [20]. Beyond machine-learning, other approaches have been proposed. With case-based player modeling, individual cases, typically built manually, encode a combination between player metrics and associated model features. During gameplay, player metrics are compared with the different cases and similar retrieved cases are used to create a player model. With vector-based player models, pre-created heuristic functions are used to calculate scalar values, accounting for model features, from game metrics. Case-based and vector-based approaches are able to model, respectively, player preferences in interactive narratives [23] and player skills in serious games [32].

From surveying the above related work on player modeling, we can conclude that, presently, models typically capture player skills (*e.g.* shooting proficiency), preferences (*e.g.* for items used, actions taken) or styles (*e.g.* explorer, achiever). A generation framework, which aims at seamless integration with player modeling techniques, should be aware of these types of model features.

2.2 Related work on experience modeling

Experience modeling techniques aim at evaluating and predicting player experience. Gameplay information and the game state are analyzed to determine how the game is or can be experienced. It is not uncommon or incorrect to consider experience modeling as a type of player modeling. We avoid this debate by clarifying that for us, as defined in Fig. 1, player modeling relates to player behavior and experience modeling relates to player experience.

Experience modeling was recently surveyed in [33], where three types of approaches were identified: subjective, objective and gameplay based. The difference between these is the type of player data they rely on to evaluate and characterize gameplay, respectively: (i) data expressed voluntarily by players (*e.g.* self reports), (ii) data obtained from alternatives modalities of player input (*e.g.* motion or eye tracking), and (iii) data obtained through normal player inter-

action, *i.e.* game metrics. The techniques described in the previous section are being applied in many of these approaches, with models linking player data and experience.

For this paper, it is more relevant to survey which types of player experience are modeled. The difficulty experienced by a player is one of them, being a straightforward way of accounting for player satisfaction. Several researchers [8, 26, 10] (and the commercial games above) model the difficulty perceived by players and predict the optimal challenge level. More complex experience models have been recently proposed. Pedersen *et al.* [18] built quantitative models that can predict player experience as being: fun, challenging, boring, frustrating, predictable or anxious. Other affective states like engagement have also been researched [1]. In serious games, player experience is oriented towards the gradual fulfillment of learning goals. As such, experience is typically modeled, implicitly, by monitoring and evaluating in-game learning goals [17].

The surveyed research allows us to identify the experience model features to consider in an adaptation and generation framework: challenge level, affective states (*e.g.* fun, boredom) and learning goals (*e.g.* treating victims).

2.3 Related work on adaptation and generation

Adaptation and generation methods are still less researched than modeling techniques. This happens because user modeling is a unified and broad research field. In contrast, the different game components able to be adapted can relate to many different research questions from different fields.

NPC behavior is the most common aspect being researched for adaptive games. Techniques like agent organizations [32], dynamic scripting [26] or case-based reasoning [6] are able to successfully adapt NPC behavior to better suit players (*e.g.* their skills). Narratives, quests and scenarios have also been made adaptive. Typically, these are dynamically generated by a central manager which analyzes player data and recombines predefined basic elements into complete quests [27], scenarios [14] or narratives, these last ones as surveyed in [21]. As for generation of game environments, some research has been done in simple and linear level structures. Procedural content generation has already been proposed for adapting racing tracks, using evolutionary algorithms [28] or platform games levels, using exhaustive search of generated content [22].

All surveyed approaches are strongly coupled to their integrated user models, and are dedicated to specific game types. Also, immersive 3D game worlds are still not being considered [12]. We can therefore conclude that there is plenty of room for researching adaptation and generation that: (i) is abstracted from modeling techniques and game types, and (ii) focus on complex and immersive game worlds.

2.4 Related work on virtual world semantics

Semantics in virtual worlds is a recent research field. Its motivations can be traced back to the early proposals by Deussen *et al.* [4] for an ecosystem simulation model to generate an area with vegetation. Its input data, *i.e.* terrain and plant properties, and its production rules, *i.e.* space, soil and sunlight competition, raised a discussion for the need of more complete and ubiquitous information on virtual worlds and objects.

In this direction, smart objects were proposed [9], containing information about the possible interactions that can be executed on them. Peters *et al.* [19] took the notion of smart objects further by creating objects with information about their functionality, how NPCs can interact with them, and where important features of the object are situated. Research in Virtual Reality (VR) has also been exploring semantic representations, specifically to apply in the design of virtual environments. Latoschik *et al.* [11] proposed *Semantic Entities*, an object modeling method where the actions and functions of virtual objects can be specified and applied. In the same direction, De Troyer *et al.* [3] introduced a conceptual modeling approach for creating VR worlds, where designers can specify high-level concepts on how complex objects are composed and moved.

Our own previous work on semantic modeling explores these ideas further, by considering, in an integrated manner, both geometric constraints/relationships and functional information (we leave its detailed discussion for Section 3). Additionally, our semantic representation has been applied not only during runtime (as Peters' smart objects), but also to procedural content generation and layout solving (thus reassuming Deussen's requirements). We feel that it is a natural step to further integrate this approach with semantic information about the gameplay value of game worlds, and match it with player data.

3. GENERATION FRAMEWORK

In the previous section, Fig. 1 outlined the architectural principles typically used to support adaptive games. We are specially interested in researching within this field from an adaptation and generation perspective. Our aim is to develop a generation framework able to create game worlds that can provide the *potential* and the *context* for personalized dynamic gameplay. We use the terms *potential* and *context* since we believe it is never up to the content alone to fully realize experiences. The player and the game engine are responsible for fulfilling that potential. The goal for this framework is then to maximize the appropriateness of the generated content, to enable the fulfillment of personalized experiences.

In this section we describe the conceptual scheme of our framework, whose implementation is underway. Fig. 2 schematically describes our proposal for this generation framework for adaptive game worlds. It focuses on generating adaptive game worlds, stemming from our goal of investigating new ways of potentially affecting gameplay. As seen in Fig. 2, the generation process can be integrated with player and experience modeling techniques, as the ones surveyed in the previous section. Our aim is that this framework can be seamlessly reused with distinct user modeling methods.

With our approach, semantics encoding the gameplay value of game worlds is deployed atop geometry. This semantic information can be compared, using mechanisms similar to case-based reasoning, with outputs from the integrated player and experience modeling methods. This comparison and retrieval allows us to predict what should be the content capable of providing the next desired player experience. Using semantics we can: (i) explore the link between adaptive game worlds and procedural content generation, and (ii) consequently, enable game designers to control the generation process and author adaptivity. In the next paragraphs we will further detail our semantic generation framework.

3.1 Semantic library

This generation framework builds on our previous work on semantics in game worlds [29]. We define semantics, in the context of game worlds, as all information about the world and its objects, beyond their geometry. This includes object properties, high-level attributes and functional information, as well as interrelationships among different objects.

Each object in our game worlds typically carries all its semantics. They belong to some class of a *semantic library* [30], a hierarchical class database, partly based on the WordNet [15] ontology. This library allows game designers to specify semantics, atop geometry,

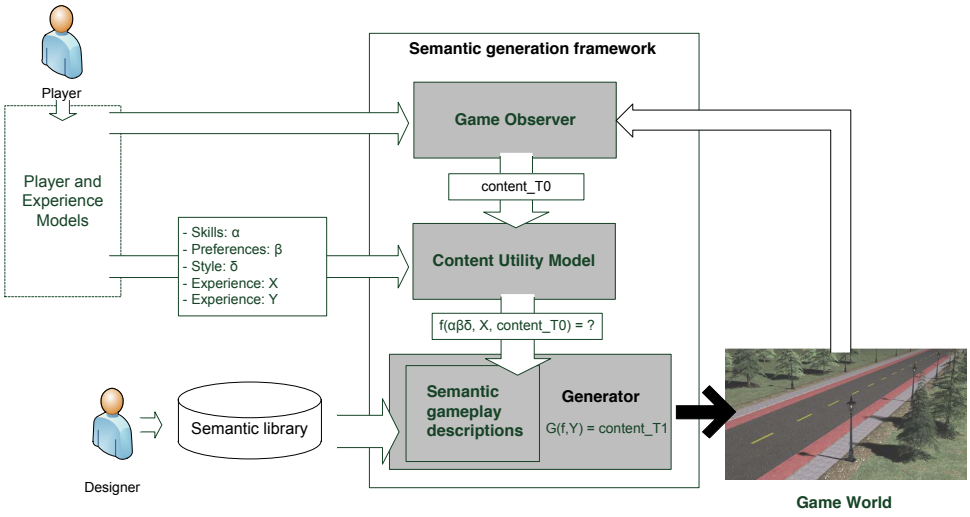

Figure 2: Scheme of our generation framework for adaptive game worlds: Semantic library, Semantic gameplay descriptions, Content utility model, Game observer and Generator

on game worlds. This semantics can be used to control and constrain algorithmic procedures that generate specific world content. This semantic level provides designers with a powerful front-end that generates and steers an underlying procedural level, while encapsulating the complexity of the latter. This approach has already been successfully deployed, *e.g.* in interior layout solving [30] and building generation [31], as shown in Fig. 3.

Designers use a library editor to specify semantics on each class. Two types of classes are present: entities and abstractions. Entities refer to what is possible to instantiate in a game world (*e.g.* physical objects, materials, substances). Abstractions are either characteristics of entities (*e.g.* attributes, states, services) or of sets of entities (*e.g.* groups, scenes). Associating entities and abstractions allows us to specify for each game object a set of attributes, including functional information, as well as geometric relationships with objects of other classes.

Building upon this approach, a new layer of gameplay semantics is required, so that the semantic library classes can include knowledge on how they can affect player experience. For this, we designed a set of gameplay abstractions: *player skill, player preference, player style, experience, game genre* and *actor*. Each gameplay abstraction can be associated with classes of the semantic library. The basic idea is to characterize semantic classes in terms of their gameplay value to players. Typically, a designer would create gameplay semantics by adding this type of abstractions on each class. To do so, they will use the the following scheme, available for each class, where allowed values for Z, W, Y, X, V and U are already encoded in the semantic library:

Class A:
can provide gameplay **experience(s)** Z
to players with: **skill(s)** W, **preference(s)** Y, **style(s)** X
when owned by **actor(s)** V
in **game genre(s)** U

We propose these gameplay abstractions since they naturally derive from the conclusions described in Section 2, *i.e.* they match the main features in player and experience modeling. Also, as exemplified in Section 4, they can be further parameterized using scalar values. Each class in the semantic library can be altered to include several associations as the above. For example, a baseball bat can provide different experiences in sport or fighting games, when owned by the player or NPC.

The nature of the semantic library allows high flexibility when creating gameplay semantics. Since this can be defined for each class, both entities and abstractions can be considered. As such, gameplay semantics can be defined for a variety of different aspects, as, for example, physical objects, groups of entities or even generic attributes or states. Furthermore, the semantic library allows multiple levels of specification, where property values can be constrained or instantiated. This allows us to define and restrain gameplay semantics to different levels of class property values (*e.g.* to all sizes of an object vs. to a specific size).

3.2 Semantic gameplay descriptions

As we described in the previous paragraphs, and as illustrated in Fig. 2, game designers create gameplay semantics atop a game world semantic library. Our generation framework accesses this information through *semantic gameplay descriptions*, containers listing links between semantic classes (*i.e.* game world content) and player experience. The aim of semantic gameplay descriptions is to compare them with the observed behavior and experience of a particular player, at stages that require content generation. An example of semantic gameplay descriptions is shown in Fig 4.

Semantic gameplay descriptions are inspired on case-based reasoning. They are meant to encode valid combinations between content and the gameplay experiences they can provide, for a given set of preconditions. In our case, these preconditions relate to player features (*e.g.* $\alpha\delta$ in Fig. 4) and game genres (*e.g.* G1 in Fig. 4). Semantic gameplay descriptions emerge from the semantic library (see Fig. 2) at design stage, for each new adaptive game. The gameplay semantics of each library class is analyzed and semantic descriptions are created and assembled accordingly. The logic is simple: analyze library classes, identify and aggregate preconditions, create respective gameplay descriptions and add the applicable classes and respective experience to them. The accuracy of semantic gameplay descriptions is therefore dependent on the knowledge specified by

Figure 3: Interior layout example, generated using the semantic library [31]

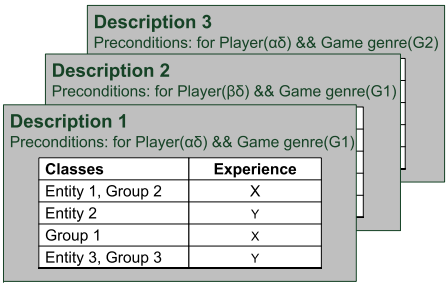

Figure 4: Example of semantic gameplay descriptions, with notation consistent with Fig. 2

game designers.

Any class from the semantic library can potentially be used in a gameplay description (*e.g.* objects, scenes, groups) not only with different levels of instantiation (*i.e.* with attribute values already specified, constrained or not) but also linked to different ways of creating its geometry (*e.g.* geometric models, procedures).

3.3 Content utility model

To be able to compare semantic gameplay descriptions with player behavior and experience, we need to not only model these, but also the content that enabled and provided the two aspects.

For this, we introduce *content utility modeling*, supported by two steps: (i) integration with player and/or experience modeling and (ii) monitoring the relevant content which enabled player behavior/experience. We plan to incorporate external methods, like the ones surveyed in Section 2, to create and maintain player/experience models, rather than creating our own. As discussed before, these models typically include, player behavior features and experience features, not only describing which experience was observed but also which should be the next. Their integration is achieved by a model translator, a component which converts the format of the player/experience models' output to the format of our semantic player/experience features. If complex experience modeling is not needed for integration (for example, for difficulty level adjustments), the model translator can be built to encode the behavior (even if static) of the model left out.

Content utility models are responsible for relating player/experience features with the relevant content that enabled them. To do so, we propose a virtual game observer that monitors and selects content appropriate to incorporate in the content utility model. It is critical that this observer only selects content which, with a high degree of confidence, contributed for the conclusions from the player/experience models. Otherwise, our new model would include incorrect data.

As seen in Fig. 2, the game observer accomplishes this by monitoring both the game world and the integrated models. For this to be possible, we devised a set of criteria for selecting relevant content, which spans both domains. Regarding the game world, the observation criteria account for content that was interacted by: players, NPCs and the game engine. The nature of these interactions accounts for the use of objects and the occupation of spaces, in the same time interval as what was processed by the integrated models. Regarding the integrated models, a single criteria accounts for the content that is directly related with the game metrics they use. The idea is to observe which game metrics, measured by the model, reflect content interaction and include the content related to those metrics. For example, if a player modeling method monitors the number of walls broken by the player, the game observer should monitor and include broken walls in the content utility model. The game observer needs to be customized for every instance of a new game or a new integrated modeling technique, so that these criteria are totally fulfilled.

We are confident that these criteria, monitoring game world interaction and the player models, are enough to account and capture all relevant content which provided the observed gameplay.

3.4 Generator

The generator (see Fig. 2) is responsible for the creation of game worlds, by applying all the information modeled in our framework. In an approach inspired by case-based reasoning, it tries to match the content utility model with gameplay descriptions. The plausible assumptions are that: (i) if gameplay descriptions include valid content-experience combinations, and (ii) if the content utility model associates content, player features, and experience, all measured for a given moment, and (iii) if a semantic gameplay description exists with that same association, then (iv) the remaining combinations in that description can also be considered valid and usable.

Under these assumptions, generation becomes a matter of retrieval of semantic gameplay descriptions. As illustrated in Fig. 2, if the content utility model includes *content_T0* which enabled experience *X*, for a player modeled as $\alpha\beta\delta$, then the generator needs to implement the following function *G*, where *Y* is the experience required for the next stage:

$$G(f(\{\alpha\beta\delta\}, X, content\_T0), Y) = content\_T1 \qquad (1)$$

Function *f* needs to retrieve and return a set of semantic gameplay descriptions where its arguments are valid. Afterwards, function *G* examines the retrieved semantic gameplay descriptions and selects the content (*content_T1*) which provides the required experience *Y*. The frequency and moments on which the generator, *G(x)*, should act in a game world will need to be decided when implementing the integration of our framework in a game.

If only a single description is retrieved by *f(x)*, the generator has a simple task in creating the relevant content. Content is selected and instantiated by *G(x)* according to its semantics, using the associated geometric models or generation procedures. The generator will include a procedural mechanism, based on semantic layout solving [30], responsible for combining the selected instances into game world sections.

If several descriptions are retrieved by *f(x)*, then the generator has a more emergent behavior. In this case it will recombine several

instances of content declared in independent descriptions. This way, the content to generate emerges from distinct descriptions, maximizing the potential for a richer and more complex variability.

The variability in this emergent recombination mechanism is possible in two ways, both implemented in *G(x)*: *precondition selection* and *content selection*. For the former, we consider as valid all gameplay descriptions with preconditions which are a subset of the player modeled features. For example, in Fig. 2, the player was modeled as $\alpha\beta\delta$ and, in Fig. 4 there are descriptions for players modeled as $\alpha\delta$ and $\beta\delta$. Function *f(x)* will consider these as retrievable and selectable in *G(x)*. To solve possible conflicts between content of the retrieved descriptions, they will be ordered by degree of similarity with the player modeled features.

As for content selection, increasing the number of retrieved gameplay descriptions can lead to redundant or conflicted content. So, solving these situations differently can lead to different generated worlds. Besides the degree of similarity described above, we expect to use and compare semantics to identify and remove redundant or conflicting content, also introducing randomness in this process. Even though the details of such a mechanism are still being researched, we are confident in reusing some conflict-solving techniques for semantic game worlds, already present in our previous work in SketchaWorld [24], an interactive semantics-based procedural virtual world editor.

Finally, a remark on "starvation" situations where function *f(x)* does not retrieve any semantic gameplay description for a given step: even though we do not envision such a use for our generation framework, easy solutions can be found, *e.g.* using predefined static content, as done with non-adaptive games, in these steps.

4. APPLICATION SCENARIOS

In the previous section we described our proposal for a semantic generation framework for adaptive game worlds. We are currently working on its implementation, namely on the semantic library integration and gameplay descriptions. In this paper, we therefore aim at presenting and discussing the framework's conceptual scheme, and not yet its experimental evaluation. Instead, in this section, we will describe two simulated scenarios of the application of our framework, in order to help us evidence and discuss its advantages. Each scenario will focus on one game world section where generation is required, using what was modeled (player, experience and content) in the previous world section. Please refer to Fig. 2 for the framework modules mentioned throughout this section.

4.1 Scenario 1

The first scenario occurs in a First Person Shooter (FPS) game, where generation is required to occur between one room and the next one. Player and experience modeling are deployed to capture style and affective experience (using, for example, [20] and [18]).

Two different players have been modeled as (i) explorer, *i.e.* likes to explore the environment, and (ii) achiever, *i.e.* likes collecting items and leveling up his abilities through balanced gameplay. Both were modeled as being bored and needing a 50% increase in excitement. The content utility model registers the following content, for both players: previous room had one division, four empty boxes were opened, three crates were used as hideouts for NPC and a time-bomb was activated by the player.

For the explorer player, a single gameplay description is retrieved stating that rooms with one division, zero hidden chambers and time bombs increase boredom. This gameplay description also declares that hidden chambers in rooms increase excitement in 10%. From this, the generator decides to create the next room with five hidden chambers.

For the achiever player, a single gameplay description is retrieved where empty item boxes and NPC-owned crates increase boredom. This gameplay description also states that leveling up items that increase player abilities increase excitement in 25%. So the generator instantiates two leveling up items in the next room.

4.1.1 Discussion

In the scenario above, we can highlight the flexibility of the semantic library in allowing designers to specify how content can influence player experience. Boredom for the explorer is characterized by simple one-division rooms and by time bombs. This way, gameplay semantics can be used to capture that simpler rooms leave no space to explore and time bombs, if activated, can leave no time to calmly explore. The same authoring expressive power is shown for the achiever, where empty item boxes and NPC-owned crates increase boredom. This expresses the designer's knowledge on the fact that achievers like to progress in the game, by having multiple opportunities of collecting items, in a balanced way, where advantages by NPC are disliked.

The potential of our generator is also demonstrated in this scenario. In the end, two players with different styles were given two different rooms, with content more appropriated for each one. This shows the value in considering game world content in adaptive games. Also, in this scenario's case, the main gameplay was not hindered. The type of generated content was specified to not relate with the main goals or the mission of the game; it affects the player experience in that specific moment and room. So, although somewhat different, the game still remains the same for both players. This scenario also highlights the ability of our generator to function in a mixed mode: the room could actually be the same geometric space with some static content, and only some key objects are generated.

4.2 Scenario 2

The second scenario occurs in a driving simulator, a serious game where a player drives around a city and executes what an instructor suggests. Generation occurs on each street. Player modeling is integrated to capture the skills of the player, and experience is modeled to encourage, in an transparent way, the development of the captured skills (as, for example, in [32]).

For a player, skills are modeled as: 0.4 (out of 1) proficiency in parking sideways and 0.6 in clutch balancing. In this case, the modeled experiences are redundant and directly mapped from skills, *i.e.* low and medium proficiency in, respectively, parking sideways and clutch balancing (here defined as features *a* and *b*). Consequently, the learning goals to be encouraged next are thus to improve on these skills, using different learning levels, with an accessible level on parking sideways (*c*) and a challenging level on clutch balancing (*d*), both due to the measured skill levels. Besides these player and experience features, the content utility model includes: a one lane steep road *(e)*, misplaced parked cars on the side of the road *(f)* and traffic lights placed on a steep road *(g)*.

Two gameplay descriptions are retrieved, for the two types of modeled player skills. Tables 1 and 2 describe the content of these descriptions. We signaled both the classes and the features which match what was captured in the content utility model, *i.e.* arguments *X*, *content_T0* and *Y* of the *G(x)* function, as explained in Section 3.

Since both descriptions are retrieved, their entries are combined, generating a road with two lanes (*z*), with cars parked inside parking spaces (*y*), traffic lights with longer waiting periods (*w*) and with steepness angle 45 (*x*). The dilemma between a leveled or a steep road (*z* and *x*) would be solved by comparing skill proficiency. The parking sideways skill is closer to be matched (0.4 from the model to 0.5 from the description) than the clutch balancing skill (0.6

Table 1: For players with proficiency $\leq$ 0.5 in parking sideways

Semantic class	Experience
e) Road(lanes=1)	*a)* low proficiency: parking sideways
f) Car(parked=misplaced)	*a)* low proficiency: parking sideways
z) Road(lanes$\geq$2 , steepness=0)	*c)* accessible level: parking sideways
y) Car(parked=parking space)	*c)* accessible level: parking sideways

Table 2: For players with proficiency $\geq$ 0.5 in clutch balancing

Semantic class	Experience
e) Road(steepness=30)	*b)* medium proficiency: clutch balancing
g) Stoplight(location=*e*,wait=40)	*b)* medium proficiency: clutch balancing
x) Road(steepness=45)	*d)* challenging level: clutch balancing
w) Stoplight(location=*e*,wait=60)	*d)* challenging level: clutch balancing

from the model to 1 from the description), and could therefore be considered less important to be encouraged.

4.2.1 Discussion

This scenario highlights the authoring flexibility of our framework, showing its applicability in a different context: skill-based learning. Here, gameplay semantics can still be created to evidence the need to respond to different players by using different content. For example, for parking sideways, more street lanes lead to more space and thus less pressure, leveled streets mean more visibility and cars inside parking spaces lead to more space to maneuver. In other words, using the semantic library, designers can specify that dissimilar players can benefit differently from different content.

Scenario 2 also highlights the emergent behavior of the generator. The final scene is created through the combination of two retrieved gameplay descriptions. This can increase the variability in the generated content, in an unique way. In this scenario, this is made possible by considering two player experience requirements simultaneously, a usual behavior in present adaptive games. The conflict resolution in this example is based on a similarity degree between the models and the descriptions, together with a criterion to focus on the skill needing more improvement. For now, we foresee that mechanisms like these should be specified in the generator. However, we are aware that future evaluation of our framework might as well recommend giving designers control over this. This scenario also highlights the ability of our generator to act in a fully procedural mode, where all content is generated.

Although not directly mentioned in these scenarios, we should also discuss performance and burden on designers. This framework is envisioned to perform on-line, while the game is running. Considering that each semantic class should be already instantiated as of the import of gameplay descriptions (*i.e.* at the design stage), the remaining bottlenecks are likely the retrieval of descriptions and the layout solving by the generator, both at run-time. We are confident that, with our experience, new indexing and procedural techniques can help to achieve satisfactory generation efficiency.

As for the process of creating gameplay semantics, it needs to avoid burdening designers with overwhelming manual effort. The semantic library already provides mechanisms to facilitate the creation of semantics, including control on class inheritance and automatic consistency checks. Our aim is to extend these with new propagation mechanisms, specific to the gameplay semantic layer.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the conceptual scheme of our semantic generation framework for adaptive games. We have discussed its main components, highlighting the role that semantic modeling can play in adaptive game worlds. Through this framework and, specifically, by means of gameplay semantics, game worlds will be generated to match integrated player and experience models, *i.e.* adapting to the players' goals and needs. Semantics about personal gameplay value, associated with game world geometry, will steer a procedurally-based generator in creating a more suitable context for personalized dynamic gameplay.

Even though the implementation of this framework is underway, and its evaluation still has to take place, its novelty and advantages can be easily anticipated, and are therefore highlighted here.

First, this framework will allow us to include 3D immersive game worlds among those game components targeted by adaptivity. Achieving this, in turn, will likely trigger investigation on new, unexplored ways of affecting gameplay.

Second, regarding the process of creating adaptive games, the framework will allow content generation methods to be loosely coupled with player and experience modeling.

Third, the framework brings about the first inclusion of gameplay information in the area of semantic modeling. Although it is here being used for the procedural generation of game worlds, in the future, this new semantic scheme can become valuable, for example, for runtime interaction with game objects.

Finally, we can expect that deploying more and richer semantics will enable game designers to much better author adaptivity.

As for future work, we aim to continue with the implementation of this framework proposal, in the short term. As discussed before, our main challenges ahead relate with the design, creation and implementation of conflict-solving methods for our generator component. Also, we already identified the player and experience modeling techniques that we will need to integrate with our framework in prototype adaptive games. The implementation of both will begin in the near future.

As for the long term, we already identified valid future research directions. One of them is the automatic creation of semantic gameplay descriptions. With our framework, playtesting a game would allow to use content utility models to create semantic gameplay descriptions, independent from the semantic library. The basic idea would be to use our game observer as a mass data collection method to model content with player behavior and experience, automatically creating semantic gameplay descriptions. These would be applicable to similar players as the ones observed, in future game sessions.

Relying both on manual (*i.e.* designer-based) and automatic (*i.e.* through statistic observation) ways of creating gameplay semantics, rich immersive game worlds can become adaptive. When so, they will have the potential to provide a personalized dynamic gameplay, making games even more unpredictable, effective and fun.

6. ACKNOWLEDGMENTS

This work was supported by the Portuguese Foundation for Science and Technology under grant SFRH/BD/62463/2009.

7. REFERENCES

[1] G. Chanel, C. Rebetez, M. Bétrancourt, and T. Pun. Boredom, engagement and anxiety as indicators for adaptation to difficulty in games. In *Proceedings of the 12th International Conference on Entertainment and Media in the Ubiquitous Era*, pages 13–17, New York, NY, USA, 2008. ACM.

[2] D. Charles, A. Kerr, M. McNeill, M. McAlister, M. Black, J. Kucklich, A. Moore, and K. Stringer. Player-centred game design: Player modelling and adaptive digital games. In

Proceedings of DiGRA 2005 Conference: Changing Views - Worlds in Play, pages 285–298, 2005.

[3] O. De Troyer, W. Bille, and F. Kleinermann. Defining the Semantics of Conceptual Modeling Concepts for 3D Complex Objects in Virtual Reality. In *Journal on Data Semantics XIV*, volume 5880 of *Lecture Notes in Computer Science*, pages 1–36. Springer Berlin / Heidelberg, 2009.

[4] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic Modeling and Rendering of Plant Ecosystems. In *SIGGRAPH '98: Proceedings of the* 25^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 275–286, New York, NY, USA, 1998. ACM.

[5] K. M. Gilleade and A. Dix. Using frustration in the design of adaptive videogames. In *ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pages 228–232. ACM, 2004.

[6] T. Hartley and Q. Mehdi. Online action adaptation in interactive computer games. *Computers in Entertainment*, 7(2):28:1–28:31, 2009.

[7] IGN US. GDC: 72 percent of players finished Heavy Rain. http://uk.ps3.ign.com/articles/115/1153279p1.html (as of March 2011), 2011.

[8] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin. Polymorph: dynamic difficulty adjustment through level generation. In *PCGames '10: Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 11:1–11:4. ACM, 2010.

[9] M. Kallmann and D. Thalmann. Modeling Objects for Interaction Tasks. In *Proceedings of the Eurographics Workshop on Animation and Simulation*, pages 73–86, 1998.

[10] S. Kazmi and I. Palmer. Action recognition for support of adaptive gameplay: A case study of a first person shooter. *International Journal of Computer Games Technology*, 2010.

[11] M. E. Latoschik, P. Biermann, and I. Wachsmuth. Knowledge in the loop: Semantics representation for multimodal simulative environments. In *Proceedings of the 5th International Symposium on Smart Graphics 2005*, pages 25–39, 2005.

[12] R. Lopes and R. Bidarra. Adaptivity challenges in games and simulations: a survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(2):85 –99, june 2011.

[13] B. Magerko. Adaptation in Digital Games. *IEEE Computer magazine*, 41(6):87–89, June 2008.

[14] B. Magerko, B. Stensrud, and L. Holt. Bringing the schoolhouse inside the box - A tool for engaging, individualized training. In *Proceedings of the 25th Army Science Conference*. ASC, November 2006.

[15] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38:39–41, 1995.

[16] O. Missura and T. Gärtner. Player Modeling for Intelligent Difficulty Adjustment. In *Proceedings of the ECML–09 Workshop From Local Patterns to Global Models (LeGo–09)*, Bled, Slovenia, 2009.

[17] J. Niehaus and M. O. Riedl. Scenario adaptation: An approach to customizing computer-based training games and simulations. In *Proceedings of the AIED 2009 Workshop on Intelligent Educational Games*, pages 89–98, 2009.

[18] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67, 2010.

[19] C. Peters, S. Dobbyn, B. MacNamee, and C. O'Sullivan. Smart objects for attentive agents. In *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2003.

[20] D. Ramirez-Cano and S. Colton. Player classification using a meta-clustering approach. In *Proceedings of the 3rd Annual International Conference Computer Games, Multimedia & Allied Technology*, pages 297–304, 2010.

[21] D. L. Roberts and C. L. Isbell. A Survey and Qualitative Analysis of Recent Advances in Drama Management. *International Transactions on Systems Science and Applications*, 4(2):61–75, 2008.

[22] N. Shaker, J. Togelius, and G. N. Yannakakis. Towards Automatic Personalized Content Generation for Platform Games. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 63–68, October 2010.

[23] M. Sharma, S. Ontañón, C. Strong, M. Mehta, and A. Ram. Towards player preference modeling for drama management in interactive stories. In *Proceedings of the Twentieth International FLAIRS Conference (FLAIRS07)*, pages 571–576, 2007.

[24] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra. A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics*, 35(2):352 – 363, 2011.

[25] P. Spronck and F. den Teuling. Player Modelling in Civilization IV. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 180–185, 2010.

[26] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma. Adaptive game AI with dynamic scripting. *Machine Learning*, 63:217–248, June 2006.

[27] A. Sullivan, M. Mateas, and N. Wardrip-Fruin. Rules of engagement: moving beyond combat-based quests. In *Proceedings of the Intelligent Narrative Technologies III Workshop*, pages 11:1–11:8, New York, NY, USA, 2010. ACM.

[28] J. Togelius, R. De Nardi, and S. Lucas. Towards automatic personalised content creation for racing games. In *IEEE Symposium on Computational Intelligence and Games, 2007. CIG 2007*, pages 252–259, April 2007.

[29] T. Tutenel, R. Bidarra, R. M. Smelik, and K. J. de Kraker. The role of semantics in games and simulations. *Computers in Entertainment*, 6(4):1–35, 2008.

[30] T. Tutenel, R. Bidarra, R. M. Smelik, and K. J. de Kraker. Using semantics to improve the design of game worlds. In *Proceedings of the Fifth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 100–105, 2009.

[31] T. Tutenel, R. M. Smelik, R. Lopes, K. J. de Kraker, and R. Bidarra. Generating Consistent Buildings: a Semantic Approach for Integrating Procedural Techniques. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3), 2011.

[32] J. Westra, F. Dignum, and V. Dignum. Keeping the trainee on track. In *IEEE Conference on Computational Intelligence and Games, 2010. CIG 2010.*, pages 450–457. IEEE, August 2010.

[33] G. N. Yannakakis and J. Togelius. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, 99, 2011.