Authoring adaptive game world generation

Ricardo Lopes, Elmar Eisemann, and Rafael Bidarra

***Abstract*—Current research on adaptive games has mainly focused on adjusting difficulty in a variety of ways, for example, by providing some control over adaptive game world generation. These methods, however, are mostly ad-hoc and require quite some technical skills. To the best of our knowledge, so far there has been no adaptive method that is truly generic and explicitly designed to actively include game designers in the content creation loop. In this article, we introduce a generic method that enables designers to author adaptivity of game world generation, in a very expressive and specific fashion. Our approach uses *adaptation rules* which build atop gameplay semantics in order to steer the on-line generation of game content. Designers create these rules by associating *skill profiles*, describing skill proficiency, with *content descriptions*, detailing the desired properties of specific game world content. This game content is then generated on-line using a rule matching and retrieval approach. We performed user studies with both designers and players, and concluded that adaptation rules provide game designers with a rich expressive range to effectively convey specific adaptive gameplay experiences to players.**

***Index Terms*—Procedural content generation, adaptive games, gameplay semantics.**

I. Introduction

Adaptive games are steadily becoming a focus of interest. There is significant academic and even commercial investment into games that dynamically adjust their content or mechanics to better fit individual player-dependent needs, preferences or goals. However, most work has been focusing on the development of new methods for modeling the behavior of players or automatically generating game content or mechanics in an adaptive fashion [1].

In this work, we are motivated to empower game designers to *author adaptivity*. In particular, our goal is to harness and integrate designers' specific knowledge into adaptive content generation methods. We believe their rich design knowledge has a strong and unique role to play, when authoring more dynamic games. This can lead to a new game design paradigm: designing multiple personalized experiences by coordinating the use of content generators [2]. Game designers would then use co-creation tools to create not standardized game worlds but rather sets of instructions which would steer in-game content generators, yielding personalized world experiences [3].

The main contribution in our work is a semantics-based method that enables designers to author adaptivity of game world generation, in an expressive and specialized fashion. In particular, we propose the use of *adaptation rules* to steer adaptive, fully on-line game world generation. Designers specify these adaptation rules as associations between player profiles and content descriptions. A player profile is basically a set of abstract concepts, representing distinct gameplay proficiencies (termed as player gameplay abstractions), and a content description is a specification of game world characteristics, expressed in terms of semantic game world entities as proposed by Kessing *et al.* [4]. In this context, we define semantics as *all information about the game world and its objects, beyond their geometry*. To support the use of adaptation rules, we implemented a new generation method in which semantic game world entities are retrieved for a given input of a player model.

To demonstrate and evaluate the use of adaptation rules, we implemented an authoring tool for adaptive game world generation around an existing game. For this case study, we performed both design tests and player tests, to assess to which extent our method can effectively be used to author adaptive game world generation.

Ricardo Lopes, Elmar Eisemann, and Rafael Bidarra* are with the Computer Graphics and Visualization Group, Delft University of Technology, 2628 CD Delft, The Netherlands (*corresponding author: +31 15 2784564, email: r.bidarra@tudelft.nl).

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

II. Related work

Our research goal is to enable game designers to author adaptivity. The key motivation for adaptive games is to improve individual player experience beyond the *one-size-fits-all* approach of most current games and their static, manually-designed content [1], [5]. By dynamically tailoring game content to respond to the players actions and choices, games can become more personal and therefore can appeal to a higher audience, become more replayable and less frustrating.

The cornerstone of achieving such goals lies in the *adaptive experience*. Imagine a Super Mario game player who is unable to defeat any enemy, *i.e.* unable to jump on their head. The adaptive experience will define how and why the game should react in order to improve the player experience. Should the game exclude all enemies? Or should it first focus on jump training and then gradually insert enemies? Or should it add more enemies to maximize practice through frustration? Typical adaptive games make one of these choices and hard code it in their adaptation logic. We believe the adaptive experience(s) should be defined much earlier, at design time and not at compile time. And we state that game designers are best equipped to author such adaptive experiences. Our goal is to enable them to both explore and reuse design choices in a more scalable way, *i.e.* with less dependency on programmers. We therefore aim for an approach where game designers can declare what the adaptation logic should produce, in an interactive fashion.

This designer-driven research stems from our previous work on declarative modeling [6]. This research enabled designers to declare what procedural content generation (PCG) algorithms should create, in a mixed-initiative approach. Using procedural sketching, designers can quickly layout at the level of large terrain features of a virtual world, while fine-grained details are automatically generated by PCG algorithms. Such approach to designer involvement was also researched by Liapis *et al.* [7] where the playability, alternative designs and other evaluations are studied by linking high-level map sketching with evolutionary algorithms which use the manual map sketch as a guideline that constrains what to automatically generate.

Our solution for enabling game designers to declare adaptive experiences was driven by case-based reasoning (CBR), where a knowledge base of cases (problem to solution mappings) is used to solve new problems through retrieval and adaptation of past solutions to similar problems [8]. Like CBR, we (i) use a retrieval step to match solutions (the content the designer deems appropriate) to problems (the current player needs), and (ii) re-use and adapt known solutions to similar problems (content appropriate for different but similar player needs). Unlike CBR, (i) the knowledge base is not meant to be a global representation of all possible cases, but only those landmarking the desired adaptive experience; (ii) cases and solutions are explicitly declared by game designers (thus not a direct formal collection of similar past experiences); and (iii) cases and solutions act as a set of constraints given to algorithms which, respectively, model player needs and generate content. The dynamic nature of such algorithms gives them a higher level of variability than that of the CBR approach.

Previous research on adaptive games has mainly focused on establishing player modeling techniques and/or PCG methods. Therefore, for most examples, the adaptive experience is a fixed initial assumption (*e.g.* dynamic difficulty adjustment). However, our goal of authoring adaptive experiences still links to this previous research since both sides of the experience, player needs and content, are taken into account through, respectively, player models and PCG algorithms.

Player models can classify players under predefined labels and preferences [9], [10], predict affective states of the player, based on neuro-evolutionary preference learning [11] or physiology monitoring [12], and classify player styles, using clustering techniques [13]. The majority of player modeling research results propose discrete and bounded representations of player characteristics, *e.g.* a discrete number of bounded classes of players [14], [5]. When players 'fall outside' those representations, they are usually 'projected' onto the closest bound. For research purposes, this observation has been typically deemed as a sufficiently good approximation. Accordingly, our research will also assume the same discrete and bounded nature of player models.

Research in adaptive games can support the generation of game worlds, using evolutionary algorithms for racing tracks [15] and recombination of annotated level segments [16], grammatical evolution [17] or gameplay-based grammars for platform games [18].

To the best of our knowledge, there has been no research on enabling game designers to combine player modeling and content generation to author adaptive games in an interactive fashion. Previously, the use of *gameplay semantics* as a suitable vocabulary to support the control over PCG methods and its integration with player modeling techniques has been proposed [19]. We believe that our approach, together with current research advancements in PCG and player modeling, already provides a mature basis to start incorporating designers as active agents in the adaptive game creation loop.

III. Authoring adaptive generation

In this section, we describe the two main elements of our approach to authoring of adaptive generation: designer creation of adaptation rules, and rule matching and retrieval. In summary, game designers declare which content should be generated for which player profiles (adaptation rules created in the tool of Fig. 1). A retrieval algorithm finds which rule(s) apply best for a player at a given gameplay moment and the content specified in the adaptation rules is retrieved and used for the generation process in the game world.

Throughout the remainder of this article, we will use the term skill profiles interchangeably with player profiles. We chose player skill models as the best way to exemplify our method since they are easy to understand and they are widely used in many games, including that in our case study. As discussed in Section VII, a player profile does not need to be described exclusively in terms of skills. Many other features (*e.g.* preferences, styles) can be modeled, as long as they can be captured in a scale of values.

A. Adaptation rules

Adaptation rules are responsible for encoding the knowledge of game designers, by associating each player profile with the game world characteristics the designers envision for them.

To construct an adaptation rule, designers create a skill profile description and associate it with some content description(s).

A skill profile is a sequence of player skill values, each one indicating a proficiency level, measured in a given skill scale. An example of a player skill could be the ability to jump over platforms in Super Mario, measured by the percentage of successful jumps. Player skill profiles can be very conveniently created and visualized on a radar chart, where each axis represents a specific skill. The shape of a skill profile is the polygon created by connecting all skill proficiency values on the radar chart axis. If a certain profile of in-game player behavior, measured along these axes, occurs inside that shape, we say the player *belongs* to that skill profile.

A content description is a set of quantitative characteristics indicating how specific game world elements should be created. Examples include the (number or size of) gaps to be jumped on a Super Mario level or even constraints over them (*e.g.* their order, proximity or placement).

As mentioned before, the main advantages of these adaptation rules lie in the *expressiveness* and *specificity* of their control over adaptive game world generation. Specificity stems from the use of individual player skills (captured in skill profiles), as well as from the wide range of world entities that can be used in a content description (any object that can be included in a game world). This range allows designers to create adaptation rules holding highly specialized, and therefore personalized, value. For example, again in Super Mario, an adaptation rule could

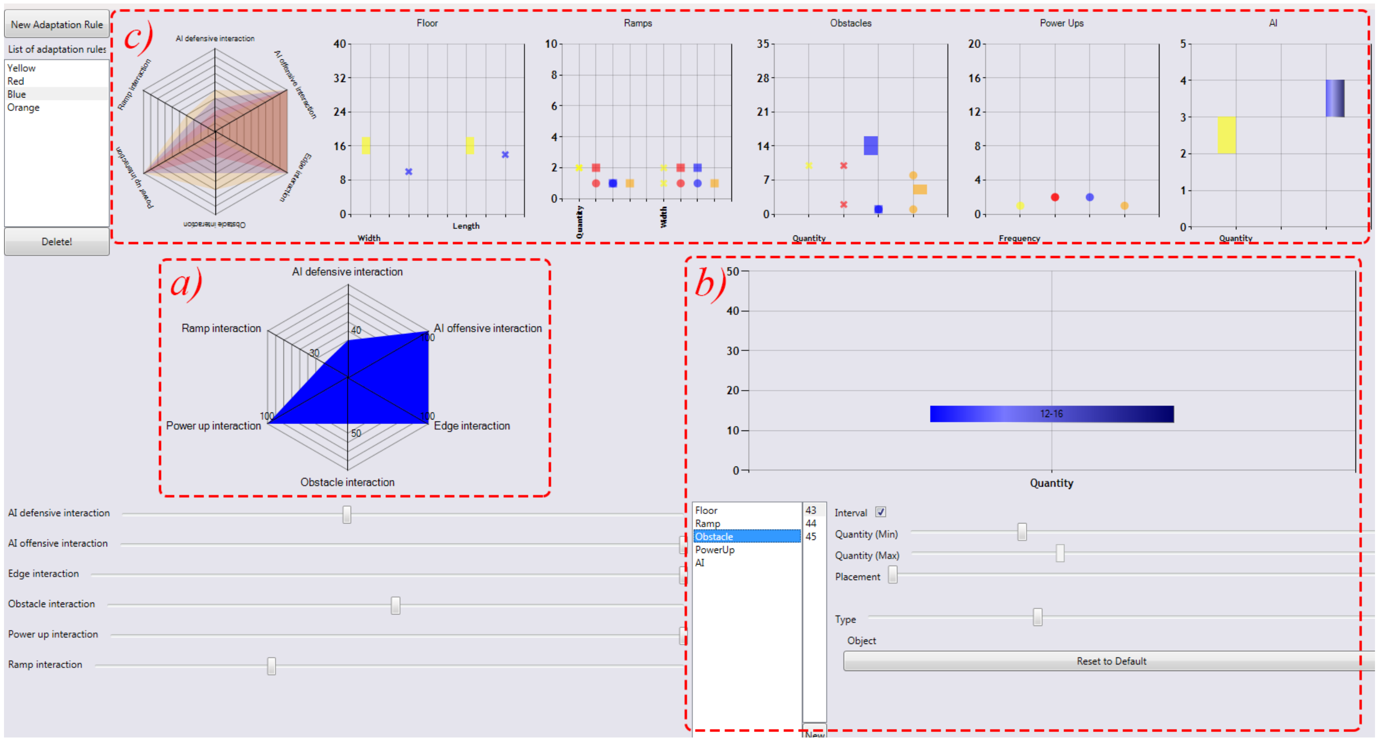


Fig. 1. Design tool for creating adaptation rules. Skill profiles are created through the specification of their shape in a radar chart *(a)*. Content descriptions are associated to skill profiles and they are created through the instantiation of semantic entities and their relevant attributes. Content descriptions are visualized through point and range charts representing the attribute values for each semantic entity *(b)*. The tool includes an overall visualization of all adaptation rules, by showing an overlapping graph for skill profiles and a sequential graph for content descriptions *(c)*.

create hardly accessible coins for a player who is excellent at collecting them.

Expressiveness relates not only to this open range of entities but also to the lack of assumptions in our model. Our skill profiles make no assumptions on player behavior or psychology, and allow designers to freely decide on the scale, for example, where and when do beginners or experts stay. We also make no assumptions on the value or characteristics that the world entities in a content description might *mean*. In short, content can be combined in any way to convey a desired experience, as intended by a designer.

Adaptation rules, skill profiles and content descriptions significantly extend and generalize our previous work on *semantic gameplay descriptions*, cases mapping content (semantic classes to use) and player experience (preconditions of application) [20]. Adaptation rules are represented internally as semantic gameplay descriptions. Skill profiles take the role of the preconditions of a semantic gameplay description, and content descriptions take the role of the semantics classes of a description. These semantic gameplay descriptions (now adaptation rules) are created directly by designers, and are not automatically derived from classes labeled with gameplay semantics, like in previous work [19], [21], [22].

Skill profiles are internally represented by sequences of player-skill gameplay abstractions [20]. This data structure is already integrated within our gameplay semantics model and allows for parameterizing skills with proficiency values.

Content descriptions are internally represented by sets of semantic entities and corresponding semantic attributes [20]. Each semantic entity can be associated to one or more geometric models, representing the geometry of that entity [4]. These models are helpful for the actual game world generation based on synthesis of game objects. However, that is not necessarily the only possibility: semantic entities can include (or be used as input for) algorithms to procedurally generate the corresponding geometry, as will be done in our case study, in Section IV.

Adaptation rules are created using a new design tool, expressly developed for this research; see Fig. 1. This design tool builds on top of Entika, our previous semantic editor, and it assumes the previous creation of semantics for valid entities, attributes and player-skill abstractions [4].

B. Rule matching and retrieval

Similarly to previously-proposed semantic gameplay descriptions [19], adaptation rules are solutions, outlined by designers, for a class of adaptation problems. For adaptation rules to work, two tasks have to be performed: (i) identify in which circumstances they apply, and (ii) trigger the creation of an appropriate set of game content. Identifying when rules apply, *i.e.* step (i) above, relies on the existence of a player model, responsible for capturing and classifying player behavior into a quantitative model. Significant changes in this model reflect significant changes in player behavior, possibly indicating the need for game adaptation.

A player model is also responsible for steering the selection of which adaptation rule(s) to apply, *i.e.* step (ii) above. To this extent, the player model, based on captured behavior, should be matched against the skill profiles in adaptation rules to identify (a) whether the player is (still) performing as a designer predicted, and (b) which content the designer specified as appropriate for that (new) player behavior. For our approach to work *off-the-shelf*, player models should capture the same features as the adaptation rules are configured to contain. They should therefore be (suitable to be) expressed in the same skill axes as the skill profiles in the adaptation rules. This allows an immediate matching process between the player model and adaptation rules.

Semantic gameplay descriptions used in previous work [19], were matched against the player model and retrieved for use in content generation. With a semantic gameplay description, designers had to be aware that it might apply to players who only partially matched its player preconditions [19]. Although not incorrect, that was not always desired, and the matching and retrieval algorithm for adaptation rules was created with this aspect in mind.

Therefore, we developed a new matching method for adaptation rules. The new method simplifies the creation process for designers while facilitating accuracy about their intent, since it excludes the need for knowledge on how the matching method or PCG algorithm work.

Listing 1. Matching and retrieval algorithm

```
$FindRules$(PlayerModel $<S_1, S_2, S_3, ..., S_N>$)
{
        for every AdaptationRule r
                Distance = $CalculateProfileDistance$(
                        SkillProfile(r), PlayerModel);
                Axis = $CalculateInclusion$(SkillProfile(r),
                        PlayerModel);

                if(Distance < MinD)
                        SelectedRule1 = r;
                        MinD = Distance;
                if(Axis <= MinA)
                        if(Distance < MinD2)
                                SelectedRule2 = r;
                                MinA = Axis;
                                MinD2 = Distance;

        if(SelectedRule1 == SelectedRule2)
                return SelectedRule1.Content;
        else
                return $ContentFrom$(SelectedRule1,
                        SelectedRule2);
}
```

The matching and retrieval algorithm aims at identifying which adaptation rule(s) best apply to a player at a given gameplay moment. To this extent, in order to have a reliable indication of the similarity between them, we compute the Euclidean distance between the skills of the player model and the skill profile in each rule. We investigated the use of weights or normalization per skill in the Euclidean distance formula but no significant differences in the adaptive mechanics were noted. So we concluded this distance was a reliable approximation of similarity.

The input for matching and retrieving adaptation rules is a tuple $< S_1, S_2, S_3, ..., S_N >$ originating from a player model, in which each element represents the proficiency value for a certain skill. For each input, we calculate the distance between the input tuple and each tuple in the skill profile of each adaptation rule. We also compute a measure of inclusion of the input tuple in the skill profile, by counting the number of axis along which the value of the input is not larger than the corresponding value of the skill profile (*CalculateInclusion*).

The importance of inclusion originates from the common practice among game designers, who often categorize players in large or small clusters (*e.g.* beginners, experts). We, therefore, considered inclusion as a fuzzy method of solving equidistance in a way that maps well to the most typical principles of game designers. One of the core principles of game flow is that a task must be sufficiently challenging to be enjoyable by a skilled enough player [23], [24]. This balance suggests us a commonly observed game design principle: difficulty is typically pulling the player to match it by improving in-game skills. This suggests game cycles of: reachable higher difficulty vs. lower player skills, flow, reachable higher difficulty vs. lower player skills, flow, and so on. Therefore, opting for inclusion, *i.e.* selecting the higher-bounded rule rather than the lower-bounded one, seems the best way to meet that typical designers' practice. With inclusion in mind, designers can use our approach to define such type of classes, choosing their magnitude (inclusion radius) and shapes. However, often the boundaries between such clusters will not be sharp, which is why we propose to independently use both criteria: distance and inclusion.

We identified two possible selection criteria to rank the candidate adaptation rules retrieved. Both look into the shape of their skill profile, selecting respectively: (i) for minimal distance, the adaptation rule which skill profile has minimum distance (along the skill axes) to the input player model tuple, and (ii) for inclusion, the adaptation rule which skill profile has the most number of axes with a value higher than that in the input tuple (and if multiple rules, the closest one). To increase variability and decrease predictability, whenever both rules are retrieved, the respective content is combined. Naturally, such combination also assumes that the designer is coherent and does not create adaptation rules which are radically distant from one another, let alone contradictory in any way. This assumption, made on our side, was validated by studying and playing traditional linear games, and it is supported by the observation that, typically, designers already think of game content (levels, narrative, difficulty, player power) in a consistently gradual manner, in this design space. For highly complex (*e.g.* GTA) or less linear (*e.g.* Minecraft) games, typically with less frequent adaptation instants, this assumption and combination mechanism might need to be revisited. Fig. 2 displays such a case, where two different rules are retrieved that match an input player model tuple.

Even when two adaptation rules are retrieved and selected, one for each criterion above, conflicting content descriptions might be issued (*e.g.* generate 10 coins *vs.* generate 20 coins). In these cases, the method *ContentFrom* examines the content descriptions by observing if the same semantic entity (say coins, in the above example) occurs in both. If that is the case, one of the semantic entities is randomly removed from the respective content description. This means that the resulting final content is synthesized from (i) merging the semantic entities in the content descriptions of both rules, and (ii) possibly with the removal of one type of semantic entity from one rule side (only if that entity is present in both rules.) In our experience, merging

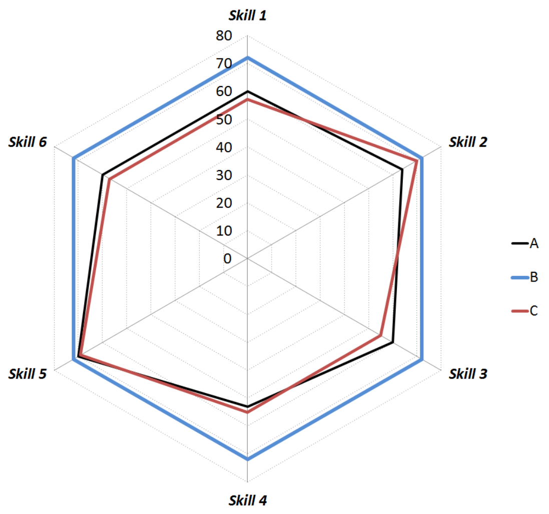


Fig. 2. *A* represents an example input player model tuple, and *B* and *C* represent (the skill profiles of) the two adaptation rules selected by the rule matching and retrieval algorithm. Rule *C* has the minimum distance to the input tuple. However, values for for Skill 1, 3, 5 and 6 are lower than in the input tuple *A*. The skill profile of rule *B* is such that simultaneously it: (i) completely includes input tuple *A*, and (ii) does so at a minimum distance of it. To avoid assumptions on what is best, both rules B and C are selected and combined.

and random removal of the same type of semantic entity yield content as intended by designers, because it occurs mostly between two adaptation rules with 'neighbor' skill profiles (due to the same assumption as the previous paragraph). If merging two rules into one creates more content requirements than in a typical rule (which may often happen), the PCG algorithm should be responsible for increasing the size of the virtual world to fit those requirements. Therefore, the resulting list of semantic entities, attributes and relationships represents a set of instructions and constraints (with possible geometric models attached) that will steer an in-game generator to create a game world, as exemplified in the next section.

IV. Case Study: Achtung Die Kurve 3D

To demonstrate and evaluate our approach, we created an adaptive modification of an existing game, and integrated adaptation rules into it. For this case study, we use a 3D graphics-based game, with full on-line PCG. Our aim was to explicitly demonstrate that gameplay semantics can be effectively applied to support and control on-line adaptive generation of whole 3D levels.

A. *Achtung Die Kurve 3D*

The game we chose to use in our case study is *Achtung Die Kurve 3D* [1], a third-person 3D version of the classic *Blockade* or *Achtung Die Kurve* games; see Fig. 3. In this game, players spawn at random places on a maze-like playing field composed of multiple floors connected by ramps. Players leave a solid tail behind their moving head as they progress through a level at a constant speed, using only left and right keys to control trajectory, until they eventually crash. The goal is to survive and be the last one standing. Crashes can occur when players collide with: their own tail, other players' tails, obstacles, floor edges, and ramp limits. As seen in Fig. 3, power-ups can spawn at random places and be collected by the players. Power-ups combine a target (self, only others, all) with an action type (increase speed, decrease speed, turn harder, turn softer, switch keys, no tail, thicker tail, thinner tail and clear all tails, all with a temporary effect) and are visually identified using a color and icon system.

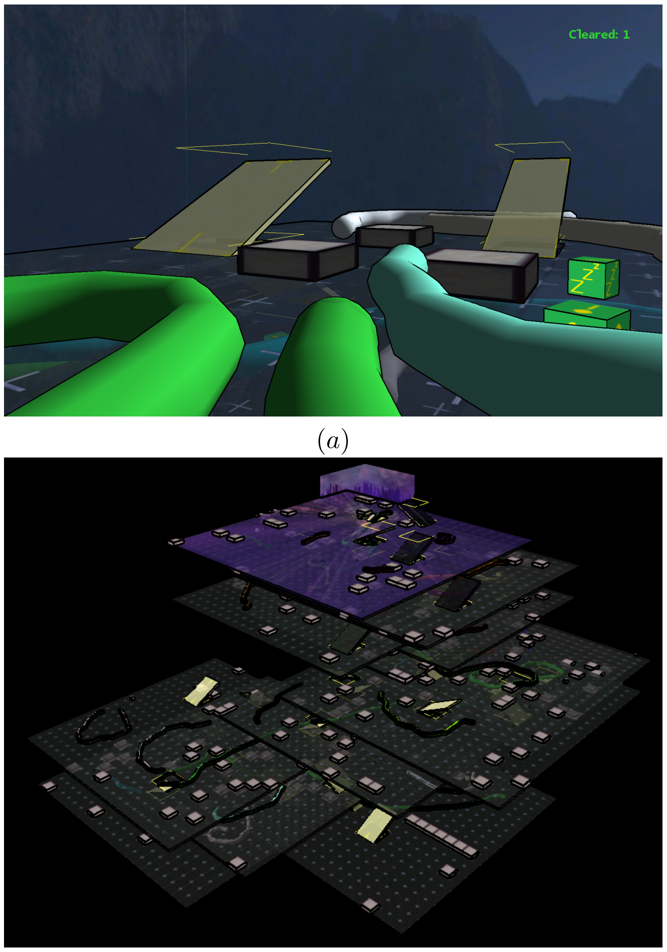


Fig. 3. *Achtung Die Kurve 3D* game. In *(a)*, notice the player's green tail, on the left, the white and blue AI enemies, power-ups on the right, and ramps and obstacles ahead of the player. In *(b)*, a tower of stacked floors was generated on-line, as the player progressed through the level

For this study, we implemented an adaptive modification of the original *Achtung Die Kurve 3D*. Our modification is a single player game, in which the human player tries to reach the highest floor possible. Each new floor is generated automatically as soon as the player enters a ramp, and floors can be generated indefinitely. Opposing players are AIs who are constrained to the floor they spawn in, only trying to kill the human player by confinement.

For our modification[2], we had to implement the following minor changes in the original game: (i) we changed the scoring

[1]http://graphics.tudelft.nl/ mkt4/2011/groep7/

[2]see http://rlop.es for a gameplay video of this modification of Achtung Die Kurve 3D

system to highest score in cleared floors, (ii) we extended the AI behavior to avoid entering ramps, and (iii) we added the feature of different ramp widths (only a single ramp width was available before). Naturally, our largest changes were in the level generation algorithm.

B. Floor generation

In the original *Achtung Die Kurve 3D* game, an off-line generation algorithm was responsible for creating a game world with equally fixed-sized floors, each represented by a $n \times m$ matrix. Each element in this matrix represented a unit-size cell in the 3D environment, indicating either empty cells, obstacles, ramps, power-ups or AI spawn points.

We drastically modified the level generation algorithm to support on-line PCG, including: (i) the level representation became a dynamic list of matrices (*i.e.* floors), with each new matrix, typically with its dimensions different from the previous one, being generated when a player enters a ramp; and (ii) each newly generated floor has only one entry, namely the ramp used by the player to enter it, which is randomly placed.

Finally, we changed the stochastic nature of level generation to a more controlled method. The generator can solve a variety of constraints requested for each new floor, typically describing its size, as well as number, type or placement of content like ramps, objects and AIs. In Sub-section IV-D, we discuss the format of these constraints in more detail.

C. Player model

As discussed in Section III, a player model, capturing a player's behavior, is responsible for providing the matching input for retrieving adaptation rules. We designed and implemented a skill-based player modeling method for *Acthung Die Kurve 3D*.

Since our main focus is on authoring adaptive generation, we defined a clear set of design principles for our player model: (i) simplicity, and (ii) intuitiveness in matching to skill profiles.

Our player model was inspired by the concept of experience points through practice, as observed in RPGs (*e.g.* Skyrim [25]). The player model captures six individual players skills, on: (i) ramp use, (ii) obstacle avoidance, (iii) floor edge avoidance, (iv) power-up selection, (v) AI defense, (vi) AI offense. We are confident these six skills are sufficient to capture most individual player behavior we could identify in this game. This choice was taken in collaboration with the original designers of *Acthung Die Kurve 3D*. Player behavior was identified by the observed in-game player actions (avoid, crash, use, kill), with relation to all content. Each skill is measured on an individual proficiency scale and, following from section II, is bounded on its lower value (positive only). An investigation on more complex player modeling methods and their (comparative) effectiveness is beyond the scope of this research. The following algorithm illustrates the player model:

Listing 2. Player Model algorithm

```
PlayerModel(Event e, Target t)
{
        if(e == Avoidance)
                if(t == obstacle)
                        skillObstacleAvoidance++;
                if(t == floorEdge)
                        skillFloorEdgeAvoidance++;
                if(t == AI)
                        skillAIdefense++;

        if(e == Use)
                if(t == ramp)
                        skillRampUse++;
                if(t == PowerUp(help, self) || t == PowerUp
                    (harm, others)
                        skillPowerUpSelection++;
                if(t == PowerUp(harm, self) || t == PowerUp
                    (help, others)
                        skillPowerUpSelection--;

        if(e == Crash)
                if(t == ramp)
                        skillRampUse -= 1*
                                ConsecutiveCrashes(t);
                if(t == obstacle)
                        skillObstacleAvoidance -= 1*
                                ConsecutiveCrashes(t);
                if(t == floorEdge)
                        skillFloorEdgeAvoidance -= 1*
                                ConsecutiveCrashes(t);;
                if(t == AI)
                        skillAIdefense -= 1*
                                ConsecutiveCrashes(t);
                        skillAIoffense -= 1*
                                ConsecutiveCrashes(t);

        if(e == Kill)
                if(t == AI)
                        skillAIoffense++;

        return <skillRampUse, skillObstacleAvoidance,
            skillFloorEdgeAvoidance, skillPowerUpSelection,
             skillAIdefense, skillAIoffense>
}
```

This player model increases or decreases a proficiency value for each of the six skills, where each increase/decrease unit corresponds to a specific action being registered. If a player is successful in avoiding a crash into a specific type of content, the corresponding skill is incremented. Successfully using ramps or killing AIs increases the corresponding skills. Power-up usage detects if the used power-up had a positive or negative effect for the player's success and increases or decreases the corresponding skill accordingly. Finally, crashing into a specific type of content decreases the corresponding skill. However, this decrease is proportional to the number of consecutive crashes into the same content type.

For this player model, we implemented the detection of all these in-game events, which was not present in the original game. The occurrence of each event fires an update to the player model. Furthermore, each event is being persistently registered in a individual log file.

Our player model presents some noteworthy issues. Its reliance on absolute proficiency values (and not in, for example, percentages) is more effective and simple in capturing absolute practice proficiency. However, just as its inspiration, RPG experience points, it requires an elementary understanding of how its values are affected to grasp the meaning of an individual value. Furthermore, our player model makes the assumption that consecutive crashes (or deaths) are an indicator that the respective skill level is currently overvalued. Therefore, they result in a decrease on that skill level, with a value that is proportional to the amount of consecutive crashes.

D. Integration with adaptation rules

In order to make *Achtung Die Kurve 3D* adaptive, we integrated our semantic model of adaptation rules in three steps, related to: player model, generation parameters and the design tool for authoring adaptation rules. The choices outlined below for the semantic model of the content generator can be considered as examples of how a PCG algorithm could be controlled. Section VII will further discuss the generalization of such choices. The player model is used as input in the algorithm for adaptation rules matching and retrieval (Section III). To match the player model with adaptation rules, they use the same six skills in their skill profiles. This means the input (a player model tuple) is directly matched with one or more skill profiles.

A matched adaptation rule will specify semantic entities and attributes in its content description. For *Achtung Die Kurve 3D*, we decided on the following semantic model:

- *Floor* entities have *Width* and *Length* attributes
- *Ramp* entities have *Quantity*, *Width* and *Placement* attributes
- *Obstacle* entities have *Quantity*, *Type* and *Placement* attributes
- *PowerUp* entities have *Frequency*, *Action*, *Type* and *Placement* attributes
- *AIs* have *Quantity* and *Placement* attributes

The non-numerical attributes require further explanation. The *Placement* attribute constrains the location to assign to that entity. It can hold three possible values: close to player, distant to player or random. The *Obstacle Type* attributes indicates obstacles that can be an individual block or a set of adjacent blocks. As for *PowerUps*, *Action* refers to its positive or negative effect (help, harm or random) and *Type* to the target (self, others or random). As for numerical attributes, they can be expressed as either a fixed value or an interval, in which case a random value (uniform distribution) will be selected in-game.

This type of attributes determines that each adaptation rule can return, in its content description, several configurations of entities and attributes. This includes configurations using the same entity. For example, a content description might include 2 ramps of width 1 and 1 ramp of width 2. Furthermore, for this research, we considered AIs as a specific piece of content, able to be generated. We felt this was a reasonable assumption since we are not changing (*i.e.* adapting) AI behavior but only (the amount of) its instantiation.

The floor generation algorithm, as described in sub-section IV-B, was implemented to accommodate the control constraints from the semantic model above. In other words, the input parameters of the generator are expressed in the same vocabulary and are able to steer it to create a floor with the specified characteristics. The Placement constraint in the generator dynamically computes a threshold, proportional to the floor size, to decide between close or distant to player. As for power-ups, positive or negative effects are randomly mapped to specific ones: increase speed, decrease speed, turn harder, turn softer, switch keys, no tail, thicker tail, thinner tail.

A useful feature of this generator is its behavior in the absence of input parameters. In this case, the input parameters of the previous floor are re-used, on an *individual semantic entity* base. For example, if the input parameters omit the floor size, the previous floor dimensions are used.

Finally, the design tool for creating adaptation rules was enriched with a semantic model that included the six matching player model skills and the semantic entities and attributes above. Using the tool's interface, designers can use this data to create and edit adaptation rules by instantiating and shaping skill profiles and associating each of them with content descriptions.

V. Designing adaptive generation

In this article, we propose adaptation rules as a generic semantic model that enables game designers to author adaptive game world generation. In the following sections, we assess this contribution by evaluating both the expressive range and the specificity range of our authoring mechanism. Ultimately, this assessment aims at finding out whether the use of adaptation rules by game designers can enable a player to experience the adaptive game they intended.

The goal of this research was to investigate *how* game designers can control adaptive game world generation, in an expressive and specific way, to create a desired user experience, *i.e.* an adaptive gameplay experience. To evaluate this approach, we asked game designers to create a wide range of adaptive gameplay experiences in *Achtung Die Kurve 3D*,

In order to maintain control and comparability within our design tests, we selected three game designers, all of them with amateur experience with game design and significant background in game technology, game development and content generation. They were all experienced gamers. The choice for amateur designers aimed at minimizing the impact that professional pre-conceived design strategies might have in the experiment, *i.e.* we wanted to take a first step in investigating the effectiveness of our approach, avoiding any bias discussion on any of the two design paradigms (adaptive vs. non adaptive).

Each of the designers was tasked with creating the same set of three adaptive versions of *Achtung Die Kurve 3D*, pertaining to three very different adaptive gameplay experiences, as follows:

- **Low-challenge game**: design a game that adapts to the player's skills, always maintaining a low challenge level.
- **Sudden-challenge game**: design a game that adapts to the player's skills, being easy to learn and hard to master.
- **Balanced game**: design a game that adapts to the player's skills in the most balanced and fair way.

Each type of adaptive experience entails (i) different degrees of dynamic behavior (the low-challenge game being the less dynamic in its adaptivity, followed by the balanced game and the sudden-challenge game), and (ii) different paces in adapting content (the sudden-challenge game offering less frequent but stronger challenge changes, the balanced game offering frequent but closely related changes and the low-challenge game offering less frequent and even more closely related changes). These two aspects were discussed in detail with the designers to confirm the same concepts were well understood and agreed upon. The goal was to capture examples of different types of adaptive experiences for the same game. So, we ended up with nine different adaptive versions of the game: three of each of the games above. We observed all design sessions, providing support and engaging in open-dialog interviews.

A. Results

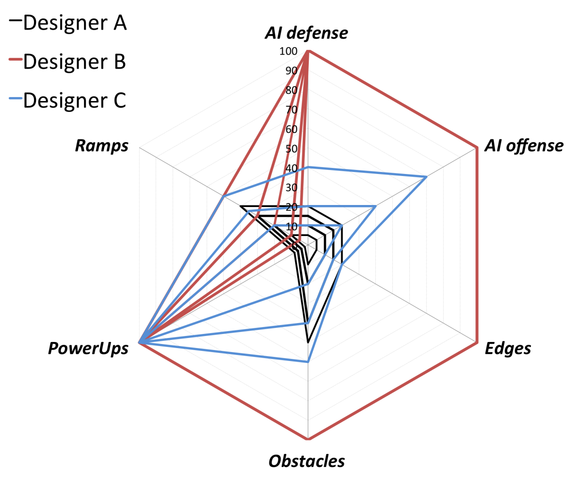
Fig. 4 (*a* to *c*) illustrates the skill profiles that all designers (*A*, *B* and *C*) considered for each game. They correspond to all adaptation rules created in these experiments.

Designer A was fairly consistent throughout his games, by creating adaptation rules focused on variation of all skills in a consistent manner, except for power up selection, which was ignored as a relevant skill. This designer increased the number of adaptation rules, using 4 in the low-challenge game, 6 in the sudden-challenge game and 10 in the balanced game.

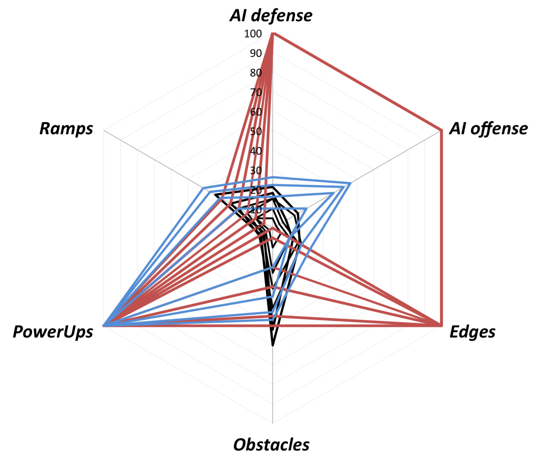
As for designer B, he varied the most throughout the three games. In the low-challenge game he chose to focus only on the ramp use skill, using 5 different adaptation rules. In the sudden-challenge game he extended his focus on ramp use, to also considering the obstacle handling skill, using 6 adaptation rules. In the balanced game, designer B focused on all skills except floor edge avoidance and power up selection, using 11 adaptation rules.

Designer C was fairly consistent, using fairly similar shapes throughout his three games. For the first 2 games, and like designer A, this designer focused on all skills, except power-up selection. However, the shapes of the skill profiles are substantially different between both designers, since they are determined by the different used values in each skill axis. In his balanced game, designer C chose to only focus on ramp use, obstacle avoidance and AI defense.

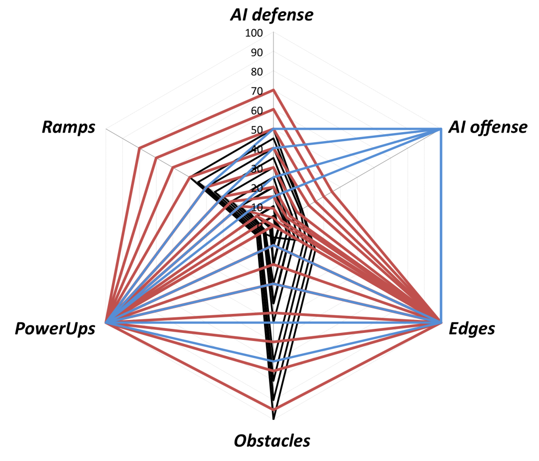
From the above description, we can appreciate how creative, diverse and specific the paths were that each designer chose, when defining which player profiles were deemed relevant for the assignments received. The same can be said regarding the content descriptions on which designers focused their adaptation rules. Designer A used all types of available content but chose to focus much more on ramp and obstacle variation throughout his games. Designer B excluded power ups from all his games and typically only varied one type of element at a time throughout his rules. He typically used lower AI variation and lower overall absolute values for the amount of content. Designer C used less adaptation rules than any of the other designers, using less wide variation in terms of content and choosing to specialize on type of content per game. More details about the content descriptions, including plots describing the declared content, can be found in [26]. Table I summarizes the most-specific elements in each game of each designer, in terms of content descriptions, by detailing the semantic entities that designers change most, and most frequently, across the three games.



(*a*)



(*b*)



(*c*)

Fig. 4. Skill profiles created for: (*a*) Low-challenge game, (*b*) Sudden-challenge game and (*c*) Balanced game, by all 3 designers (A, B, C)

B. Discussion

The results of this design experiment provide us with valuable data on the expressive range of our approach. Even though asked for the same three game versions, three different designers were able to design them in significantly disparate ways, resulting in nine games with easily recognizable differences.

Skill profiles were able to offer flexibility at capturing the

TABLE I
CONTENT DESCRIPTIONS: SPECIFIC FOCUS IN TERMS OF SEMANTIC ENTITIES VARIATION

	Low-challenge game	**Sudden-challenge game**	**Balanced game**
Designer A	Obstacles, ramps	Floor size, obstacles, ramps	Everything (gradual)
Designer B	Everything (slight)	Floor size, ramps (drastic changes)	Everything (very gradual, one entity or attribute at a time)
Designer C	AI	Floor size, ramps	Ramps, obstacles and power ups

designers' intent for distinct player characteristics. Designers A, B and C created significantly different shapes for their skill profiles. This was possible not only through a focus on different skill sets (*e.g.* notice the differences between designer A and B in the sudden-challenge game, in Fig. 4(b)), but also due to the use of different values in each skill set. Furthermore, designers were able to experiment with different skill profile shapes between their own 3 different games (*e.g.* designer B and his three games). The concept of absolute values in the player model (as explained in section IV) was easily understood by designers, with designers shaping their different desired player classes. However, in this type of experiment, the chosen values are influenced by the individual designer play testing, as well as by their assumptions about their target audience.

As for content descriptions, they were able to offer a large expressive range for designing nine very distinct adaptive games. This variety was straightforward to assess by both analyzing the content data on the resulting adaptation rules and observing gameplay. [26] Different designs naturally emerged from the variation offered by semantic entities and attributes, even within the same task. We argue that this is not only a result of the varied content, available to be combined, but mainly of the expressive power in the concept of adaptation rules. A good example is the balanced game, with: (i) designer A using all available content for very gradual but wide variations, (ii) designer B creating an adaptive experience where only one type of content varies at a time, very gradually, and (iii) designer C focusing on specific content (ramps, obstacles, power ups) on a narrower interval of variation.

Regarding specificity and the design experiment, it links with the available expressive range of our tool. As observed in Table I, adaptation rules are a valuable tool to construct specific and recognizable adaptive game features. Designer B provides the best example for this: in the low-challenge game he chose to specifically focus only on the ramp-use skill (Fig. 4*(a)*), and in the balanced game he created the very gradual content variation, just explained above.

Overall, this design experiment allowed us to observe that adaptation rules were able to effectively capture the intent of the designers authoring the requested adaptive experiences. Designers considered the design tool very expressive and were able to identify several ways of performing the assigned tasks, beyond their own designs. We very much would have liked to involve a larger number of designers, but that would have had a multiplicative effect on the rest of this study (see next section), which we just were not able to cope with. We believe it would have been much valuable to also involve some senior game designers in this study; unfortunately, the dedication it asked was incompatible with their intense schedule.

VI. ASSESSING ADAPTIVE GAMEPLAY

For the player experiment, our goal was to investigate if a desired adaptive gameplay experience could be conveyed through adaptation rules. The experiment consisted in play testing all the nine previously designed versions of *Achtung Die Kurve 3D*, inquiring players about their experience and logging their performance data. Furthermore, gameplay experience was discussed and recorded throughout the experiments. To capture meaningful data, we considered three players for each of the nine game versions, thus performing tests with a total of 27 different players. Players volunteered in an university campus location and the experiments were conducted by the authors, prior to any analysis of all the games characteristics.

Each play session consisted of: (i) brief explanation of the game and its generative nature, (ii) brief training session with the game, lasting typically 2-3 minutes, (iii) play testing one of the games for 15 min. At the end of the session, players were requested to:

1) plot how challenge/difficulty evolved over the time they played;
2) answer the question: *Which specific level features did you feel were changing while you played, and how?*

Players were asked to plot the perceived challenge over a grid representing the total game session time (*x axis*), with a minimum challenge of 0 and maximum challenge of 5 (*y axis*). Our aim was to compare those plots with the desired adaptive gameplay experiences (with the characteristics intended for the three types of games, outlined in section V), as created by designers, and check if, how and where they differ. Furthermore, we posed the second question above to confirm if the specific features of the adaptive game, as desired by each designer, were (easily) perceived by the players, if at all. The game registered all the events and player model values in individual log files.

A. Results

Fig. 5*(a)* to 5*(i)* illustrate the play test results, for the three players of each game (in rows) of each designer (in columns). Each sub-figure, thus, refers to a single game and includes the perceived challenge, as plotted by each of its three players (question 1, above). Below, each figure also includes a time line plotting each instant where a crash occurred for each player. From all the logged performance data, we considered this to be the best measure to illustrate how challenge evolved over time.

We also registered all the player answers to question 2. Table II summarizes these answers by focusing on which semantic entities were identified as the changing level features. We also asked *how* they changed, although this was a mere psychological mechanism to force reflection and, therefore, identify semantic entities. The answers largely correlate with the plotted challenge lines and with players referring to increases and decreases of semantic entities.

B. Discussion

The results of the player experiment (Figs. 5*(a)* to 5*(i)*) enable us to assess if the designers' intended adaptive experiences were conveyed by their adaptation rules and perceived by players as desired.

Overall, the challenge plots show that adaptivity is working and that the game is responsive to the players' performance, offering a variable and personalized challenge. This is visible even in the sudden-challenge game, a game with a clear tendency for gradually increasing challenge, where each increase step still includes some challenge variation.

These plots also show that the player-perceived gameplay experiences match with the intent of the game designers. All versions of the low-challenge game illustrate an adaptive game with a dynamic challenge variation, while keeping it at a low overall value. This is more evident if we compare the range of the challenge variation in all low-challenge games with the sudden-challenge and balanced games: most of the time is spent between challenge values 1 and 3.

In contrast, all versions of the sudden-challenge game show a gradual increase in challenge, with most time spent in the higher values (between 3 and 5). This fits with the designer's intent for the sudden-challenge game (very easy game for players with lower skills and extremely hard for players with higher skills).

The balanced game offered the most diverse results. This task was intended to yield the most balanced and fair game possible and was, thus, able to best accommodate all types of players. With the exception of designer B (explained later), the versions of the balanced game show a diverse challenge variation behavior (with values varying between 1 and 5) with no obvious tendency.

We consider that this lack of tendencies confirms that these games accommodate a wide range of differently skilled players, a natural expectation for a game with the characteristics of a balanced game (as requested to designers). For example, in the balanced game for designer C (see Fig. 5(i)), player 26 seems to quickly reach the top of his skills and is faced with higher challenges (subsequent failures resulting in a drop in challenge) while players 25 and 27 took more time to reach higher challenge levels.

An interesting case is the balanced game of designer B (see Fig. 5(h)), which can be explained by that game's adaptation rules. As mentioned in Table I, this game features a very gradual adaptive experience, where slight variations are inserted at a time, typically changing only one aspect with each adaptation rule (*e.g.*, increasing one AI at a time). The nature of this game allowed players to have a consistent learning curve, with gradual challenge increase.

We were interested in checking whether these observations from player perception were corroborated by actual in-game player data. Therefore, we compared these plots with logged data on crash time instants (see crossed time lines, in Fig. 5). This analysis is interesting mainly for showing different *trends* in challenge variation, since it is not accurate to match an objective time line (logged crash events) with a subjective one (player perception of challenge variation). The data on all versions of the low-challenge game shows less crashes than any of the other games with a slight increase towards the middle of the experiment. Data on all versions of the sudden-challenge game shows more crashes than any of the other games. They distribute with a lower concentration in the beginning of the game and a very high concentration towards the end. As for the balanced game, data is more diverse and does not show a clear tendency in the amount of crashes. However, its distribution seems to be more uniform across all the time line and, just like the remaining games, correlates well with the perceived challenge, supporting our hypothesis on whether player perception matched design intent.

Some of the results require further analysis, since they are exceptions to these observations. In the low-challenge game, the exceptions for the observed behavior are player 2 and player 8. When discussing gameplay with player 2, it was noticeable that this version of the low-challenge game eventually became too easy for the improving player, who became visibly bored with the experiment. We conclude that such boredom influenced his plot. In the case of player 8, the player crashed very little in the beginning, playing for a long time without dying. During play test, we observed and discussed that when challenge actually increased, the player and his lack of skill could not cope with that slight but *sudden* change. Although we observed that challenge would also slightly decrease with his now worse performance, we think the significant 'psychological shock' between the two phases of his play hindered his perception and determined his 'binary' answer.

From these results, we can conclude that the adaptation rules were effective in enabling individual players to experience the intended challenge variations, *i.e.* the adaptive experiences we had requested. The players' answers and gameplay data typically matched with what was expected and what was designed for each game. Furthermore, comparing Tables I and II, we can observe that the specific features (variation in a type of content) of each game were explicitly identified by players, who always mentioned at least one of the specific features summarized in Table I.

VII. Generalization

Supported by these results, one can wonder whether the semantic model for adaptivity authoring proposed in this article is substantially generalizable. In this section, we discuss the inherent features of our approach that support such generalization, its validity scope, as well as its limitations.

As mentioned before, adaptation rules pose no constraints on how skills are linked to content. For example, there is no obligatory amount of rules or skill profiles, nor is there a minimum or maximum amount of content that should be

TABLE II
VARIATION OF SPECIFIC LEVEL FEATURES, FOR EACH GAME AND DESIGNER, AS IDENTIFIED AND PERCEIVED BY THE 27 PLAYERS. THEY TYPICALLY REFER TO QUANTITY OR SIZE OF THE SEMANTIC ENTITY (ATTRIBUTES BETWEEN BRACKETS WERE ADDITIONALLY IDENTIFIED AS A VARYING FEATURE). ITALICS IS USED TO INDICATE AGREEMENT WITH THE FOCUS ORIGINALLY INTENDED BY DESIGNERS, FROM TABLE I.

	Low-challenge game			**Sudden-challenge game**			**Balanced game**		
Designer A	*ramps*	*obstacles*	*ramps, obstacles* (position)	*ramps, obstacles* (both drastically), power ups	*floor size, obstacles*, power ups (type)	*obstacles* (positions), *floor size*, ramps, power ups (type)	*ramps, AIs, obstacles*	*AIs, ramps, floor size, power ups (type)*	*floor size, AIs, obstacles*
Designer B	*ramps (width), AIs (position)*	*obstacles (position), AIs*	*AIs (position)*	*floor size, ramps*	*ramps*, obstacles (both drastically), *floor size*, AIs	*floor size*, obstacles	*empty in beginning; ramps, obstacles, AIs (slowly)*	*ramps (position), AIs (slowly)*	*obstacles (position), AIs (smarter)*
Designer C	*AIs* (smarter), obstacles (type), ramps (width)	*AIs* (position), ramps (width)	*AIs* (smarter)	*floor size, ramps*, zero power ups	*ramps*, obstacles (both drastically)	*ramps*	*ramps, power ups* (type)	*obstacles* (position), *ramps*	*ramps, power ups* (type)

specified for each rule. This lack of assumptions implies that the authored associations between player behavior and game world content can be applied to any domain designers see fit. This generic model is thus effectively *applicable to any game genre* where steering content generation is a suitable means for influencing gameplay.

Furthermore, the underlying building blocks supporting adaptation rules (player skill gameplay abstractions, semantic entities and semantic attributes) can all be created by designers to represent a given domain of player skills and game-world content (*e.g.* coin collection skill and coin entities). The ability to create generic adaptation rules is as large as the freedom to create such building blocks, ultimately determined by the underlying semantic model [27]. The authoring effort to create these building blocks is largely outweighed by their potential for reusability. For example, once specified, power up entities can be used in many games, like *Achtung Die Kurve 3D* or *Super Mario*, their sequels or any other game where power-up generation is useful.

Skill profiles can consist of a large variety and number of player skills. With no limitations on the number of the axes (or dimensions) of a skill profile, the model is generic enough to capture very complex multi-dimensional behavior. In fact, skill profiles and the underlying player skill gameplay abstractions are only limited by: (i) their representation on a bounded scale of proficiency values and (ii) their matching to the skills measured by a player modeling algorithm (to be developed for each game).

Content descriptions include combinations of semantic entities and attributes, which can represent a wide variety of types of content, such as materials, objects, spaces, scenes or whole worlds. Of course, the potential of these content descriptions is dependent on the existence of specific generation algorithms, which can realize those semantic entities and attributes into a specific game world, such as the *Achtung Die Kurve 3D* floor generator described in Section IV.

To summarize, our approach is generalizable for games that follow the assumptions: (i) gameplay is strongly tied with physical interaction with the game world content (*e.g.* platform, racing, dungeon games), (ii) the player model used can represent its features through bounded scale values, (iii) the game is appropriate for integration with a PCG algorithm, and (iv) the chosen PCG algorithm can be controlled or constrained through the specification of desired virtual world features.

A. Limitations and future work

From the assumptions above, one should acknowledge that player models and PCG may be seen as acting, at the same time, as both drive and limitation of our research. And justly so, as our design and proposals stem from both the advancements and limitations of both fields. In particular, we consider that player modeling limitations can be somewhat stronger. The discrete and bounded nature of most player models (section II) naturally leads to some false positives, with limitations on the range of players that can be accurately captured. As a result, adaptation will necessarily behave worse for such types of players, which may be more or less critical. We applied this bounded and discrete nature of player classification by defining floor values (positive only) for skills levels. Many current research results in the player modeling and adaptivity areas seem to show that these are good approximations: these models do not capture every possible player, but they significantly improve on capturing much more players than static gameplay experiences. Nevertheless, our method will necessarily need to be expanded before using it with other player model techniques; for example, using negative, continuous or infinite scales would require reassessing our current matching mechanism.

One might argue that our approach seems to be dependent on player skills. However, this apparent limitation is only a consequence of our focus on skill-based games, and it is easily overcome, provided that skills are simply represented by a proficiency name and a value. Such a representation can be used to capture player preferences, styles or other analogous features as long as they can be abstracted to bounded scales of values.

Using the Euclidean distance to calculate similarity between player profiles and adaptation rules was motivated by our previous research on adaptive game methods [26], where results were deemed quite successful. This was confirmed by our evaluation, both while developing the method and performing our user tests. However, we realize that for more complex games, with more complex skills and content, a more generic and standard metric like the Cosine similarity should be investigated, in order to replace the Euclidean distance and avoid its known limitations.

A valid issue to raise regards comparing the burden of authoring adaptation rules and a compatible specific generation algorithm (*i.e.*, our contribution) to the burden of authoring ad-hoc adaptive generation algorithms every time they are needed, for each game. Due to its semantic nature, our approach encourages reusability, where the semantic model and adaptation rules can be reused in games with similar domain and (player) features as, for example, game sequels. More importantly, ad-hoc adaptive generation algorithms are typically created by programmers. In contrast, as demonstrated in our design experiments (see Section V), adaptation rules enable an iterative loop between designing and testing, with no programming involved. As a result, game designers can thus directly focus on ways of controlling and tweaking adaptive generation as they see fit, without depending on programmers.

VIII. Conclusions

We proposed the use of adaptation rules by game designers to author adaptive game world generation. Adaptation rules are built atop gameplay semantics to steer the on-line generation of game content. Designers create adaptation rules by associating sets of *skill profiles*, describing skill proficiency, with *content descriptions*, detailing the desired properties of game worlds. Personalized game worlds can then be generated live through a matching and retrieval algorithm, whereby designer-specified content descriptions are selected that fit the player model being input.

We integrated this approach in a game, *Achtung Die Kurve 3D*, which served as a case study to perform both design tests and player tests. From the design tests, we concluded that adaptation rules provided game designers with a broad expressive range to control the adaptive generation of this type of game worlds and realize them into the requested adaptive experiences. Furthermore, this expressive range provided our game designers with the flexibility and variability to be specific in their intent, *i.e.* allowing them to craft similar adaptive experiences through rather different specific means.

From our play tests, we observed a considerable match between the perceived, the logged and the designed adaptive gameplay experiences, and concluded that adaptation rules were effective in conveying to players the adaptive gameplay experience intended by the tested designers.

Through our case study and subsequent reflection on our approach, we also identified its potential for generalization. Its lack of assumptions, as well as the freedom in authoring a semantic model, enable the method to be easily adopted in a variety of game genres, as long as content generation can be used as an adaptation mechanism.

We see various directions to expand this approach. Regarding skill profiling in adaptation rules, it would be interesting to investigate the possibility of weighing individual skills, *e.g.* with the goal of having the matching and retrieval process distinguish between essential and non-essential skills. This would allow for a fine-grained control over which skills are more important. Furthermore, it would be interesting to find more intuitive and interactive methods for creating content descriptions, analogous to the skills' radar charts. Inspired by current research [7], a promising direction would be to sketch a rough representation of the game world's entities and attributes.

Finally, a more thorough evaluation of this approach, its merits and potential is desirable. In particular, it will be interesting to perform more and in-depth tests with professional game designers, as well as with a broader base of players. We believe such feedback will be instrumental in establishing to which extent our approach can be used to support a new design paradigm in the future: the interactive authoring of adaptive game world generation.

Acknowledgments

We thank our game designers, colleagues and students for their collaboration in the case study. We also thank the anonymous reviewers of this article for their insightful comments.

References

[1] R. Lopes and R. Bidarra, "Adaptivity challenges in games and simulations: a survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 2, pp. 85 –99, june 2011.
[2] A. Liapis, G. N. Yannakakis, and J. Togelius, "Computational game creativity," in *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.
[3] G. N. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity," in *Proceedings of Conference on Foundations of Digital Games*, 2014.
[4] J. Kessing, T. Tutenel, and R. Bidarra, "Designing semantic game worlds," in *PCG '12: Proceedings of the 2012 Workshop on Procedural Content Generation in Games*, Raleigh, NC, USA, May 2012.
[5] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Transactions on Affective Computing*, vol. 99, pp. 147–161, 2011.
[6] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra, "A declarative approach to procedural modeling of virtual worlds," *Computers & Graphics*, vol. 35, no. 2, pp. 352–363, April 2011.
[7] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-aided game level authoring," in *Proceedings of ACM Conference on Foundations of Digital Games*, 2013.
[8] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Commun.*, vol. 7, no. 1, pp. 39–59, Mar. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=196108.196115
[9] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen, "Interactive storytelling: A player modelling approach," in *In Proceedings of the third Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE07.* Menlo Park, CA: AAAI Press, 2007.
[10] 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)*, 2010, pp. 180–185.
[11] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 54–67, 2010.

[12] S. Tognetti, M. Garbarino, A. Bonarini, and M. Matteucci, "Modeling enjoyment preference from physiological responses in a car racing game," in *IEEE Conference on Computational Intelligence and Games, 2010. CIG 2010.* IEEE, August 2010, pp. 321–328.

[13] 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*, 2010, pp. 297–304.

[14] A. M. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan, "An inclusive view of player modeling," in *6th International Conference on Foundations of Digital Games*, Bordeaux, France, July 2011.

[15] 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*, April 2007, pp. 252–259.

[16] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: dynamic difficulty adjustment through level generation," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, ser. PCGames '10. New York, NY, USA: ACM, 2010, pp. 11:1–11:4. [Online]. Available: http://doi.acm.org/10.1145/1814256.1814267

[17] N. Shaker, G. N. Yannakakis, J. Togelius, M. Nicolau, and M. O'Neill, "Evolving personalized content for super mario bros using grammatical evolution," in *Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012. [Online]. Available: http://aaai.org/ocs/index.php/AIIDE/AIIDE12/paper/view/5448

[18] G. Smith, A. Othenin-Girard, J. Whitehead, and N. Wardrip-Fruin, "PCG-Based Game Design: Creating Endless Web," in *Foundations of Digital Games 2012 (FDG '12)*, Raleigh, NC, 2012.

[19] R. Lopes, T. Tutenel, and R. Bidarra, "Using gameplay semantics to procedurally generate player-matching game worlds," in *PCG '12: Proceedings of the 2012 Workshop on Procedural Content Generation in Games*. Raleigh, North Carolina, USA: ACM, 2012.

[20] R. Lopes and R. Bidarra, "A semantic generation framework for enabling adaptive game worlds," in *ACE '11: 8^{th} International Conference on Advances in Computer Entertainment Technology*. New York: ACM, 2011.

[21] R. Lopes, K. Hilf, L. Jayapalan, and R. Bidarra, "Mobile adaptive procedural content generation," in *PCG '13: Proceedings of the fourth workshop on Procedural Content Generation in Games*, Chania, Crete, Greece, May 2013.

[22] R. van der Linden, R. Lopes, and R. Bidarra, "Designing procedurally generated levels," in *Proceedings of the second AIIDE workshop on Artificial Intelligence in the Game Design Process*, 2013.

[23] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, ser. Perennial Modern Classics. Harper & Row, 1990. [Online]. Available: https://books.google.nl/books?id=V9KrQgAACAAJ

[24] P. Sweetser and P. Wyeth, "Gameflow: A model for evaluating player enjoyment in games," *Comput. Entertain.*, vol. 3, no. 3, pp. 3–3, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1077246.1077253

[25] Bethesda Game Studios, "The Elder Scrolls V: Skyrim," 2011, Bethesda Softworks.

[26] R. Lopes, "Gameplay semantics for the adaptive generation of game worlds," Ph.D. dissertation, Delft University of Technology, September 2014.

[27] T. Tutenel, "Semantic game worlds," Ph.D. dissertation, Delft University of Technology, December 2012.

Ricardo Lopes received the B.Sc. and M.Sc. degrees in software engineering from the Technical University of Lisbon, Lisbon, Portugal, in 2007 and 2009, respectively. He received his Ph.D. in 2014 in computer science from Delft University of Technology, The Netherlands, with a thesis on "Gameplay semantics for the adaptive generation of game worlds."

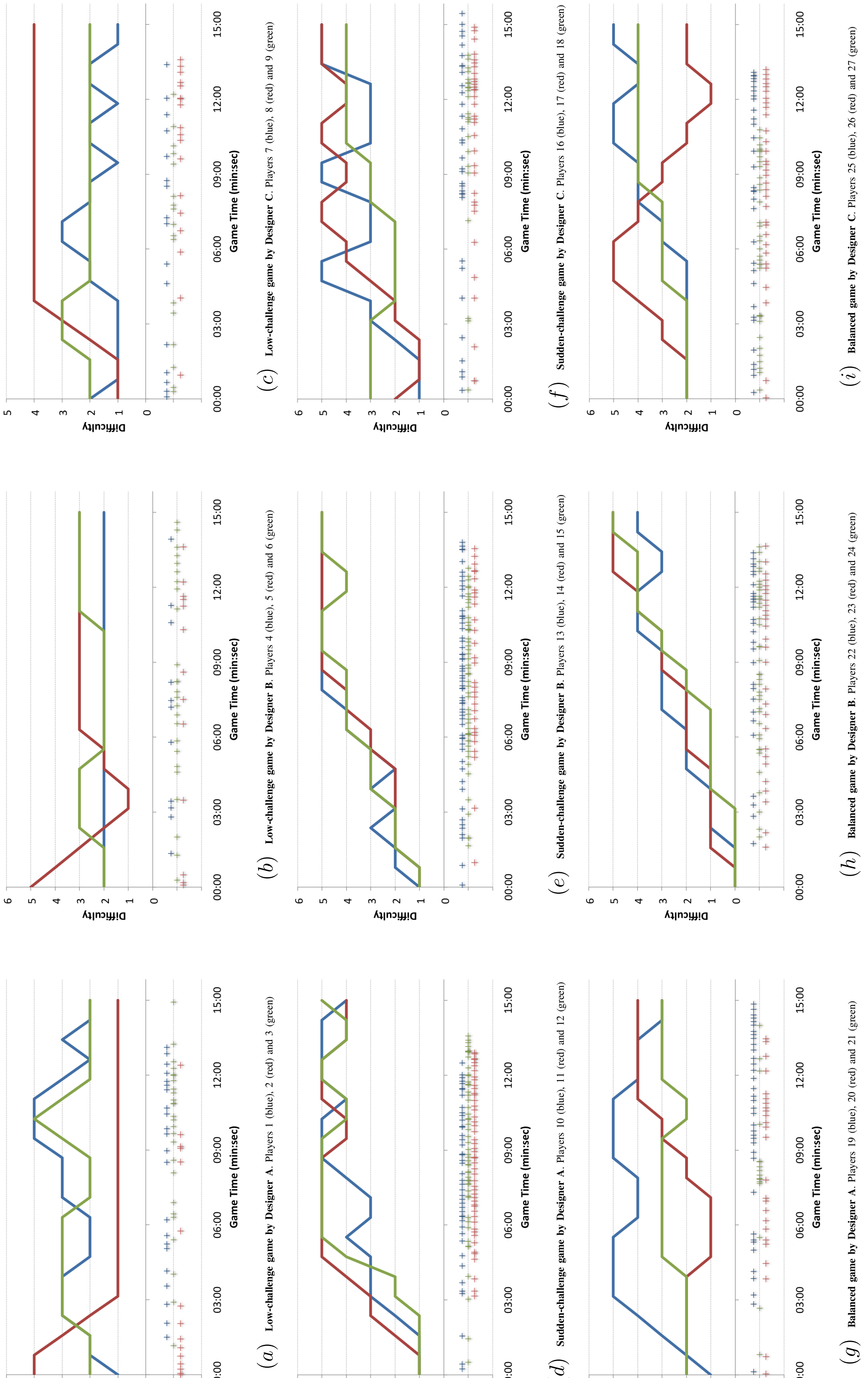
Recently, he has worked in the fields of simulation, prescriptive analytics and financial engineering. He is currently a software engineer at Cardano Risk Management b.v., the Netherlands. His current research interests include adaptivity in games, player modeling, interpretation mechanisms for in-game data, and (online) procedural generation techniques.

Elmar Eisemann studied at the École Normale Supérieure Paris (2001-2005) and completed his PhD at INRIA Rhône-Alpes (2005-2008).

He is currently a Professor in Computer Science, heading the Computer Graphics and Visualization Group at Delft University of Technology, The Netherlands. Before, he was Associate Professor at Télécom ParisTech (ENST) until 2012, and a senior researcher and group leader in the Cluster of Excellence at the Max-Planck-Institute and Saarland University (2008-2009). He is an author of the book "Real-Time Shadows" (AK Peters) and co-organized EGSR 2010, 2012, and HPG 2012. In 2011, he received the Eurographics Young Researcher Award.

Rafael Bidarra graduated in 1987 in electronics engineering at the University of Coimbra, Portugal, and received his Ph.D. in 1999 in computer science from Delft University of Technology, The Netherlands.

He is currently associate professor Game Technology at the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology. He leads the research lab on game technology at the Computer Graphics and Visualization Group. His current research interests include procedural and semantic modeling techniques for the specification and generation of both virtual worlds and gameplay; serious gaming; game adaptivity and interpretation mechanisms for in-game data. Rafael has numerous publications in international journals and conference proceedings, integrates the editorial board of several journals, and has served in many conference program committees.



(*a*) **Low-challenge game by Designer A**. Players 1 (blue), 2 (red) and 3 (green)
(*b*) **Low-challenge game by Designer B**. Players 4 (blue), 5 (red) and 6 (green)
(*c*) **Low-challenge game by Designer C**. Players 7 (blue), 8 (red) and 9 (green)
(*d*) **Sudden-challenge game by Designer A**. Players 10 (blue), 11 (red) and 12 (green)
(*e*) **Sudden-challenge game by Designer B**. Players 13 (blue), 14 (red) and 15 (green)
(*f*) **Sudden-challenge game by Designer C**. Players 16 (blue), 17 (red) and 18 (green)
(*g*) **Balanced game by Designer A**. Players 19 (blue), 20 (red) and 21 (green)
(*h*) **Balanced game by Designer B**. Players 22 (blue), 23 (red) and 24 (green)
(*i*) **Balanced game by Designer C**. Players 25 (blue), 26 (red) and 27 (green)

Fig. 5. Difficulty/Challenge trend perceived by each player, per game, correlated with a timeline of the actual player crashing (death) events.