Mobile adaptive procedural content generation

Ricardo Lopes
Computer Graphics and Visualization Group
Delft University of Technology
The Netherlands
r.lopes@tudelft.nl

Ken Hilf
Entertainment Technology Center
Carnegie Mellon University
Pittsburgh, PA, USA
kenhilf@gmail.com

Luke Jayapalan
Entertainment Technology Center
Carnegie Mellon University
Pittsburgh, PA, USA
ljayapal@alumni.cmu.edu

Rafael Bidarra
Computer Graphics and Visualization Group
Delft University of Technology
The Netherlands
r.bidarra@tudelft.nl

ABSTRACT

The nature of most modern mobile games is different from that of most computer/console games, which are typically targeted at casual gamers and are played in a wide variety of space, time and device contexts. We argue that this feature of mobile games naturally fits with adaptive procedural content generation (PCG). In this paper, we propose the integration of two PCG-based approaches (experience-driven and context-driven PCG) to support the generation of adaptive mobile game levels. We present and discuss the implementation of our approach in an existing game, *7's Wild Ride*. Gameplay semantics and player modeling are used to steer a level generator, featuring a time-dependent dynamic difficulty adjustment mechanism. From our two user studies, we conclude that (i) context-driven levels are preferable over traditional ones, and (ii) the game can adapt to different player types, keeping its gameplay balanced and player satisfaction.

Categories and Subject Descriptors

I.2.4 [**Computing Methodologies**]: Knowledge Representation Formalisms and Methods; K.8.0 [**Personal Computing**]: Games

General Terms

Algorithms, Design

Keywords

adaptive games, procedural content generation, semantics

1. INTRODUCTION

The nature of most modern mobile games, *i.e.* games played in smartphones and tablets, is different from that of most computer/console games. Mobile gaming is cannibalizing the casual gamer, by bringing more demographics into the marketplace, along with their cheaper, less powerful hardware [2], which are used in a wide variety of environments and conditions. By observing the most downloaded games in mobile marketplaces like Apple's App Store and Google Play, we can confirm that a vast majority falls within the definition of a casual game [9]: "inexpensive to produce, straightforward in concept, easy to learn, and simple to play".

With their casual nature, these mobile games need to appeal to wider audiences. Accommodating player characteristics, styles or preferences for such a variety of demographics is therefore a bigger concern. The usual solution for catering for casual players, *i.e.* shorter, simpler gameplay, tends to alienate some more demanding players, typically from the hardcore audience. Furthermore, this type of games are played in a wide variety of environments, conditions and time availability, dependent on the player context. Again, the usual approach to accommodate this diversity is to enhance the casual aspect of these games: keep them straightforward, simple and short.

For this type of games, attracting and retaining all player types, while providing meaningful gameplay in a wide variety of contexts, raises additional challenges.

1.1 The case for mobile adaptive PCG

Adaptive games have the potential to be more personal, by adjusting their content or mechanics to better serve individual player needs [7, 16]. Ideally, adaptive games can accommodate for all types of players (casual or not) and their wide variety of commitment, skills or styles, *i.e.* player experience. Additionally, adaptive games have the potential for being meaningfully responsive to a varyety of player-related contexts. For example, for a player waiting at an airport for a specific time period, an adaptive game could adjust the overall game duration to fit that time-constrained context.

Currently, standard approaches for supporting such adaptive games are increasingly based on procedural content generation (PCG) [7, 16]. We argue that this adaptive PCG can tie in very naturally with the *casual* mobile gaming paradigm (henceforth, simply referred to as mobile games), where large ranges of player commitment and circumstances should be considered. To accommodate this wide variety of player experience and player-related contexts in mobile games, we envision two different modalities of adaptive PCG: experience-driven and context-driven.

1.2 Experience and context-driven PCG

Our research goal is to investigate the integration of experience and context-driven PCG in the mobile games domain, and to assess its potential value[1].

In *experience-driven* PCG, generative methods are responsive to some sort of player-generated *gameplay-specific* data. For this paper, we propose a semantics-based experience-driven PCG approach where online content generation is used to dynamically adjust the game's difficulty. This dynamic adjustment of game difficulty (DDA) to the personal skill level of a player has the potential to attract and retain a larger variety of players.

We define *context-driven* PCG as the use of generative methods to yield content that fits some player-related context. The demands a given context puts on that content generation are extrinsic to the gameplay, and respond to that player's concrete situation (e.g. available time, weather, location, health,...) Regardless of whether a context is explicitly input or implicitly derived, the neat effect will be a direct steering of the generator (even if players are unaware of it). Moreover, such demands of context-driven PCG can stretch deeper and more meaningful than players might anticipate. For example, for the airport time-dependent context mentioned above, setting a play duration constraint should not only determine the time to play, but also smoothly scale the full gameplay experience under that constraint. We envision that this form of adaptive PCG provides a powerful basis to accommodate for a variety of player-related contexts. In this research, we investigated a first example of context-driven PCG: level generation for an *explicitly* set play duration constraint.

Our case study in this paper is a first step towards demonstrating that experience-driven and context-driven generation fit well together to support an adaptive mobile game experience. For this, we developed a game that allows you to specify how much available time you have to play; a level is then generated online, fitting both: (i) that time-constraint (context-driven PCG), and (ii) your measured skill level (experience-driven PCG). For this, gameplay semantics, *i.e.* gameplay information about the world and its objects, is used to support and steer the procedural content generator.

The above mentioned demands of mobile gaming (large variety of players, skills and contexts) cannot be met anymore by simply using handmade static levels. We believe that both experience-driven and context-driven game levels provide a much better and richer alternative, and that our adaptive PCG proposal is a valuable contribution to solve this mobile gaming challenge. Recent mobile games like *Canabalt* (2009) or *Robot Unicorn Attack* (2010) confirm this trend, by already incorporating a simple form of experience-driven PCG. Game content is there generated on the fly, based on player performance, but only sampled at the single instant generation is executed [6]. Our research, on the other hand, uses collected player performance (plus time) progression, over the whole game session, to steer generation.

This paper is structured as follows: in Section 2, we survey existing related work. Section 3 outlines the methods behind our case study's generator. In Section 4, we explain our dynamic difficulty adjustment method and player modeling approach. Section 5 presents and discusses our results (play testing), preceding our conclusions in Section 6.

2. RELATED WORK

This sections analyzes previous work related to our research goals and methods.

2.1 Dynamic difficulty adjustment

When supported by PCG, DDA can be considered a type of experience-driven PCG. Some previous research has been done on DDA, either PCG-based or not. Among others, non-PCG DDA approaches have been used to influence players health or ammo in a first-person shooter, depending on their individual performance, through probabilistic models [3], assign dynamic scripts to control NPC behavior, in response to weights associated with the player behavior [13], evolve opponent AIs in real-time strategy games, through neural-evolution methods [10] and adapt intelligent agents behavior to fit the players skills, using reinforcement learning [1].

For our case, PCG-based DDA is more interesting and applicable. Shaker *et al.* [11] generate platform levels for *Super Mario Bros*, using grammatical evolution. The grammatical evolutionary algorithm uses models of collected player experiences as the fitness functions to search through the generative space. The authors go beyond DDA, optimizing (off-line) the game levels to improve engagement, frustration and challenge. Our work differs significantly, since we focus on on-line generation of platform levels, as the player is playing, for DDA purposes.

Jennings-Teats *et al.* [4] also focus on generating (on-line) platform levels to balance difficulty with player skills. A statistical model of difficulty and a model of the player's current skill level are used, through mass data collection and machine learning techniques, to select the appropriate level segments to generate. Although our work uses a similar recombination approach to generate levels, it differs by: (i) the use of semantics to support it, and (ii) integrating context-driven PCG to the DDA approach. The use of semantics is of distinct importance because it allows designers to control the generator while enabling the reusability of our approach in similar games.

2.2 Context-driven and mobile PCG

Context-driven PCG, as we defined it, has not been so widely investigated as DDA. There are however some related methods that share context-driven PCG requirements. Smith *et al.* [12] created *Endless Web*, a platform game in which the player directly interacts with the level generator. The game goal is to reach certain locations in the map of the generative space, by changing the way the level is generated with explicit player actions. It somehow relates to context-driven PCG since it allows the player to have a direct effect on the procedural generator.

Verma and McOwan [15] proposed a methodology to adapt a classic mobile game (*Snake*) to suit the player skills. The user plays a version of the game, and then selects to play an either easier or more difficult game than the previous one. A genetic algorithm uses player and previous level data to search

[1]a video on this research can be found at http://graphics.tudelft.nl/˜rval

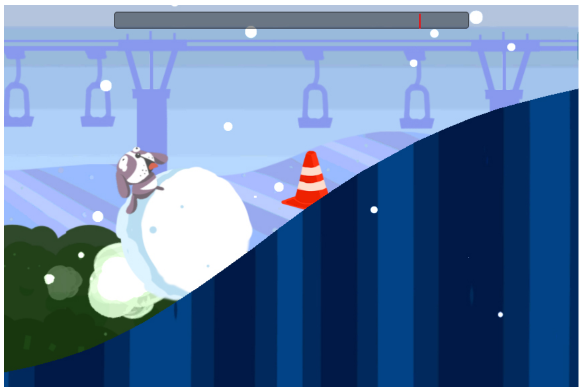


Figure 1: *7's Wild Ride*: player character is struggling with balance on the snowball due to an upwards slope (danger zone). Also, the snowball is about to pickup the obstacle cone halfway the slope.

for a game that fits the request. Although mobile-based and allowing for user interaction, our research differs significantly not only because generation occurs on-line, but also because we consider a context-driven strategy: converting requested play duration into an appropriate generated level.

3. CONTENT GENERATOR

To demonstrate our approach proposed in Section 1, we implemented an adaptive version of a mobile game developed (in the Unity game engine) by a multidisciplinary team for a course project at the Entertainment Technology Center of Carnegie Mellon University. The game, *7's Wild Ride* (in Fig. 1), is a side-scrolling platform game where players have to prevent the main character from falling off a rolling snowball. They keep the character in balance by: (i) tilting the mobile device left and right to counteract the gravity effects of navigating slopes, and (ii) by jumping on the snowball over obstacles that it picks up. Additionally, score points, power ups and achievements are part of the game. To balance the character, the player has to keep it in a safe zone, on the top of the snowball. If the player is riding the snowball in an unstable position, an animation is triggered to warn that he must correct his balance to avoid falling (see Fig. 1).

This mobile game was originally developed using a fixed set of designer-handmade levels. Therefore, our first step was to develop a content generator able to create levels on-the-fly[2]. Control over the generator (as explained in Section 4) supports its dependency to player experience and context.

The generator creates levels by selecting and combining *level segments* (chunks) from a content library, as the player advances through the level. Level obstacles are also selected independently from the content library and placed on valid locations of the level segments. This means that level segments can be dynamically coupled with obstacles, to synthesize different combinations, thus ensuring increased variability.

The generator is always running as the player advances through the level. When the player reaches the midpoint of a level segment, it immediately selects and retrieves the next chunk from the library, placing it accordingly. Supported level segments can have different lengths. The character's speed and the hardware capabilities of current mobile devices ensure that online generation does not cause performance drops. The framerate remains identical to the original game. Additionally, and since this game is a right side-scrolling game (going left is not allowed), previous level segments are removed, to save up memory. This means that, at any given moment, only a fixed number of level segments exist before and after the player's current position. Players are unaware of this due to the limited camera field of view (see Fig. 1). Fig. 2 shows examples of the level segments included in the library, and an example global view of a generated level, at a given instant.

[2]from this point on, all *7's Wild Ride* mentions refer to the new adaptive version

4. ADAPTIVE CONTROL

Control over the content generator can be exerted by selecting which level segments, obstacles and obstacle locations to combine. In this section, we describe the main methods used to support this on *7's Wild Ride* adaptive case study.

4.1 Semantics and DDA

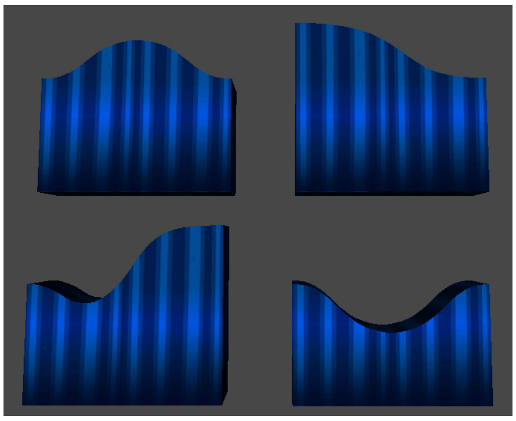
To investigate experience-driven PCG in the context of our research goal, we implemented DDA on *7's Wild Ride*. We used virtual world semantics to control the procedural generator. In our own previous work [14], we define virtual world semantics as all information about the world and its objects, beyond their geometry. This includes object properties, high-level attributes and functional information, as well as interrelationships (geometric, functional, etc) among different objects. Our semantic library *Entika* [5], a hierarchical class (relational) database, is responsible for storing all game objects and each of its associated semantics. Game designers use this library to specify semantics, atop game world geometry, in a generic and reusable way. Semantics imported from this library can be used as knowledge to automatically constrain and control PCG methods. This approach has already been successfully deployed to generate experience-driven game worlds, which adapt to a player's behavior [8].

In *7's Wild Ride*, additional knowledge on the semantic level segments and obstacles entities can be specified, prior to the game's deployment. The generator searches and selects content only if its semantics match some desired input, typically expressed in the same semantic terms. An example is obstacle placement: locations for placement are only selected if they were marked semantically as valid.

Semantics was also used to label and select the difficulty associated with the different segments and obstacles in the library. Partnering with the original *7's Wild Ride* designers team, we were responsible for not only specifying the new semantics, but also for designing the new DDA mechanism.

Three independent difficulty scales were defined: (i) slope selection, (ii) obstacle placement, and (iii) obstacle frequency. These were deemed sufficient to capture and influence all difficulty-related aspects of the core gameplay. Semantics was pre-specified for all content, describing (i) the difficulty of the shape of a level segment (combinations of two level segments were also considered), and (ii) the specific difficulty of each valid location for the obstacle placement. For this, two independent numerical scales were defined.

Control over the generator can be exerted by supplying input values for these three independent difficulty scales. For a certain input, the generator algorithm retrieves which library content was specified as appropriate to that difficulty


(*a*)

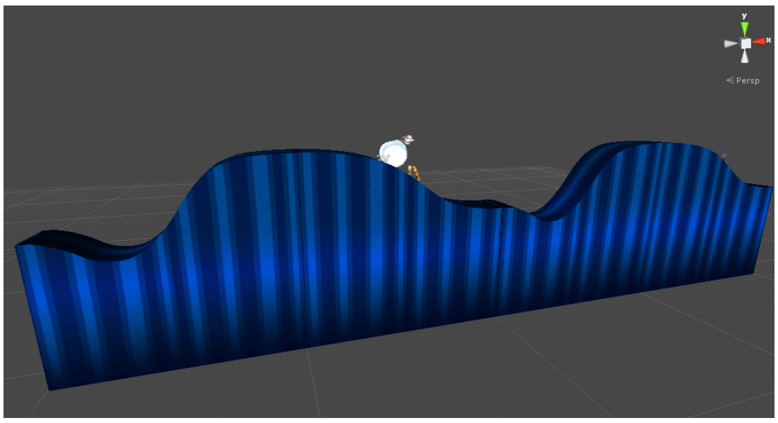

(*b*)

Figure 2: *(a)* Examples of chunk level segments, to be used by the content generator, *(b)* Example of a level generated during play (debug view). Notice the two placed obstacles (next to the player and at the far right).

scale. Additionally, obstacles placement follows a frequency constraint: only every N level segments is an obstacle selected and placed. Each input combination of the two difficulty scales might have several possible results (level segment and obstacle placement combinations). Since they were all deemed as equally difficult, the generator selects one randomly.

With this approach, semantics can enable designers to author the DDA mechanism, with no programming involved. By changing the semantic attributes of placement and difficulty for all associated content, designers can effectively influence the behavior of the generation algorithm. This opens new possibilities for designers to hold some expressive power over this form of adaptive PCG.

4.2 Experience-driven PCG

The above semantics-based control of the generator already supplies a basis to support DDA. To fully realize this we developed a specific player model to steer the generator. Working jointly with the original *7's Wild Ride* team, we were most familiar with its design goals and principles. As such, we created this player model.

The proposed player model directly maps measured skills into the three difficulty scales described above. Player performance is measured and converted into a desired new level of difficulty for that player. Each difficulty scale is influenced by the following measured skills:

- slope selection: *BalancePerformance*, *BalanceDeath*
- obstacle placement: *BalancePerformance*, *ObstacleDeath*
- obstacle frequency (or period): *ObstacleDeath*

BalanceDeath and *ObstacleDeath* flag whether the player died because of, respectively, loosing balance or hitting an obstacle. *BalancePerformance* measures the rate of transitions between the character's safe and danger zones (see Section 3). A player with less of these transitions is typically more skilled at balancing his character.

For all skills and difficulty scales, the same strategy was applied: an improvement on skill performance leads to an increment on the corresponding difficulty scales, and a decrease, to a corresponding decrement. Since all difficulty scales are independent, individual and specific behavior can be captured by the player model. This can lead to specialized reactions by the generator. For example, if the player is mostly dying from hitting obstacles, only the obstacle related difficulty scales are affected. Below is the pseudo-code for the mapping of skills performance into the difficulty scales they affect.

```
// All difficulty scales are pre-initialized to
    their minimum values

//transitions measured per time elapsed per level
    segment
//performance is normalized to the current
    difficulty
balancePerformance = transitions / (timeElapsed/
    obstaclePeriod);
balancePerformance = normalize(balancePerformance,
     slopeSelection);
//BalancePerformance
if(balancePerformance>balancePerformanceAvg OR
    transitions = 0)
        slopeSelection+=s1;
        obstaclePlacement+=o1;
else
        slopeSelection-=s2;
        obstaclePlacement-=o2;
//BalanceDeath
if(BalanceDeath AND timeAlive<timeAliveAvg)
        slopeSelection-=s3;
//ObstacleDeath
if(ObstacleDeath AND timeAlive<timeAliveAvg)
        obstaclePlacement-=o3;
        obstacleFrequency-=f1;
```

This player model is re-evaluated every N level segments, which might lead to difficulty scale changes. If, in the current N segments, a player improves his *BalancePerformance* (compared to his past average) on the current N segments, the values for both the slope selection and the obstacle placement difficulty will increase. A decrease of these occurs in the opposite case. Both difficulty scales are affected since jumping over obstacles can provide additional balance challenge. When the character dies and his last life duration was shorter than the average, difficulty scales are decreased accordingly, for each death type. The player model above only contemplates obstacle frequency decreases. The simultaneous presence of frequency increases and our context-driven

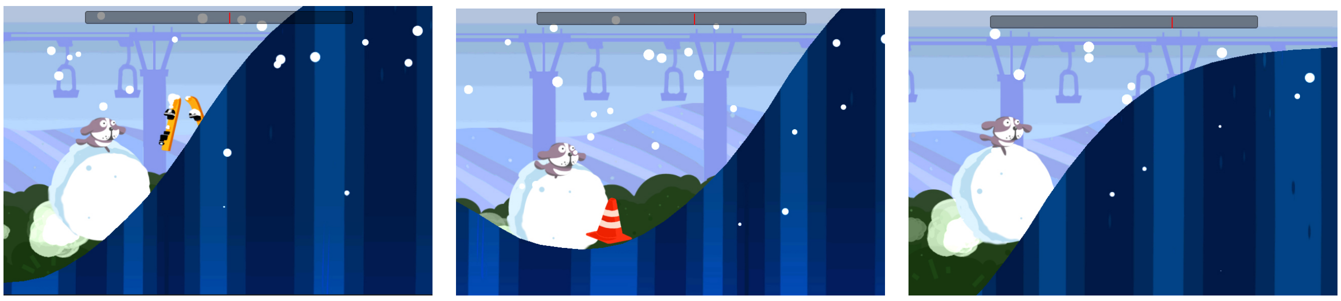

(*a*) (*b*) (*c*)

Figure 3: In *(a)*, player dies by not jumping successfully over the obstacle. Later on, in a subsequent life, an obstacle is placed at an easier location, in a similar situation *(b)*. Eventually, after repeated deaths by obstacle collisions, no obstacle is placed at all *(c)*.

PCG approach, explained in the next section, would lead to steep difficulty increases, overburdening players. All the increase and decrease operations might not lead to immediate difficulty changes. We implemented two aspects to assure that the generator affects gameplay and difficulty only when continuous consistent performance improvements or declines occur. First, each increment is smaller than 1 (*e.g.* $s1$=0.2). Second, the generator only considers new inputs from the player model when a difficulty scale changes its integer value. Each of the difficulty scales are numeric scales: 1-5 for slope selection, 1-4 for obstacle placement, 0 to M for obstacle frequency. All the values for the parameters (N,M,s1,s2,s3,o1,o2,o3,f1) were determined experimentally, with users, as described in Section 5.

With this player model in place, player performance can be converted into gameplay skills which are translated into dynamic difficulty scales. These scales steer the semantic generator to synthesize the appropriate content, classified according to that difficulty. This way, the difficulty of the level generated ahead is adjusted to match the player performance. In Fig. 3, you can see an example of the functionality of our experience-driven DDA, as observed in a user evaluation.

4.3 Context-driven PCG

To investigate context-driven PCG, the generation of *7's Wild Ride* game levels has been made dependent on a time constraint, explicitly specified by the player. The consequence of such a constraint is that it will directly steer the generator, adjusting the game to a context, in this case, the available time.

We implemented a straightforward solution to support this, based on our generation approach (online recombination of level segments). A timer is responsible for stopping level generation and gameplay, thus respecting the requested context. Additionally, we opted for a strict strategy for this timer, where the pause and death menus do not interrupt time counting, even though no generation is occurring. This way, we give high priority to fulfill the player's request, regardless of interruptions.

More importantly, and as explained in Section 1.2, setting a time constraint should also scale the full gameplay experience to that requested time. In our case, the requested time has a direct effect on the difficulty of the game level, which is adjusted (scaled) not only to fit the player experience, but also to fit the requested time.

This scaling relates to game progression. In many games, difficulty typically increases with the advancement on a handmade game level, where the final sections provide harder challenges than the initial sections. The same happens between different handmade levels, to provide the player a sense of progression.

In our case, we applied the same progression principle, but in the context of available time. As the player advances in the generated game level, approaching the requested time limit automatically increases difficulty to give the player a notion of proportionally scaled progression.

The generator starts by dividing the requested time into five equal slots which correspond to the five slope selection difficulty values. The second and third time slots are grouped to create four time-based level sections: beginning, middle, pre-final, final. As you progress through these sections, all difficulty scales increase automatically by a parameter t, every P level segments. Both t and P change according to the section the player is in. The closer a player approaches the final section, the more difficulty increases in frequency and value. As before, all parameter values were determined experimentally, with users, as described in Section 5.

With this approach, the context-driven PCG loop is closed: the requested time context affects difficulty increases, which has a direct effect on the generated content, which scales gameplay progression to the available and requested time context. Additionally, this difficulty progression is still being balanced by the experience-driven DDA, which guarantees that each progression is still adjusted to each individual player performance. Ultimately, difficulty and content are being dynamically adjusted to both the individual player performance and requested time context, thus creating personalized and unique gameplay experiences.

We are aware that PCG and, specifically, DDA can potentially prevent fair comparison of game scores and achievements, among players. We are interested in the debate on the (un)desirability of such comparisons, but that concern is not currently present in our research goals. Therefore, no scoring or collectibles game mechanics were implemented in our case study.

4.4 Semantics and reusability

Our proposed approach can be generalized and applied to other games beyond *7's Wild Ride*. This is, in fact, enabled by the semantics-based generator, while saving considerable work.

The content generator can be applied to any game (most notably platformers) which levels/world can be generated by the sequential combination of world segments and placement of additional game objects. The semantic library combines all information about the individual content (geometry) with the knowledge that steers its retrieval and placement by the generator. In our case, we defined the difficulty scales as these steering semantic attributes, which means they can be reused and extended in other projects. Using other generation control mechanisms beyond difficulty can be easily achieved; it would entail using Entika to create and label new semantic attributes and minor changes to the retrieval process. As for our simple player model, it is certainly specific to *7's Wild Ride*, as it converts player performance data into our semantic difficulty scale values. Except for some machine-learning methods, typical player models are mostly dependent on specific games. However, with semantics, the implementation of a conversion from performance data to semantic attributes, as a new final step within such models, will assure their smooth integration with the generator.

As for our first experiment on context-driven PCG, this approach does not take advantage of gameplay semantics yet. In our future work, we plan to not only investigate new forms of context-driven PCG (beyond time), but also their integration with gameplay semantics. As with DDA, this would ensure their generalization and reusability.

5. RESULTS AND DISCUSSION

To evaluate our approach, we performed two distinct user studies. We opted to use two distinct sets of participants to investigate each of our PCG-based methods: time context-driven and DDA. This not only avoids longer questionnaires, but also appeals to the different characteristics of both user groups, as explained below.

5.1 Time context-driven PCG

In our first user study, 17 participants (college students) played several game sessions and were interviewed. The goal was to define which parameters (see Section 4) should be used in the generator, to maximize a satisfying gameplay experience. Participants played two different versions of the game (different sets of parameters), and even re-played them by directly changing the parameters (available in these versions interface). This resulted in the best set of parameters to use in our second user study.

Since college students understand better the concept of limited time (when compared to our second user group), participants also evaluated our context-driven PCG approach. We performed a questionnaire to investigate the value of our context-driven PCG implementation, as seen by players. Participants were invited to choose the time available for their game session, using a slider available in the game start menu. Additionally they were briefed on the meaning of our time-based level constraint, *i.e.* on the fact the game session would be generated to fit the requested time window.

We asked participants to rate, from 1(less) to 5(more), how valuable they thought this time context-driven generation was. Additionally, we asked them which type of levels they would rather play again, on their mobile devices: time-based generated levels or normal levels. 35% of participants ranked the value of time-based generation as 3, another 35% as 4 and 30% as 5. 65% of the participants would prefer to play time-based generated levels again, and 35% normal levels.

These results demonstrate the potential value in games which generate content in response to a player-specified time context. All the participants, even the minority which would not prefer this mechanism over traditional ones, recognized some positive value in this type of approach. It was clear from the interviews that everyone saw multiple applications for this method, even beyond time constraints.

Additionally, we decided to ask about the role of adaptive PCG when it comes to comparing gameplay (scores, achievements). After explaining to participants to what extent adaptive PCG was supporting this game, 58% of the participants found the loss of gameplay comparison (between players) as important. This demonstrates that this is an important issue needing to be addressed. Finding ways of normalizing PCG-enabled gameplay and making dynamic game levels comparable (in terms of gameplay) remains an open question that deserves future research.

5.2 Dynamic difficulty adjustment

In our second user study, we focused on assessing our DDA's case study. 22 participants (children between 6 and 12, visiting a science museum) played *7's Wild Ride*, using the set of parameters discovered in the first user study. Data about their performance was logged automatically and a short interview was individually conducted to assess the DDA mechanism. Before each game session (3 minutes), players had to complete a tutorial to learn the basic game mechanics: balancing and jumping. The children selected for this user study were identified *in loco* as more casual players with less experience, thus less commitment and skill.

Among other data, we logged the variation of the slope selection difficulty scale, measured throughout the game session for all participants. From the analysis of the collected data, we can identify two player categories: without progression (Fig. 4a) and with progression (Fig. 4b). For players without progression, the difficulty scale typically shifts back and forth between two or three values, between 1 and 3. For players with progression, the difficulty scale gradually increases more or less linearly, between 1 and 5. These patterns capture an underlying skill evolution, where players improve (or maintain) their performance. As expected, players without progression died more often (average of 6.72 deaths) than players with progression (average of 3.6 deaths), indicating this skill difference.

Challenge - We asked all participants to rate the challenge they felt throughout the game, from 1(less) to 5(more). Answers are summarized below, in Table 1.

Table 1: Questionnaire ratings: game challenge and unfairness

	1(less)	2	3	4	5(more)
Challenge	0%	0%	55%	41%	4%
Unfairness	9%	45%	23%	15%	9%

Challenge results for rate 3 seem to indicate that the game is *balanced*, an indicator of a successful DDA mechanism. However, the remaining participants felt a high degree of challenge. By correlating these answers with the two player categories from Fig. 4 (Table 2), we find some explanations.

This seems to indicate that participants rated the experience as more challenging due to their improved skill

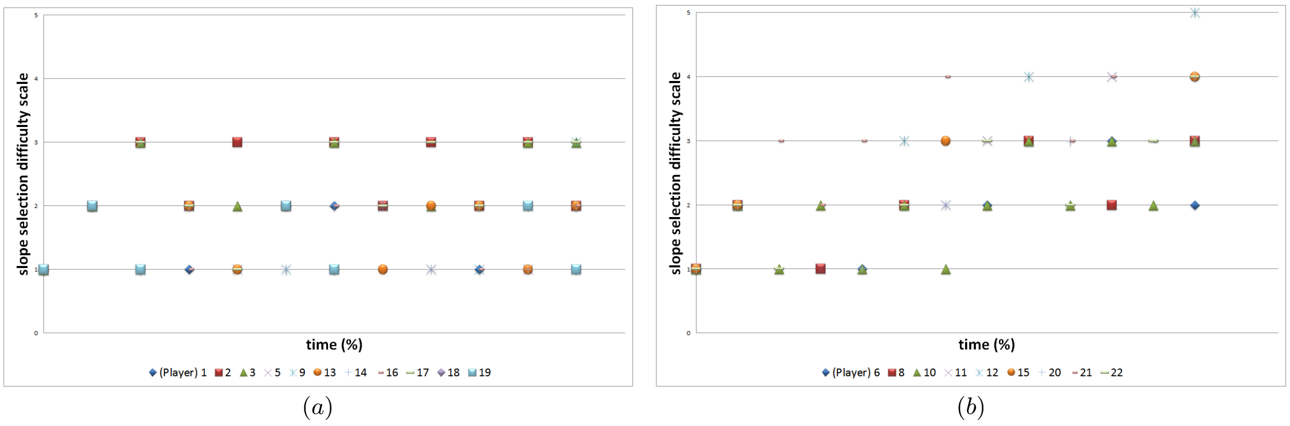


Figure 4: Variation of slope selection difficulty scale during the game session time, for all user study participants. Two categories were identified: *(a)* players without skill progression and *(b)* players with skill progression .

Table 2: Questionnaire ratings: game challenge

Players	1(less)	2	3	4	5(more)
without progression	0%	0%	73%	27%	0%
with progression	0%	0%	33%	56%	11%

Table 3: Questionnaire ratings: game unfairness

Players	1(less)	2	3	4	5(more)
without progression	9%	37%	18%	18%	18%
with progression	11%	56%	22%	11%	0%

progression and, consequently, to having reached higher difficulty scale values. Similar results on the obstacle placement difficulty scale confirm these observations.

Fairness - We asked all participants about the fairness of the game's progression. We wanted to detect whether the DDA mechanism was effective in providing the most appropriate balance, in a non-obtrusive way. Participants were asked to rate the unfairness felt throughout the game, from 1(less) to 5(more). Being children, this concept was hard to explain. Therefore, this question was posed with a more negative connotation than the challenge question, where participants would assess frustration (or satisfaction) and injustice. The answers are summarized in Table 1.

These results show that the majority of the participants found the game either fair and satisfying (rate 1,2) or balanced (rate 3). The positive/negative assessment nature of this question seems to further confirm our previous findings: again, the game seems *balanced*, an indicator of a successful DDA mechanism.

Further conclusions are observed if we correlate these answers with the two player categories from Fig. 4. As shown in Table 3, for players without progression, 64% rated the game's fairness and frustration as 1,2 or 3 (fair, satisfying and balanced). For players with progression, 89% rated the same way. This seems to demonstrate that: (i) the majority of the participants are satisfied with the game's balance of difficulty, and (ii) this satisfaction is stronger for players with progression. With these results, we observed that although gradual difficulty progression (Fig. 4b) implied a higher degree of challenge, that actually lead to a higher degree of satisfaction. This seems to indicate that the balance of difficulty to skill (DDA) was actually improved with the integration of a context-dependent difficulty progression.

Nevertheless, and since we had hoped for a balanced challenge level, we also made an effort to identify the reasons for that not to happen. We asked only the participants who identified high challenge (rate 4 or 5) to justify their answers. Although no one identified progression as the reason, 40% mentioned the new paradigm of using the accelerometer as the game controller, and 60% identified the jumping mechanics. Logged data seems to validate these answers: (i) players with jump issues died more performing jumps than everyone else (8.8 deaths to 7.6), and (ii) players with control issues died more of lack of balance than everyone else (6.25 deaths to 5.5).

Jumping seems to be a special case, deserving attention. The concerns that it raised among players essentially relate to gameplay mechanics. In the game, the snowball picks up an obstacle (see Fig. 3b) which attaches itself to the snowball. To avoid the obstacle, the player has to jump when it is approaching from behind, in the snowball, as it rolls on. Participants who identified the jumping as a challenge concern (and even others) often over-reacted by jumping as soon as they saw an obstacle on the screen, before snowball attachment. They were mimicking behavior found in classic platform games, where the player character simply jumps above obstacles. These results give valuable feedback on the design of *7's Wild Ride* jumping mechanics, which might not be the most intuitive for some casual players.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed the integration of experience-driven and context-driven PCG to support mobile adaptive game levels. We investigated this approach by implementing time-constrained level generation and DDA in the game *7's Wild Ride*, and evaluated it with two distinct user studies.

Evaluation with users allowed us to conclude that our DDA mechanism can accommodate for different players and skills. It has the potential to adjust the game to different player categories(*e.g.* with and without progression), while keeping it balanced and players satisfied. User study participants were receptive on our context-driven PCG approach, even valuing our time context-driven level generation above traditional mobile game levels, while recognizing its usefulness in the mobile games domain. Furthermore, our results allowed us to observe that the integration of context-driven PCG (specifically the gameplay progression it implied) with DDA actually lead to an increase in player satisfaction.

Some future research still remains open. More user studies could be performed to formally compare our PCG-based levels with the original designer-made fixed ones (which would involve correcting gameplay issues like jumping and adding scoring and collectibles). For the long-term, research should include the normalization between PCG-supported levels to allow some sort of common gameplay comparison (and "bragging") among different players, like with normal levels. Additionally, there are many research opportunities in other forms of context-driven PCG, using, for example, player location, local weather or time of the day to control the generator.

Anyway, we can conclude that these two modalities of adaptive PCG can work well together in accommodating some of the characteristics of mobile gaming. The casual and context-dependent nature of mobile gaming seems to naturally call for this integration of PCG approaches, and we anticipate further research on this topic in the near future.

7. ACKNOWLEDGMENTS

This work was supported by the Portuguese Foundation for Science and Technology under grant SFRH/BD/62463/2009. We thank the TU Delft Science Center for hosting our play test sessions. We thank Marlou Pors, Noeska Smit and Gerrit Rijken for their invaluable contribution to those play test sessions. We thank the CMU ETC student team responsible for creating the original *7's Wild Ride* game: Amol Deshpande, Brian Lee, Marc Mundy, Chris Thompson, and Zifeng Tian, along with co-authors Ken Hilf and Luke Jayapalan. Finally, we thank Brenda Harger of CMU ETC for her role as faculty advisor to the student project.

8. REFERENCES

[1] G. Andrade, G. Ramalho, A. S. Gomes, and V. Corruble. Dynamic game balancing: an evaluation of user satisfaction. In *AAAI conference on Artificial Intelligence and Interactive Digital Entertainement*, pages 3–8, 2006.

[2] M. B. Farrell. Apps shake up video game industry. The Boston Globe, November 2012.

[3] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment technology*, pages 429–433. ACM, 2005.

[4] 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*, PCGames '10, pages 11:1–11:4, New York, NY, USA, 2010. ACM.

[5] J. Kessing, T. Tutenel, and R. Bidarra. Designing semantic game worlds. In *Proceedings of the third workshop on Procedural Content Generation in Games (PCG 2012)*, Raleigh, NC, USA, May 2012.

[6] C. Lager. Adam Atomic on Canabalt. Gaming Daily, September 2009.

[7] 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.

[8] 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, 2012. ACM.

[9] Nielsen Company. Insights on casual games, September 2009.

[10] J. K. Olesen, G. N. Yannakakis, and J. Hallam. Real-time challenge balance in an rts game using rtneat. In *IEEE Symposium On Computational Intelligence and Games, 2008 (CIG '08)*, pages 87–94, 2008.

[11] 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.

[12] 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.

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

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

[15] M. Verma and P. McOwan. An adaptive methodology for synthesising mobile phone games using genetic algorithms. In *The 2005 IEEE Congress on Evolutionary Computation, 2005*, volume 1, pages 864 –871, sept. 2005.

[16] G. N. Yannakakis and J. Togelius. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, 99:147–161, 2011.