The effect of temporal adaptation granularity and game genre on the time-balancing abilities of adaptive time-varying minigames

Amin Tavassolian *, Kevin G. Stanley, Carl Gutwin

Department of Computer Science, University of Saskatchewan, 110 Science Place, Saskatoon, SK S7N 5C9, Canada

A R T I C L E   I N F O

A B S T R A C T

Article info

Received 2 November 2012
Revised 4 June 2013
Accepted 30 June 2013
Available online 25 July 2013

Keywords:
Game balance
Time balancing
Minigames
Adaptation

Game designers spend a great deal of time developing well-balanced game experiences. However, differences in player ability, hardware capacity (e.g. network connections) or game mechanic constraints make it difficult to balance games for all players in all conditions. Adaptive balancing systems have been employed in an attempt to automatically compensate for these differences in real time as the game is being played. However, due to the complex non-linear mechanics underlying modern games, automated balancing systems can be highly unstable for all but the simplest mechanics, restricting the design space. In prior work we advanced the concept of using adaptive minigames deployed from within a larger game to decouple the adaptive mechanics from the main game mechanics. In particular, we looked at time-adaptive minigames (ATMs) which attempt to control the time to completion of a minigame. In this paper, we extend the ATM framework with additional time-adaptation algorithms and analyze the interaction between adaptive algorithm, game mechanic, and game difficulty in a controlled experiment. We find significant effects and interactions for all three factors, confirming our intuition that these processes are important and linked. We further find that finer temporal granularity leads to less-perceptible adaptation and smaller deviations in game completion times. This work provides an empirically-grounded algorithmic foundation for the design and practical deployment of ATMs in larger games, a foundation that can improve the balance and experience in these games.

© 2014 Published by Elsevier B.V.

1. Introduction

Video games attract players with many different skill levels – from casual gamers to tournament champions. When differently-skilled players try to play the same game, the experience is often problematic for both people. Games should be balanced in terms of fairness, in that players with greater skill should usually prevail, but also in terms of competitive flow, in that the game should provide an engaging and competitive experience for all players even if they have different skill levels. Several aspects of games have been investigated as potential means for accomplishing balance (such as the available strategies for different character types, or the allocation of initial resources), but our interests lie in the use of time – that is, the amount of time needed for players to complete certain activities in the game (such as obtaining resources, building units, or moving to different locations). For game mechanics with a significant temporal component, the time taken for different activities is the most obvious way that more-skilled players differentiate themselves from less-skilled players.

When players with different skill levels play time-based games, the game can lose its flow, becoming either dull or frustrating for the players. Therefore, it is important for game designers to be able to adjust the time balance of multi-player games, without making the game seem unfair. As Rollins and Adams state in On Game Design, “you need to keep the players in the balance sweet spot for as long as is practical in order to keep the game fun and let the underdogs have a chance to catch up. [However,] the major factor that determines winners [should be] player skill” [1]. In essence, we want the best players to finish first, but by a smaller margin.

Time-based activities can be seen in many current games: in race-based games such as MarioKart, in games requiring synchronized motion between heterogeneous agents [2], in games employing rates of production such as StarCraft, and in games with ‘cooldown’ mechanics such as World of Warcraft. The time-based mechanisms and actions in these games could be manipulated to balance players of different skill levels; however, directly manipulating the time or timing parameters of these main game activities can be disruptive for the player and complex mechanics could be rendered unstable by the feedback loop created by the adaptation algorithm. An alternative approach is to manipulate time through...
activities that are outside the main game – such as through mini-
games that appear at various points within the game environment, but whose (usually simple) mechanics are different from the main
game activities.

In our previous work [3], we introduced a novel way of carrying
out time balancing through the use of adaptive time-variant mini-
games (ATMs). ATMs are simple activities contained within a larger
game that balance temporal flow by adding varying amounts of
time to a player's main-game task or mission. For example, a player
might have to complete a lock-picking minigame to break into a
building – and the amount of time taken can be controlled by
appropriate parameterization of the lock picking activities. Adap-
tive time-variant minigames provide designers with considerable
flexibility: in an ATM, the minigame is parameterized over a range
of completion times, based on the game state and player skill.
Minigames can be started as part of traditional game mechanics,
such as when a character casts a spell in World of Warcraft or
when a production order is issued to a building in StarCraft. The
minigame would then spawn as part of the main-game mechanics.
In order for the primary task to be completed, the minigame must
be completed successfully.

The ATM approach has several strengths: it decouples the bal-
cancing activity from primary game play; it allows the creation of
specific minigame-based interactions to mask the temporal adap-
tation; and it provides the designer with two primary points to ex-
tert balance in the game: the initial difficulty level (often based on
main game state), and dynamic elements of the game adjusted
during gameplay (often based on player performance in the
minigame).

Our previous work reported two studies of four different ATMs
[3]. The first study examined whether the minigames were able to
manage time correctly in isolation, and showed that both the
minimum-time prediction and the adaptation mechanisms
worked well, leading to game times that tracked the desired val-
ues. The second study tested the real-world effectiveness of ATMs
in a real mixed-reality game called Stealth Hacker. Our results
showed that the adaptive time-variant minigames were able to
provide temporal balance without detracting from the main
game. These experiences with ATMs suggested that the underly-
ing principle could be used more generally to assist designers
with time balancing in a wide variety of single-player and mul-
ti-player games.

Despite these early positive results, however, it was clear from
the prior studies that the adaptation mechanism used in [3] was
insufficient. The mechanism adapted the minigame at only one
point during game play, leading to several problems:

• The size of the adaptations was often too large as the adaptation
algorithm reacted to accumulated player error;
• The adaptations were noticeable and disruptive to gameplay;
• the ATMs achieved better player balance at the cost of a reduced
sense of fairness;
• The interaction between the adaptation mechanism, the type
of game, and the starting level could not be adequately analyzed.

In this paper we provide a much more in-depth investigation of
ATMs to address these issues. We compare three different adapta-
tion algorithms in four different game types. The adaptation algo-
rithms are:

• Discrete balance, which replicates the one-shot algorithm from
our earlier work;
• State balance, which adapts game parameters when particular
game states change;
• Continuous balance, which adapts on every game update (i.e.,
every heartbeat).

All of the algorithms employ a calibrated baseline model, an
exemplar which describes the expected progress of the player
through the minigame. The Discrete method uses the average ex-
pected completion time; the State and Continuous methods use
moment-by-moment comparisons with the expected progress
through the game as encoded by the exemplar.

To investigate their effectiveness, we carried out a study that
compared these three approaches in a 24-participant controlled
experiment. The study showed that all of the adaptive algorithms
worked well, and that the Continuous balancing technique pro-
vides the best results in balancing performance, as measured
through survey instruments and log file analysis.

Our work provides three contributions. First, we provide addi-
tional evidence that adaptive time-varying minigames are effective
tools for time balancing. Second, we show the differences between
three adaptive approaches with different adaptation granularities,
and show that the type and difficulty of the minigame can have a
substantial effect on the adaptation. Third, we demonstrate that
Continuous balancing performs best both in terms of time manip-
ulation and perceptibility. Overall, our results provide new and
valuable information for multiplayer game developers on the de-
sign, deployment, and evaluation of minigame-based techniques
for time balancing.

Having established the efficacy of ATMs for Mixed Reality
Games in [3], we have turned our attention to the construction
of the minigames themselves in this work. In particular, we are
interested in exploring the impact of adaptation algorithm on bal-
ance performance and player experience. Because the focus of this
work is on the minigames themselves, we do not provide an anal-
ysis of their integration into a larger gaming context.

2. Background: game balance and player balance

Video games are designed to generate interactive, engaging, and
entertaining experiences [4,5], and the balance of the game is
widely recognized as a design issue that has profound effects on
enjoyment [6], mutually influencing both challenge and user satis-
faction [7,8].

2.1. Balancing fairness in multiplayer games

A primary issue in competitive games is that the different teams
or players should have equal chances to win the game [1]. Balanc-
ing fairness can involve manipulations to different game elements
– for example, the capabilities and initial resources allocated to
player types such as Orcs and Humans in WarCraft [9]. This type
of balancing (called ‘static balancing’) is often carried out through
repeated playtesting of the game rules [10], such as tuning the
capabilities of individual weapons [1].

2.2. Balancing competition

One aspect of flow [4] is the degree to which a game provides an
experience for players that has an appropriate level of challenge: if
the game’s challenges exceed the player’s ability, it leads to frustra-
tion; if challenges are lower than the player’s skill, the player be-
comes bored [11]. There are three main ways that designers can
balance competition in multiplayer games (also called player bal-
ancing [12]). First, a few methods exist for balancing competition
without changing the game itself – for example, ranking systems
and ladder tournaments help match players with opponents who
have similar skill levels. Second, games can be designed so that a
stronger player is given an explicit disadvantage, such as handicap-
ing in golf or a “head-start” in playground games. In computa-
tional environments, games can also be designed with
asymmetric roles, placing the stronger player at a disadvantage. Although this strategy can be successful, the balancing mechanism is readily apparent to the players, potentially reducing the sense of fairness. Third, some games naturally evolve in such a way to make winning more difficult as the game progresses. For example, in 8-ball billiards, the leader has fewer balls to aim at, and more of their opponent’s balls to avoid [1]. Fourth, some player balancing techniques dynamically alter the characteristics of game elements during play to even out the competition. This approach was used in a version of Pong that was intended to allow parents and children to play together: the game automatically adjusted a player’s capabilities (paddle size and movement speed) based on the current score [13]. A similar capability adjustment is seen in the ‘Fatboy’ mod of Unreal Tournament, which adjusts the width of a player’s avatar based on their kill-to-death ratio, making it easier to hit better players. A third example is a system which provides differential targeting assistance using techniques such as target gravity or sticky targets [12]. The amount of assistance given to players is based on the score differential: as a player falls further behind, their targeting cursor becomes more attracted to the targets. A study of this technique showed that it increased competitiveness, and that neither the strong nor weak players noticed the adaptation [12].

2.3. Player balancing through time balancing

The dynamic player-balancing techniques described above all act on player capabilities; fewer techniques have explored adjustments to the time required for different player actions and tasks. One game genre that does frequently use time balancing is the racing genre – many racing games implement ‘catch-up’ or ‘rubber-band’ effects [12] in which a slower player receives a speed boost. For example, Mario Kart provides the ‘Bullet Bill’ power-up only to players who are far behind the leaders, which dramatically increases speed without the need to steer.

In our previous work [3] we introduced a new type of time-balancing mechanism that can be used in a wider variety of game types. This new mechanism uses adaptive time-variant minigames (ATMs) to adjust the time taken for main-game tasks that incorporate a minigame as part of the overall action. As stated in [3], “minigames are particularly attractive for time balancing because they are intended as short-duration activities, and can unobtrusively and selectively delay specific players without unduly disrupting the overall gaming experience.” [3, p.3].

Only a few projects have considered the use of minigames as balancing agents. Manhattan Story Mashup [14] used static minigames to implicitly manage game balance. Players were given a clue as a part of their ‘mission’ and were then asked to take a picture of the most related object before a timer ran out; however, the timer in the minigame was fixed. In previous work, we applied the ATM idea to the StealthHacker mixed-reality game: an informal study of this deployment found that the minigames were successful in adapting to the changing temporal requirements of differently-skilled players and role-based movement speeds [3].

3. Time balancing with adaptive time-variant minigames

Time balance in games requires that tasks be completed within a certain time-envelope distribution. Better players should be able to complete the task faster than slow players, but the variance should be modest to increase competitiveness, and the mean should coincide with the duration the designer desires. Although time manipulation can be used for many design elements in games (e.g., to artificially synchronize player action [15]) our focus here is on time manipulation as a player-balancing tool.

At a conceptual level, minigame-based time manipulation has three main steps: first, the designer must identify elements and mechanics in the minigame that affect completion time, and must determine the parameterization of those elements; second, the designer must determine which elements should adapted at the start of the minigame, and which can be adjusted dynamically during play; third, the designer must determine an adaptation algorithm and decide the frequency at which adaptation decisions will be made during gameplay.

3.1. Identification of parameterizable game elements

Many game elements can be described by parameters. For example, in Pong, the ball and the paddle can both be described by two parameters (velocity and size); changing these parameters changes the difficulty and speed of the game (e.g., a larger paddle makes the game easier). The number of points needed to win is also a game parameter that can be manipulated. As described in the related work section, substantial research has been performed on using parameter manipulation or selection for generating games of a specific difficulty [16] or for balancing player abilities [17]. These parameterizable game elements can also have varying effects on how long the game takes to play. In Pong, for example, the speed of the ball has a relatively straightforward effect on the time needed to reach a set score, but the speed of the paddle has a more complex relationship with game time, as a faster paddle allows the player to reach more shots and extend the rally, but may also increase the number of player errors.

3.2. Static and dynamic balancing phases

Minigame-based time balancing can be divided into two phases: a static phase and a dynamic phase. The static phase, which occurs before the minigame starts, sets the minigame’s parameters and mechanics to satisfy an anticipated time constraint – potentially determined by the main game state. For example, elements such as the size of the game, such as state space, game-tree size and etc., the number of levels to complete, or the starting difficulty can all be set before the minigame begins. Instantiating a game with a specific difficulty level can be achieved by solving a constraint satisfaction or optimization function with time to completion as the dependent variable [16].

In the dynamic phase, dynamic balancing is achieved by periodically comparing game state to an a priori desired state, and adjusting one or more parameters of game elements such that the completion time of the minigame will approach the desired time. There are several ways in which these adaptation decisions can be made, as described below.

3.3. Temporal exemplars

To make adaptations that deliver a particular completion time, the system must have a model of how long the minigame should take. This model can be as simple as a single completion time value (as used in previous work [3]), must be more complex if techniques such as continuous adaptation are to be used. For our study, we developed exemplar models using real-world data: we asked several people to play the minigames without any adaptation, and created a time-vs-progress model from the averaged data (see example in Fig. 1).

3.4. Adaptation algorithm and adaptation temporal granularity

The next step in time balancing with ATMs is to determine the way that adaptations will be made, and how often. The adaptive algorithm controls the type and magnitude of adaptations; we
assume that these algorithms will all compare the player's current performance to some model of desired performance. Within this general class, adaptive algorithms can still vary across several characteristics. For example:

- **Aggressiveness**: the algorithm can be more or less aggressive in correcting a disparity between the player and the ideal. For example, Bateman and colleagues noted that cautious adjustments were sometimes not able to make up a disparity within the time of the game [12].
- **Number of elements**: algorithms can change a single parameter of a single game element at a time, or can change several simultaneously. Changing multiple elements can reduce the noticeability in any one game element, but can also be more difficult to model.
- **Interaction with game narrative or appearance**: algorithms may attempt to make their adaptations less noticeable by interacting with the game narrative – a change to an element's parameter could be explained through additional narrative elements (e.g., there are more enemies near the player, which might make the adaptation less noticeable). This can also be more difficult to model.

In addition to these characteristics, the **frequency** at which adaptation decisions are made is a critical part of the adaptive algorithm. Game state adjustment could be continuous, such as adaptation decisions are made is a critical part of the adaptive process, because the granularity of adaptation can dramatically affect noticeability.

In our new adaptation algorithms, we investigate two methods of balancing time continuously: state-based and timer-based, and compare these methods with both a one-shot adapter (as used in previous work), and with no balancing at all. Each of these balancing algorithms is described below.

- **No Balance**: players play the game without any time adaption. Parameters are set at the start of the game, and are held constant throughout.
- **Discrete (One-Shot) Balance**: players play the game with the starting parameters until a preset duration is exceeded, then a single immediate increase in parameters occurs to enable completion of the task. This setting is analogous to the one reported in [3] except in this work the time threshold is set to the desired completion time rather than the minimum required completion time.
- **State Balance**: a player's performance is compared with an exemplar every time a particular game state changes (e.g., a subtask is completed). Balance is recalculated every time the state changes, based on the player's performance relative to the exemplar.
- **Continuous Balance**: a player's performance is compared with an exemplar at regular intervals, and balance is recalculated based on the player's performance relative to the exemplar.

## 4. Four example minigames

As in our previous study, we used four minigames to test the efficacy of our time balancing algorithms: Puzzle, Electris, Click and Hack and BrickOut (shown in Fig. 2). Each game has both static and dynamic balancing mechanisms as described above. While the primary purpose of this contribution was to evaluate the dynamic balancing algorithms, we also evaluated two static balancing settings (deployed as two difficulty levels) for each game to ensure that the dynamic algorithms' performance was not specific to a given starting configuration.

In addition, the games were redesigned to incorporate gameplay elements into the new adaptation algorithms. The adaptations in speed were smoothed in some games, and a difference threshold was added to prevent adaptation to small changes in performance. The appearance of Brickout and Electris was altered when the speed changed, in order to integrate the adaptation into the simple narrative of the minigame. Specifics related to the realization of each adaptive algorithm, details of the visualization changes, and the static conditions studied are described for each of the four games in the following sections.

All the games were implemented in C# using the XNA framework, on a Microsoft Windows 7 PC. Continuous updates were tied to the XNA game heartbeat which updates every 16.7 ms. No balancing action was taken in the game unless the difference between the player's performance and the exemplar exceeded 1 s. To mask the speed changes in Puzzle and BrickOut, the speed changed linearly over a 2-s period when adaptation was required.

### 4.1. Click and Hack

Click and Hack is a variant of the fairground game “Whack-a-mole,” players must click on the “Hack” button and then quickly click on a computer image that appears at a seemingly random location on the screen (Fig. 2). Click and Hack is essentially a Fitts' Law task [19], where the difficulty of the challenge is proportional to the size of the target and the distance from the Hack button.

#### 4.1.1. Static and dynamic elements

The static balancing mechanism is the number of targets that must be clicked to complete the game. The combined distance-size tradeoff – generally termed the index of difficulty – is the dynamic balancing mechanism.

#### 4.1.2. Dynamic adaptation method

In [3] we used target size as the adjustable parameter, but in the version reported here we use distance. During normal game play, the computer can appear anywhere on the screen. When the player is progressing much faster than the exemplar we draw targets from a distribution biased to provide more distant targets. When the player is slower than the exemplar we draw targets from a distribution biased to provide closer targets. In the Discrete algorithm, once the target completion time is exceeded, all targets are preferentially drawn from the close distribution.
4.1.3. Experimental purpose of game

Click and Hack is distinct from the other games because the Continuous and State systems are largely equivalent in that the update parameter is a discrete state rather than a continuous variable like speed. Continuous balancing cannot update faster that State because the update parameter is the same as the parameter measured by the State algorithm.

4.2. Puzzle

In the puzzle game, players must align a series of disks to make a continuous path from a computer to server. There is only one solution, so the game poses a similar gameplay challenge to a physical geometric puzzle.

4.2.1. Static and dynamic elements

The static balancing mechanism is the number of pieces in the puzzle. The dynamic balancing mechanism is the rotational speed of the pieces.

4.2.2. Dynamic adaptation method

In all cases the speed begins at 12°/s. For the Discrete algorithm, puzzle pieces increase to 18°/s once the target completion time is exceeded. In the State case, the speed is recalculated every time a player moves a disk to the correct location. The number of recalculations for the State algorithm is therefore equal to the number of disks (assuming that players do not move a correctly placed disk to an incorrect position). The Continuous algorithm measures the game state every 16 ms and updates the rotation speed at the same rate if necessary.

4.2.3. Experimental purpose of game

Puzzle is distinct from the other games because of the significant difference in the update timing for the State and Continuous cases.

4.3. Electris

Electris is a variant of falling brick games such as Tetris or Bejeweled. Electrical components fall from the top of the screen down a well, changing type sequentially with every downward step. The player must match a particular electric circuit, Mastermind-style, that is shown at the top of the screen. Like most falling brick games, the bricks fall at a set rate from top to bottom. While the component falls, the player can move it left and right, and can commit the component by pressing spacebar, which causes the component to stop cycling and fall at a faster rate.

4.3.1. Static and dynamic elements

The primary static balancing mechanism is the number of rows which must be completed. The primary dynamic balancing mechanism is the speed at which pieces fall after the spacebar is pressed.

4.3.2. Dynamic adaptation method

In the Discrete case, pieces fall at a rate of 120 pixels/s until the target time is exceeded, and then pieces fall at a rate of 360 pixels/s. To intertwine the adaptation with the narrative, the color of the background varies (red for fast, green for neutral, blue for slow) to indicate the falling speed for the piece in adaptive cases. For State-based adaptation, the piece speed is set based on the number of correct pieces placed when compared with the exemplar. If an incorrect piece is placed, the number correct goes to zero because the entire line is now wrong. In the Continuous case, piece speed is recalculated as the piece falls, but the speed does not change until the player commits the piece.

4.3.3. Experimental purpose of game

Electris is distinct from the other games we studied because it has a steep cost associated with failure. Once a component has been played it cannot be removed or altered. An incorrectly played
component thus negates an entire row requiring the player to finish filling the row so they can begin a new row on top.

4.4. BrickOut

In BrickOut, the player must guide a bouncing ball such that it hits a series of bricks at the top of the screen. Bricks disappear when struck, and the game is complete once all the bricks have been eliminated.

4.4.1. Static and dynamic elements

The static balancing mechanism is the number of rows of bricks. The dynamic balancing mechanism is the speed of the ball.

4.4.2. Dynamic adaptation method

The ball moves at a slow, medium or fast speed depending on the performance of the player with respect to the exemplar. The ball color changes when the speed is adapted, using the same scheme as the Electris background (described above). In the Discrete case, when the time exceeds the target, the ball always moves at its fastest speed. In the State algorithm, completion time is compared with the exemplar every time a brick is hit, and the speed of the ball is changes on the rebound. In the Continuous algorithm, the speed of the ball is continuously and gradually adjusted based on the difference between the current expected completion time and the exemplar.

4.2.3. Experimental purpose of game

BrickOut represents a baseline for the other styles of adaptation because the brick count (and therefore number times back and forth, or total distance traveled) and ball velocity represent the most direct mapping to total time as the ratio of distance and speed.

5. Experimental methods

The study involved two main phases. In the first phase, we recorded empirical data to build our exemplar models, as described in Section 3.3. In the second phase, we tested the different adaptation approaches with a new set of participants.

5.1. Exemplar models

Prior to the main study, eight volunteer participants were recruited to play each of the game conditions. We averaged the performance of the eight players within each game for the exemplars, where ‘performance’ was defined differently for the different games: time per disk in Puzzle, time per brick in Breakout, time per targeting action for Click and Hack, and time per piece for Electris. The performance data was used to create time-vs-progress models similar to those of Fig. 1.

We note that we are not limited to using empirical data as a baseline, and in fact, the shape of the time-vs-progress curves provides a significant design opportunity for managing the feel of a game.

5.2. Participants and apparatus

For the main study, we recruited 24 test subjects (12 male and 12 female, average age of 27 years) from the university community. Participants were all experienced with mouse-and-windows software, and had a wide range of experience with video games (18 played games rarely, 5 played regularly, and 1 played frequently). The study was carried out in a controlled environment, on a standard Windows 7 PC with a 1920 x 1080 screen. Custom minigames were run full-screen, and were all controlled with a standard two-button optical mouse. The study software recorded all performance measures; questionnaire data was gathered using paper forms.

5.3. Minigames and adaptation conditions

Participants played two versions of each of the four minigames described above (Click and Hack, BrickOut, Electris, and Puzzle). One version had an ‘easy’ starting difficulty, and therefore a lower expected completion time, and one version had ‘medium’ difficulty and a longer expected time. The specific starting values for easy and medium were dependent on the type of game, and are shown in Table 1.

Participants played the eight different minigames (four game types and two difficulty levels) under the four different adaptation approaches described above (No balancing, or Discrete, State, or Continuous balancing).

5.4. Procedure

Each player was provided with an informed consent form in keeping with university ethics policy, and briefed on the different games and process of the study. Players were told that we were testing different game configurations, but not what the differences between the conditions were or the ordering of the conditions. The players then played the eight minigames shown in Table 1 (four game types and two difficulty levels) under each of the four conditions. Players saw all of the conditions within each game (i.e., starting difficulty and adaptation algorithm) in a different order, based on a Latin square design.

After every game, participants were given a short questionnaire to determine their play experience and their impression of the perceptibility of the time-balancing algorithms in each case. Game state, and in particular all parameters associated with the time-balancing algorithm were logged. Once the participants had completed all game conditions, they were given a final questionnaire on their experience and a brief demographic survey.

5.5. Design and analyses

The study used a factorial within-participants design, with three factors:

- **Adaptation Algorithm:** None, Discrete, State, Continuous
- **Difficulty:** Easy or Medium starting difficulty
- **Game:** Click and Hack, Electris, Puzzle, BrickOut.

Presentation order of the games, and presentation order for the difficulty and adaptation conditions within each game, were balanced using Latin square designs. The main dependent measures were game completion time and game performance (progress over time was also recorded for the adaptation mechanism and is also used in our analysis).

Timing data gathered from computer logs were analyzed with three-way ANOVA tests; post hoc tests were conducted using

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Games types and two difficulty levels for each.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Click and Hack</td>
</tr>
<tr>
<td>Puzzle</td>
</tr>
<tr>
<td>Electris</td>
</tr>
<tr>
<td>Breakout</td>
</tr>
</tbody>
</table>
Tukey's HSD. Survey results were analyzed using Friedman's ANOVA for related samples. For all tests, \( \alpha \) was set at 0.05.

6. Results

The study was designed to investigate two main issues:

1. Accuracy in managing completion time:
   - Q1: are the adaptive approaches more accurate than the non-adaptive condition?
   - Q2: which adaptive approach is most accurate?
   - Q3: does game type or difficulty level affect accuracy?

2. Player experience:
   - Q4: Were there differences in the players' perception of the different adaptive approaches?
   - Q5: Did differences adaptation alter the players' enjoyment of the game?

The results reported below are organized around these issues and research questions.

6.1. Overall completion time

The ultimate goal of ATMs is to provide games designers with the ability to deploy situation-dependent time-balancing minigames from within a larger game, and maintain tight control over the minigame completion time by using dynamic adaption to drive individual player's performances towards an exemplar. Fig. 3 shows the completion time distributions for all conditions.

As is apparent from the graph, differences between game starting difficulty and adaptation algorithms exist. This is desired, as the game challenge will depend on the type of game, and game difficulty forces a significantly longer completion time by design. Within each game category, time-base adaptation is usually minimum in time and variance, and the no adaptation case is usually maximum. The exception is Puzzle 5 which was dominated by a few notable outliers in the State and Continuous cases, where completion time was dominated by the difficulty of the puzzle, not the speed of the disks. In all cases the quartiles (represented by the extent of the box) and the 95% confidence interval (represented by the whiskers) are smallest for the Continuous case, indicating that player performance more closely adhered to the exemplar.

6.2. Accuracy in managing completion time

Accuracy was determined by subtracting the completion time for each different game from the desired time indicated by the exemplar model; this provides an error amount for each minigame. Given the four adaption scenarios we examined – None, Discrete, State and Continuous – we should expect to see adaptive cases converge toward the exemplar, and the no-adaptation case depart from the exemplar. Given the nature of the games, we should also observe differences in the interaction between the balancing algorithms and the time to completion between each game and level.

ANOVA showed significant main effects of all three primary factors on error amount (Algorithm: \( F_{3,69} = 14.67; \) Game: \( F_{3,69} = 48.27; \) Difficulty: \( F_{1,23} = 16.13, \) all \( p < 0.001 \)). A summary of mean error amounts for these factors is shown in Fig. 4.

Our primary interest in following up these main effects is in exploring differences between the different adaptive algorithms (Q2). A Tukey HSD test shows that there are significant differences between the algorithms (all \( p < 0.05 \)): all of the adaptation conditions had significantly lower error amounts than the non-adaptive condition, and the Continuous algorithm had significantly lower error than Discrete and State; no other differences were found.

The ANOVA test also showed significant interactions between Algorithm and Game \( (F_{9,207} = 7.20, p < 0.05) \), and between Algorithm and Difficulty \( (F_{3,69} = 4.19, p < 0.05) \). Fig. 4 summarizes these differences; as the figure indicates, the different algorithms performed differently on different games and difficulty levels. In particular, all algorithms performed better on Click and Hack than on the other games; and for some games (Puzzle and BrickOut), differences between the algorithms were larger with the more difficult starting conditions, whereas for others (Electris), the differences were larger with the easy version of the game.

Based on these results we conclude that adaptation does have an effect (Q1), and that the choice of adaptation algorithm does affect accuracy, with Continuous having significantly lower error...
amount than other approaches (Q2). However, these results depend to some degree on both the type of game and the difficulty level (Q3).

6.3. Player performance under adaptation

Considering the differences of current adaptation algorithms, we were interested in the influence of each adaptation on specific player performance. Fig. 5 shows the error times (Actual completion times – baseline exemplar time) of players for all of the adaptation algorithms for puzzle (medium), and Brickout (medium). Fig. 6 shows the performance and exemplar of a single example player for the same pair of games.

Each graph shows the absolute error performance of an individual participant for the given game-level combination. Players are sorted by completion time in the no-adaptation case independently. Several notable outliers are evident in the State adaptation case for the Puzzle game. These outliers are primarily due to feedback effects and the low frequency of State updates in the Puzzle game, which only calculates balance once a disk has been correctly positioned. Players who perform particularly well on a particular piece are unduly punished with a speed reduction on the next piece, potentially dramatically increasing completion time. This performance oscillation is evident in Fig. 6, which shows the game performance for a single player overlaid on the exemplar. In the State case, small oscillations in the exemplar and player performance feed back upon each other to drive increasingly larger swings in performance, culminating in a final completion time substantially slower than the exemplar or the Continuous case. In BrickOut (Fig. 6) player performance follows the exemplar more closely, except for the Discrete-balancing case. The Discrete algorithm shows a marked departure from the exemplar near the end of the game. This performance lag was due to the player missing the last brick, and having to bounce the ball back and forth over the width of the screen and back again to achieve the correct angle to strike the final brick and end the game, demonstrating that while adaptation can drive the player performance distribution towards a desired shape in aggregate, individual player performance still matters for the outcome of the game.

6.4. Player experience

The experiment established that minigames have useful properties for the parameterization of adaptation. However, appropriate balancing is of little utility if the adaptation algorithm destroys the game experience. To investigate the effects of different
balancing techniques on player experience, we gave participants questionnaires after playing all conditions for a single game and at the end of the session.

A Friedman ANOVA of the responses indicated no significant differences in level of fun between the different games, indicating either that adaptation algorithm did not affect player enjoyment of the game, or that our test was insufficient to find the differences (Q5). We asked participants to score the fun level of the game out of 5; which 1 represents “Not Fun” and 5 represents “Very Fun” and results showed that minigame were generally fun in every situation while the State Balance was viewed as the least desirable. As it is shown in Fig. 7, Out of 24 participants, 69%, 60%, 59% and 46% have mentioned that the No Balance, Discrete, Continuous and State Balancing techniques were either “Fun” or “Very Fun”

We also wanted to determine the relative perceptibility of the adaptation algorithm in each game. At the end of the experiment participants were asked “Did you notice a difference in the game mechanics between the four versions of <Minigame>?” and “Did you notice that the game mechanics in <Minigame> would change based on your performance in the game?” The yes/no responses to these questions are plotted in Fig. 8A and B.

The majority of participants noticed a difference in game mechanics between the four cases, although this is possibly due to the appearance changes in the games when adaptation was employed.

Participants were also asked after each game condition to comment on whether they noticed any changes within the game. The question was kept intentionally vague to avoid biasing the within-subjects design. For all the games, only three participants (for Continuous and State) and five participants (for Discrete) responded affirmatively. Most of those responses commented on the change in game appearance. No respondents noted that the speed change seemed to be tied to their performance. In the Continuous case participants actively stated that changes in mechanic were unrelated to their performance. For example for the Continuous cases:

“The background of Electris changes all the time during the game but it wouldn’t affect my performance.”

“The colour changes are fine, but seem to coincide with speed reductions in the parts of the game I do not control.”

“The color change in the middle of the Electris game does not have any significant meaning, and was initially misleading.”

These comments are distinct from feedback for the State cases where all players noted that the change in display was related to the speed of the game, indicating that the larger, less frequent speed changes in the State case were more noticeable.

“In Electris: I think the Idea that the background colors changed was not bad but the speed kept changing too... made it less predictable.”

“The blue is a nice touch, though it seems to indicate slower gameplay, so I found myself looking forward to the red.”

“I liked the change of ball color in the BrickOut game, perhaps it indicated the speed of the ball.”

Based on the survey results and participants’ comments, we conclude that while some players noticed the change in the dynamic adaptation parameter, none perceived that it was tied to their performance, indicating that dynamic adaptation was not noticeable in the experiment (Q4).
7. Discussion

This paper makes several contributions to the design and engineering of adaptive game mechanics. We confirm the hypothesis proposed in our earlier work [3] that subtler and more accurate adaptation algorithms are possible, and provide an initial analysis into tradeoffs between time balancing and game design. In particular, we demonstrate the connection between adaptation granularity and perceptibility, and show that individual differences are still preserved even when time balancing is being carried out.

Our evaluation of the adaptation methods provides six main results about minigame-based time balancing:

- all of the adaptive algorithms were more effective than the non-adaptive condition in manipulating minigame completion time;
- the Continuous algorithm was significantly more accurate than all other algorithms, and State-based balancing was more accurate than Discrete;
- the Continuous algorithm had the lowest standard deviation of all algorithms;
- some changes to game parameters were noticed by participants, but people did not notice the connection between the changes and their performance;
- the more frequent adaptation algorithms (Continuous and State) appeared to be less noticeable overall;
- the adaptive methods did not reduce participants’ subjective level of fun.

7.1. Explanations for main results

7.1.1. Accuracy of the adaptive methods

The results of the study show that the adaptive algorithms performed well, and their success is a basic confirmation of our initial premise in this research – that the simpler mechanics of minigames can be analyzed and understood to the point where manipulation of completion time is possible. Both the overall data and the in-depth examinations of player progress (e.g., Fig. 5 and 6) show that the algorithms were effective in recognizing divergence from the desired time, and effective in altering the games to shift the player’s time toward the exemplar.

7.1.2. Differences between game types

The adaptation methods performed differently for the different games. In Click and Hack, there was very little difference between any techniques, including no adaptation at all. In this case we believe that game time is so well described by the underlying Fitts’ Law model that setting the static initial parameters may be enough to provide a particular time value. In contrast, time error in Electris was much larger and more different across the different algorithms. In this game, the gameplay follows a much less linear path than Click and Hack, primarily due to the effects of making errors. These examples indicate that the complexity of the game mechanics play a large role in the behavior of dynamic balancing algorithms.

7.1.3. The value of more-frequent adaptation

Making adaptation decisions more frequently (as in the Continuous and State algorithms) was less perceptible and caused fewer oscillations in the players’ performance. The significant oscillations evident in some cases (e.g., for some players in Electris) suggests that effective time balancing may only be possible in simple games such as minigames. More complex games (i.e., most main games) have much more sophisticated mechanics, and are likely to exhibit non-linear behavior when adaptations is introduced. The substantial feedback effects noted in some cases here, despite the simple mechanics, highlight the difficulty of adaptation in more complex systems. By constraining the adaptation to games with simple linear dynamics we reduce the risk of complex and difficult-to-control behaviors disrupting game balance.

7.1.4. Retaining individual differences

Our study also showed that employing an adaptive algorithm does not remove all variability from the games – as stated earlier, it is important to provide competitive balance but without negating the effects of player skill or game design. In the study results, there were larger variances between games and between difficulty levels than there were between algorithms. This is desirable because it demonstrates that the adaptation is not the dominant factor in determining completion time, and that designers have freedom to create the timing profiles they desire by appropriately choosing the game and difficulty level prior to instantiation. It is also worth noting that the Continuous algorithm did not disrupt game timing when the players’ performance was near the

Fig. 8. Perceptibility of adaptation algorithms (A). Perceptibility of game mechanics (B).
exemplar. Overall, the completion times still formed a distribution (albeit with significant variation in mean, and variance between games), with means driven towards the desired values specified by the exemplar.

7.1.5. The performance of the continuous adaptation

As mentioned earlier, one major step in time balancing using ATMs is to decide how the adaptation algorithm should be carried out, for example in terms of type and magnitude; therefore, one important question is the frequency at which the adaptation algorithm should be invoked during game play. Since in each type of adaptation – Discrete, Continuous and State – the current state of the game is compared with a previously calculated exemplar, the frequency of the comparisons can affect the final result of the adaptation. With this in mind, we might conclude that the Continuous adaptation is the best at any situation.

The performance of an adaptation algorithm depends on other parameters such as game type, player type, frequency of adaptation and underlying game mechanics. Moreover, each algorithm should be hidden from the player's perspective while managing the game-completion time. In fact, an adaptation algorithm that performs accurately is not necessarily the most desirable if it interferes with player experience. For example, consider an adaptation method which compares the elapsed time with an exemplar and when reaches a certain time, finishes the game suddenly. This method is accurate because the total completion time of the game will be exactly as specified, but the heavy-handed manipulation could destroy the gameplay experience for the player.

Although the main goal of this research is to investigate the role of temporal adaptation granularity and game genre in time balancing capabilities, we implicitly consider the players' game experience. While it is a reasonable hypothesis that a higher frequency adaptation leads to improved accuracy, it is still necessary to evaluate the players' experience. In this study we asked 24 participants to play the 4 minigames with different configurations to determine the perceptibility of adaptation algorithms and game mechanics, and consequently to see how enjoyable these algorithms are for players.

The adaptive algorithms were evaluated with respect to accuracy of completion time, and player experience. Player experience was further divided into the enjoyment of the game (fun), and the perceptibility of the adaptation and adaptive mechanic. As Fig. 3 shows, the Continuous adaptation is the most accurate method among all other adaptation algorithms we used. Moreover, it deviated least from the desired completion time of all the algorithms. We found that more frequent operation of the mechanism dampens oscillation in players' performance (Fig. 6) and consequently leads to smaller variations in game mechanics and is therefore less perceptible to the players (Fig. 8).

Since Continuous adaptation compares the current progress of the game with a previously obtained model, there should be a model with a sufficiently high temporal granularity and measurement accuracy to serve as a baseline. Generating these models usually requires time and energy because they consider all the effective elements of the game that impact the total completion time. In this research we used an exemplar that was derived from empirical data which may be prohibitively expensive in some commercial games.

7.2. Application and deployment

Minigame-based time balancing can be employed in any game in which there are obvious breaks in pacing where a minigame can be inserted. This is often done in current mainstream games in an unadaptive way with quirktime events, where the primary gameplay mechanic is suspended and replaced with a rhythm/pattern-matching mechanic. While a more fulsome examination of the applicability of this approach is the subject of future work, we provide an initial discussion here of the applicability of the minigame time balancing mechanic to two general types of game interactions: races, and action timing.

In race games, time is the final mediator. Whoever completes the challenge fastest – be it solving a puzzle, building a structure or navigating a maze – is the winner. Significant attention has been paid to providing balanced outcomes in racing games, from subtly increasing the top speed of the loser, to providing context sensitive power-ups based on position. Minigames could be used to help provide timing balance if the primary game mechanic provides for a break in the race. This could be a pit stop in a car-racing game, a locked door in a maze racing game, or the scheduled discipline switches in a triathlon. This type of timing intervention is analogous to the Stealth Hacker mixed reality game we discussed in our previous work [3]. In this case, designers should use games like Puzzle to maximize potential control over the timing, or Electrics to provide an additional time penalty for poorly executed minigame play.

While racing games are based on time-to-completion, many other games, such as first person shooters (FPSs), real time strategy (RTS) and roleplaying games (RPGs) are based to a large extent on relative rates, such as damage-per-second (DPS), or power-to-build-time tradeoffs. Minigames could be spawned during changeover events, such as reloading a weapon or casting a spell to replace the fixed cooldown timers that are explicitly (for an MMORPG spell) or implicitly (through a reloading animation) rendered in existing games. This could be integrated into the game as an additional exercise in skill: players that can cast spells or reload their weapons faster would have a DPS advantage. Because of the tight timelines imposed by these small cooldown timers, designers would likely want to opt for low mean, low variance minigames such as Hack a Computer to add small amounts of balance to regularly repeated actions, rather than large mean and variance games suitable for infrequent actions. It is easy to imagine a direct variant of Hack a Computer in an MMO context where mystic ruins would appear and players would have to click them in order to complete the spell.

Time balancing is appropriate wherever relative timing – particularly between multiple players – is an underlying mechanic of the game. However, interfering directly with core mechanics, particularly in complex and delicately balanced systems such as RPGs, could lead to troubling non-linearities in the response and unexpected outcomes of game behavior. While this underlying mechanic exists in many games (for example a classic MMORPG could be considered a carefully balanced set of cooldown timers) the problems of temporal balance are particularly acute in mixed reality games (MRG). In a MRG, the virtual portion of the game must be carefully balanced with the portion which takes place in reality [3]. Pacing in the virtual game must be carefully managed to reflect changes in physical reality outside of the designers' control.

On the other hand, ATMs are short, self-contained play experiences that have their own logic, game state and mechanics and could be, in principle, designed to be integrated into many types of games. Because estimating the completion time of an ATM is tractable given its limited state-space it could be integrated into other gaming genres. As future work, we intend to integrate ATMs into other types of games to determine their overall impact and acceptability, which will allow us to compare the utility of ATMs in different game genres.

7.3. Limitations and future work

The limitations of this paper relate to the relative youth of multiplayer balancing algorithms in general, and time-balancing
algorithms in particular. We highlight four primary shortcomings and the future work required to address them.

1. Scope: We have only tested a fraction of the proposed balance methodologies described in [3] which are in turn only a subset of all the possible minigame balancing mechanics. While our research demonstrates the feasibility of the approach, fertile ground remains for examining the breadth of applicability and generalizability of the concept. In particular, many game mechanics are based on psychometric principles (e.g., movement, memory-based recall) that have well-studied models, and could be used to provide a better understanding of how particular kinds of game elements can predict completion time. Our observation of the results has led us to the preliminary conclusion that the complexity of the underlying game mechanics has an impact on the performance of the game. This conclusion manifests itself both in the primary mechanic of the game and in the cost of failure, which can depend both on game mechanic and game state. Breakout has a well described adaptive mechanic where the speed of the ball is directly mapped to the time differential from the exemplar. When there are many bricks in play the dominant factor is the speed of the ball, as any successful deflection by the player is likely to lead to a destroyed brick. In contrast, during the endgame, the dominant factor is player aim, as deflections which do not successfully destroy bricks may result in sequences of useless reflections as the player attempts to reposition the ball to a point where a hit can be attempted again. Anecdotally, it appeared that increased ball speed was a detriment in these cases, because it made the delicate final aiming tasks more complex.

In future we will further investigate the impact of adaptive algorithm on the game as it progresses to determine if the changing effects of adaptation in Brickout was anecdotal or an actual statistical effect. We will also investigate how non-linearities in the game design, such as the cost of creating an incorrect line in Electris, impact the overall balance of the game. Of particular interest is the possibility of introducing what appear to be random benefits or inhibitions – similar to the powerup distribution in MarioKart – to overall balance and player experience.

2. Breadth: We focused our analysis on the adaptation mechanics, and did not examine the impact of integrating the minigames within a larger gaming context. This simplification was undertaken to control for overall game engagement. Given that we have already established the viability of the integrated approach in [3], this was a reasonable experimental methodology. Future work in this area involves consideration of how minigames can be designed to fit into the overall narrative of the main game, and how timing requirements can be identified within the main game and used as the initial conditions of the minigame.

3. Sample Bias: As with any experiment involving human subjects, there is the possibility for sample bias. We cannot conclude that our findings will hold for all players equally; in fact it is reasonable to hypothesize that competitive gamers would be more sophisticated at spotting small adjustments in game mechanics than the dedicated but not elite gamers studied here. Broader studies with different games and demographics could extend the results.

4. Baseline: We have only examined two simple variants of the exemplar model. In [3] we examined the case where the timing profile was entirely defined by the designer as the minimum required completion time. Here we examined the entirely empirical case where desired average time was based on play-tester performance. In the future, we expect more sophisticated exemplar variants based on the synthesis of designer intuition and empirical metrics garnered during playtesting and by mining play logs after game deployment.

8. Summary and conclusion

In this paper we examined the relative impact of adaptive algorithm, game type, and difficulty for adaptive time-balancing minigames. All three minigame components had a significant effect on the completion time and balance for the game, and showed strong interactions. We found that a continuous time-based update strategy, coupled with design techniques meant to integrate or mask the adaptability led to average completion time tending toward the desired value, while minimizing player disruption. We discussed how these techniques could be integrated into a number of gaming genres, including the popular Racing, RPG, RTS and MMORPG genres. In the future, we will investigate additional minigame mechanics, more sophisticated adaptation algorithms and the integration within larger gaming contexts. This work represents a strong foundation for the continued research, development and deployment of time-adaptive minigames.

References