Time Balancing with Adaptive Time-Variant Minigames

Amin Tavassolian, Kevin G. Stanley, Carl Gutwin, and Aryan Zohoorian

Department of Computer Science, University of Saskatchewan
110 Science Place, Saskatoon, Saskatchewan, Canada, S7N 5C9
kstanley@cs.usask.ca

Abstract. Balancing timing of tasks and abilities in multiplayer games is an important design element, but two time balancing issues are currently difficult to deal with: individual differences in experience or skill, and real-world elements that impose fixed temporal constraints on the game (as in mixed-reality games). We introduce *adaptive time-variant minigames* as a way of addressing the problems of time balancing. These minigames are parameterized to allow both a guaranteed minimum play time (to address fixed temporal constraints), and dynamic adaptability (to address temporal variances caused by individual differences). We developed three adaptive time-variant minigames and carried out two studies with them. The studies showed that the adaptation mechanisms allow accurate prediction of play time, that the minigames were valuable in helping to balance temporal asymmetries in a real mixed-reality game, and that they did not detract from the overall play experience.

Keywords: Game balance, time balancing, minigames, adaptation.

1 Introduction

Game balance is an important aspect of every game and the one that has received considerable attention from researchers and game designers [1]. There are several types of balance that need to be considered in a multiplayer game – such as balance between different strategies [2], balance between the capabilities of different teams [3], or balance between players of different abilities [4]. One main resource in many balance considerations is *time* – that is, the time required for certain activities (such as moving to a new location in the game, building a structure, or training a unit) is a resource that can be used in balancing against the resources of the other team either explicitly in terms of time or implicitly in terms of capabilities.

Some issues of time balancing can be dealt with during the design of the game (e.g., ensuring that faster units are less powerful), but two particular situations cannot be completely solved in design, leading to temporal asymmetries that must be addressed at run time. First, individual differences in experience or skill mean that two players will take different amounts of time to complete particular tasks; this situation affects a wide variety of multi-player games. Second, some games – e.g., *mixed-reality games* [5], [6] – involve aspects of the real world that impose fixed temporal constraints. For example, the amount of time it takes for a player to run from one game area to another is determined by the size of the real-world game space, and

J. Anacleto et al. (Eds.): ICEC 2011, LNCS 6972, pp. 173–185, 2011.

cannot be changed in the design of the game. If the game space is fixed, the only way to balance time is to make the players faster, which is not often possible.

In this paper we introduce a novel way of addressing the problems of time balancing, through the use of *adaptive time-variant minigames* (ATMs). Minigames are simple activities contained within a larger game, and are common in commercial titles (e.g., *Mario Party*, *Sid Meier's Pirates!*, or *Assassin's Creed 2*). Minigames can help designers balance temporal aspects because they can add time to a player's main game task or mission; however, traditional minigames have fixed difficulty, and may not provide the timing control that designers require. Adaptive time-variant minigames provide additional flexibility: in an ATM, the minigame is parameterized over a range of completion times, based on the game state and players' skill.

ATMs have inbuilt mechanisms to support both configuration of the minigame to set minimum and expected times when launched, and adaptation of minigame mechanics and parameters to influence completion time during play. The time needed for the various elements of the minigame (based on factors like speed or number of levels) can be determined when a game instance is launched. A guaranteed minimum time can be calculated using a simple formula, while the expected and maximum time can be determined empirically. The expected average time and desired variance is addressed by dynamically adjusting variables (such as the speed at which a ball moves, or the distance to a target) to increase or decrease completion time.

To test the effectiveness of ATMs as tools for balancing time, we developed four different minigames and carried out two studies using these games. The first study examined whether the minigames were able to manage time correctly in isolation, and this study showed that both the minimum-time prediction and the adaptation mechanisms worked well, leading to game times that tracked the desired values. The second study tested the real-world effectiveness of ATMs in a real mixed-reality game called *Stealth Hacker*. Although this investigation was a limited trial, our results show that the adaptive time-variant minigames were able to provide temporal balance without detracting from the main game. Our experiences with ATMs suggest that the underlying principle can be used more generally to assist designers with time balancing in a wide variety of multi-player games.

2 Prior Work

Unlike other types of media, video games are designed to generate interactive and engaging experiences [7], and game balance is recognized as a design issue that has profound effects on enjoyment – mutually influencing challenge and user satisfaction [8], [9]. Previous methods of managing the game complexity and balance, such as tuning the difficulty of static pre-defined levels, are often labor-intensive [10]. In fact, maintaining optimal game balance often needs to be a dynamic process because of the evolution of the player's behavior and skill [11]. If the game's challenges exceed the player's ability, it leads to frustration, and if challenges are lower than the player's skill, the player becomes bored [12]. Several previous approaches focus on the game's AI to address to dynamic balancing, but neglect the timing component [13], [14], [15], [16]. Bateman et al. [4] divide game balancing into gameplay balancing and player balancing for player's differing in skill and effort level.

Mixed-reality games, which incorporate real and virtual play simultaneously, face particularly acute time balancing issues. Generally, the physical portion of the game relies on existing infrastructure such as buildings, roads, and bridges, and is difficult to modify; similarly, the behavior of real world participants is dictated by physics and human physiology and cannot be altered. The majority of time balancing must therefore take place in the virtual portion of the game. Several approaches have been proposed to balance mixed-reality games. For example, online players may play on a scaled-down representation of the real playground with speeds adjusted proportionally to be appropriate for this scale [5]. *NetAttack* [17] divided players based on their roles and balanced play, but not timing based on role. In *Manhattan Story Mashup* [18] static minigames have been employed 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 within a cooldown timer, but the timer in the minigame is fixed, and variations in skill or the surrounding context do not change the duration.

Most solutions to time balancing in MMRs have presumed that virtual interfaces are point-to-point mapped to the real world – that is, that virtual players play in simulacrums of the real playground. Timing is implicitly addressed by setting virtual locomotion speeds to be approximately equivalent to expected real world locomotion speed [19]. While straightforward and easy to implement, this assumption is overly limiting and constrains the design space for MMR games.

3 Adaptive Time-Variant Minigames

Minigames are generally short, self-contained play experiences within a larger game framework, but with their own internal logic, game state, and mechanics. Because minigames have their own internal mechanics, they can be configured independently of the main narrative or action, making them an attractive alternative for dynamic balancing because the initial and goal game states can be made contingent on the overall game state without disrupting that game state when the minigame ends. 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.

Minigames, as games, have several parameters that can be manipulated to speed up and slow down the game duration. The simplest example involves controlling the scope of a repetitive task, such as shooting asteroids or aliens, where the number of times the task must be repeated changes. Another simple example is the manipulation of game physics (or physics analogues) to increase or decrease the speed of active components: for example, increasing the speed of falling bricks in Tetris can allow faster completion times because the blocks cross the screen faster. In principle, many game states could be parameterized, but in the minigames implemented for this paper we limit ourselves to the 'count' and 'physics' components described above.

To test the idea of using minigames as context-sensitive and player-sensitive time balancing tools, we implemented four minigames based on canonical arcade, board and puzzle games. The games are set in a 'computer hacking' theme, which is consistent with the larger game in which these minigames are used (see section 3.2).

- Hack-a-Computer: Inspired by the classic fairground game *Whack-a-Mole,* this game requires the player to click on a computer when it appears on the screen, then quickly return to the base of the screen to press the "Hack" button (Fig 1.A).
- Spinning Puzzle: This game is inspired by the picture puzzles from *Assassin's Creed 2*, where puzzle pieces rotate about a central axis. Players must rotate the pieces until the correct image resolves (Fig 1.B).
- Electris: Like the arcade classic *Tetris*, players guide falling electrical components to match the circuit train displayed at the top of the page. Unlike Tetris, successful rows do not disappear, making the size of the board the number of possible attempts, like the classic board game *MasterMind* (Fig 1.C).
- Brickout: This game is *Brickout*, pure and simple. Players must guide a ball using a paddle to destroy lines of bricks at the top of the screen. (Fig. 1.D).

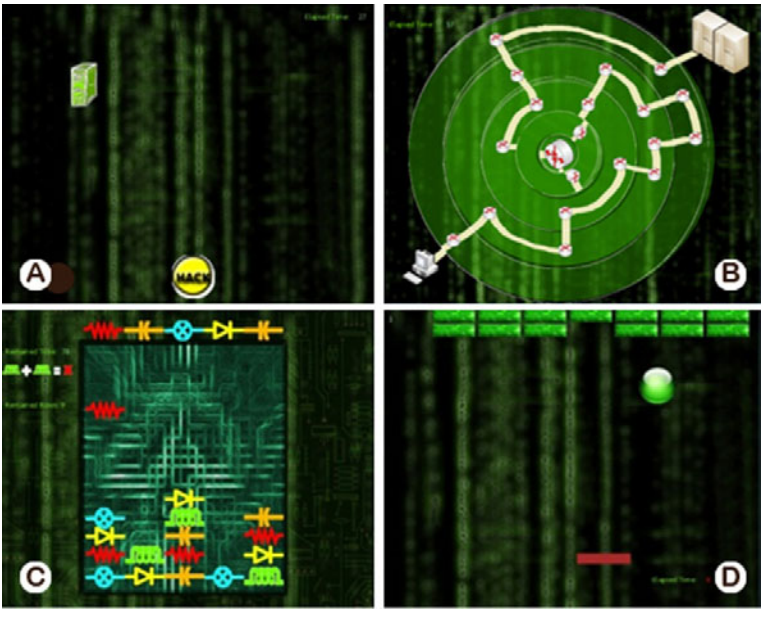

Fig. 1. Minigames: A. Hack-a-Computer; B. Spinning Puzzle; C. Electris; D. Brickout

3.1 Timing Algorithm

There are two classes of balance we are manipulating through the minigames: an overall balance based on game state, possibly set a priori, and a dynamic balance based on player ability set during play. By building games which have a configurable minimum completion time, a disguised countdown timer can be implemented. Timing balance can be enforced by minimum completion times, and adapted to expected or desired completion times. While an entire taxonomy of games and time varying manipulations is possible, we present only the proof of concept games here.

Table 1 lists the games, the manipulable parameters, the parameters used to fix minimum time, the parameters used to adapt playing time and the formula for

minimum time. In principle other components such as play area could be utilized as manipulable parameters, but we only list those parameters actually implemented. In the formulae of Table 1, x_i is distance in pixels, h_i is the vertical screen dimension, v is the component velocity (ball, piece, etc.), θ_i is rotational distance and ω is rotational speed. N and n correspond to count variables corresponding to the number of elements such as pieces, bricks or targets.

Table 1. Minigame parameters and formulas

Game	Manipulable Components	Initial Settings	Adaptive Component	Minimum Time
Hack-a-Computer	Mouse speed, target size, number of targets	Target size, # of targets	Target size	$\sum_{i=1}^{n} \frac{2x_i}{v}$
Electris	Piece speed, number of lines	Number of lines	Piece speed	$N \sum_{i=0}^{n} \frac{h_i}{v}$
Spinning Puzzle	Rotation Speed, number of disks, min. # turns required	Number of disks	Rotation speed	$\sum_{i=1}^{n} \frac{\theta_i}{\omega}$
Brickout	Number of bricks, speed of ball	Speed of ball	Number of bricks	$\sum_{i=1}^{n} \frac{2h_i}{v}$

The initial settings column in Table 1 corresponds to the scale of the minigame created when the game is instantiated, configured each time the game is launched but static over the duration of a single minigame. This parameterization is meant to compensate for quasi-static temporal asymmetries. Temporal asymmetries due to player differences must be fixed at run time, and are addressed using the Adaptive Component in Table 1 which functions like a dial that can alter the speed of a game based on the player's current performance. For example, the speed of each subsequent component falling in Electris could be subtly altered to make the game proceed faster or slower during play.

Because we are able to express the minimum completion time in a closed form, we can find the minigame and initial game state that best fits the situation at run time, adding a degree of flexibility to game balancing. While the minimum time is often explicitly calculable, there is a practical upper limit to the maximum time based on player interest and game design. For example, it is possible to delay a player for thirty seconds using the Hack-a-Computer minigame, but would be disruptive to force them to play for ten minutes. Minigame selection and configuration proceeds as follows:

1. A game is selected based on its guaranteed minimum time and practical maximum time. This operation could be performed during level design, as part of the overall balancing, or dynamically based on the overall game state.
2. The base difficulty of the game is set by changing the manipulable parameters to achieve the desired minimum and expected time. Again this can be set a priori as part of game design or dynamically with game state.

3. Once a set time has elapsed (often but not necessarily the minimum time) the game can adapt the speed of the manipulable parameters to compensate for differences in player ability. This must occur during game play.

To evaluate the efficacy of the minigames and approach, we implemented a mixed reality game, *Stealth Hacker*, which pits a single hacker, playing on a computer opponent against a team of security personnel playing in the real world.

3.2 MMR Platform: Stealth Hacker

Stealth Hacker is a mixed-reality location-based game inspired by the playground game *Cops and Robbers*, played with several cops and a single hacker. The shared playground is a network of computers which the hacker attempts to infiltrate. The cops navigate this playground physically, moving from computer to computer and scanning them with smartphones (Fig. 2.A). The hacker, fittingly, moves from computer to computer virtually, by navigating a simple avatar around a network diagram (Fig. 2.B). The movement speed of the hacker, fitting the narrative, is on the order of seconds, providing the feeling of zipping across the network from computer to computer. Cops, in contrast, transit from computer to computer on foot, with elapsed times on the order of tens of seconds, fitting the Newtonian physics which governs motion in the real world. This asymmetry of spatial representation and navigation speed creates an interesting timing dichotomy: in the real world, the cops predominantly spend time moving between nodes, but spend little time at each node, while the hacker can transit between nodes quickly, and therefore must predominantly spend game time at network nodes to maintain time balance.

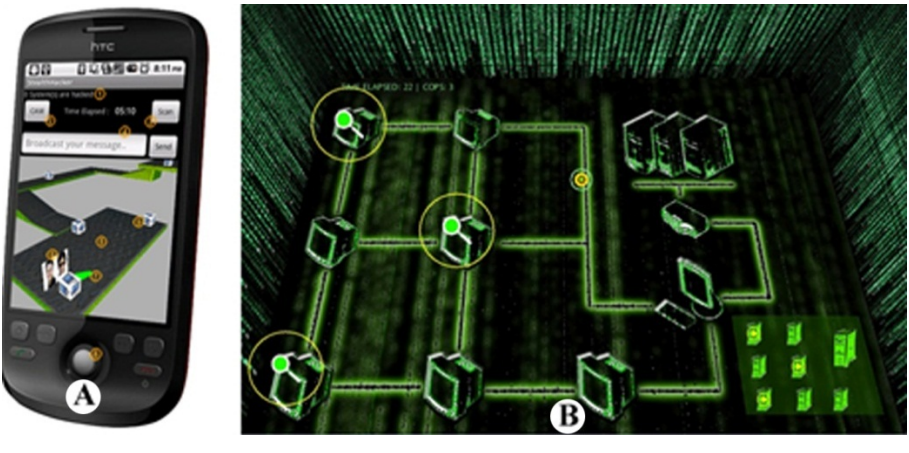

Fig. 2. A. Cop's interface on a smartphone, B. Hacker's PC interface

The cops' interface provides them with information on the location of their partners (both current location and planned movements), the last known location of the hacker, and a chat interface. When the cops scan a computer the program records the computer's Bluetooth MAC address, and transmits it to the server wirelessly. The scanned computer does not actually contribute anything other than its Bluetooth

address, because the game state is managed by the server, and the hacking and scanning are simulated as minigames on the smartphones or the hacker's PC. The hacker tries to hack every computer in the network. Minigames are launched when the hacker attempts to infiltrate a computer and provide dynamic balance through guaranteed minimum and expected mean and maximum times at each node.

In *Stealth Hacker*, one of three minigames (Hack-a-Computer, Electris, or Spinning Puzzle) is allocated to an individual node when the hacker arrives and attempts to break in to the computer at that node. The game choice and its initial complexity are based on the average real-world distance from the attacked computer to the two nearest cops, based on an estimated foot speed of 4.95 ft/s [20]. The appropriate equation in Table 1 is used to calculate parameter settings that will provide a target completion time that matches the estimate of the cops' time (from the distance and speed heuristic). Once the hacker has played the minigame for the minimum time, the game adapts to allow the hacker to finish whatever tasks remain as quickly as they are able, by increasing the speed parameter listed in Table 1.

The Brickout game is given a special role. It is triggered if the hacker is caught by one of the cops. Catching the hacker occurs if a cop arrives at the same physical location as the hacker's virtual location, and 'scans' the computer. For every cop that 'scans' the hacker during a single instance of the Brickout game, an additional row of bricks appears, making the game more difficult to complete before a timer expires. If the hacker completes the game before a timer expires, they escape back into the network. If the hacker fails to complete the minigame, they are captured and the cops win. The minimum game completion time for Brickout, as set by the ball speed, is slightly longer than the average physical transit time between any two physically adjacent nodes in the network. Well-organized teams of cops therefore have the chance to 'gang up' on the hacker by ensuring that reinforcements are sufficiently close. This special case demonstrates that minigames can be tuned to manipulate game balance based on user input as well as initial game state.

4 Evaluation

4.1 Methods

Prior to evaluating the minigames in situ, we evaluated their time characteristics in isolation, to determine whether the chosen adaptive parameters did in fact improve player performance. Fifteen participants (10 male and 5 female, aged 22 to 33) played all the minigames. Players trained on each minigame at each difficulty level once to reduce training effects, and then further played each minigame once again for every difficulty level both with and without adaption. Difficulty levels for each game are shown in Table 2. The experiment ran on a Dell 6500 laptop (Intel Core 2 Duo, 2.53 GHz) with a 15-in 1800x1200 display, and standard keyboard and mouse.

Table 2. Experimental conditions for first study

Game	Difficulty Parameter	Value of Parameter
Spinning Puzzle	Number of rings	4,5,6,7,8
Electris	Number of rows	1,2,3
Hack-a-Computer	Number of targets	10,20,30,40,50

To test the ATMs in situ, we ran 9 rounds of the Stealth Hacker game described in section 3.3. MacBook computers with known Bluetooth MAC addresses were used as the physical nodes. These computers were placed over two floors in the University of Saskatchewan Computer Science Department. The maximum distance between two computers was 71.85 m and the average distance between two computers was 35.9 m. Four male players (Computer Science students mean age 27) played the game. One round of the game was played without data recording and with coaching from study organizers to familiarize the players with the game mechanics. Each player played six rounds as a cop and two rounds as the hacker. All game events and digital player-player communications were logged. After completing all nine rounds players filled out a demographic and experience survey.

4.2 Results 1: Evaluation of Minigames in Isolation

This study was intended to verify that time of completion is controllable through adaptive minigames, through *a priori* and dynamic adaptations. We found an almost linear dependence on difficulty level for all three games: completion times were faster with adaptation, and these curves were also linear with difficulty, albeit with a smaller slope. Results for the three tested games are shown in Fig. 3.

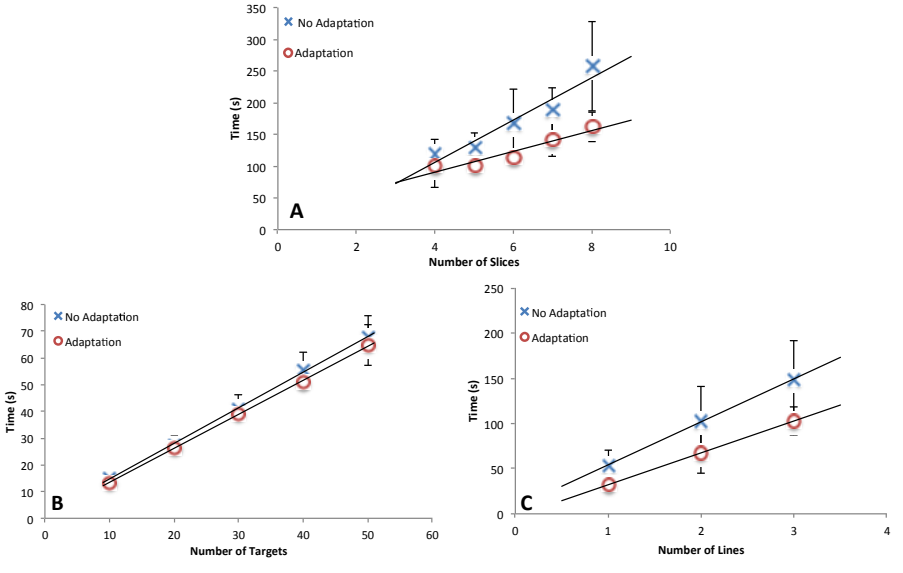

Fig. 3. Comparing average baseline and adaptive mode game play time: A: Spinning Puzzle. B: Hack a Computer. C: Electris

These results demonstrate three important properties of our adaptive minigames: first, they have linearly increasing mean time of completion with difficulty; second, there is a game-dependent decrease in completion time with adaption; third, the means and variances of each of the games are significantly different. The first result

provides conveniently constant increases in completion times with difficulty. The second result demonstrates that dynamic adaptation provides a useful difference in completion times. The final result demonstrates both that player-player differences exist, and that games can be selected for not only completion time, but the variability of that completion time and the degree to which dynamic balancing aids the player.

4.3 Results 2: Evaluation in the Stealth Hacker MMR Game

The minigame calibration experiment established that the games had the properties of linearity and uniqueness, but mathematical properties alone do not make for viable games. For the approach to provide value it must dynamically adjust the challenge of each game to fit the current game state in the Stealth Hacker MMR game. We observed three outcomes: that players had positive experiences playing the game as the hacker, that the game remained balanced in opportunity if not outcome, and that the minigame timing reflected the game state at the time of instantiation.

To evaluate the subjective user experience during play, we administered a survey. Players felt that the minigames were fun (mean rating 3.6 out of 5), and added to the overall game (4 Yes, 0 No) which was also seen as fun (mean rating 4.25 out of 5). We also asked participants to rate the percentage of time they spent playing minigames (mean 62.5%) which was substantially less than the value measured from the logs (mean 79.8%), which could indicate time dilation effects related to flow [7], another indication that players were absorbed by the minigames.

We examined the balance of opportunity and outcome by determining the number of hacked systems per game and plotting an annotated node occupancy diagram for one of the shorter games played. The 'number of hacked systems' metric is a measure of overall balance because if the number is too small, it indicates that the hacker had little chance of winning; if too large, it indicates dominance by the hacker. The hacker hacked all seven systems 3 times, winning the game, but still managed to hack at least 3 and an average of 5.75 systems in the 5 losses. The dynamic timing balance achieved by the adaptive minigames is shown for a single game in Fig. 4.

In Fig. 4, the dark boxes represent the hacker playing a minigame and the numbered light boxes represent the three cops. The y-axis is the node location (one of the seven computers). Early in the game the cops were nearby the hacker, and the minigame engine spawned three relatively easy games. Once the hacker moved to a more distant node, a much more difficult game was spawned, which the hacker successfully completed. In the final game, a more difficult game was also spawned, as the cops were initially far away, but rapidly converged on the location of the hacker and trapped him with three consecutive rows of Brickout, shown by the occupancy of location 3 at the end of the game.

To verify that the minigame duration reflected the game state as the game evolved, we considered Fig. 4, which shows that in a single instance at least, the cops were often proximate to the hacker, while the hacker played the minigame. However, a single game does not provide compelling evidence of efficacy.

Fig. 5 shows every minigame played over all 8 conditions. The estimated average time for two cops to reach the hacker, closely tracks the minimum calculated game time demonstrating that our techniques have sufficiently high temporal resolution to capture highly variable game states. The actual time of completion follows the

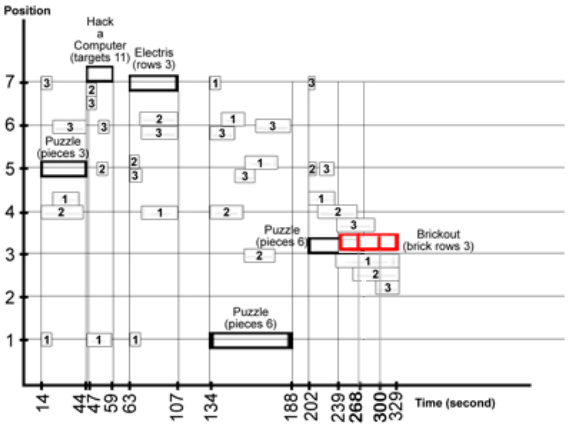

Fig. 4. Player locations and actions in a single Stealth Hacker Game

minimum values and shows variability both within and between subjects demonstrating the techniques provides game balance control without artificially limiting the game, by still allowing for player expertise and chance to play a role.

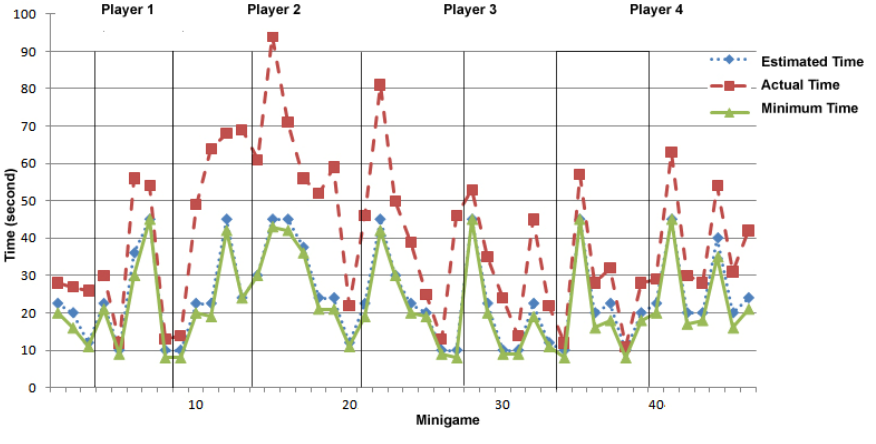

Fig. 5. Minimum completion times for all minigames in the experiment

5 Discussion

Our evaluations provide evidence for the efficacy of adaptive time-variant minigames as a mechanism for balancing time. Our first study showed that the minimum and expected times were predictable, and our second study with a real mixed-reality game showed that the minigames were enjoyable, and provided the balancing effects for which they were designed. Here we discuss further issues of generalizability, and possible improvements to the adaptation capabilities.

What other types of ATMs are possible?

The idea of adaptive minigames can be applied much more widely than just the example systems demonstrated here. The core elements of the approach involve analyzing the time requirements for each game mechanic in the minigame, determining how the game can be parameterized to control completion time, and designing an adaptive algorithm for responding to run-time events.

This process is applicable to a wide variety of game types. For example, a search minigame (e.g., *Where's Waldo*) involves visual search as the main game mechanic. The time needed for visual search is a function of the number of items that must be searched, and the time needed to evaluate each item, allowing parameterization of items to the visual differences between the target and the distracters.

Many possible game mechanics can be considered: signal detection and discrimination, pattern matching, aiming, pursuit tracking, short-term memory, spatial memory, and choice reaction tasks. The timing profiles of some mechanics have been modeled (e.g., Fitts' Law, Hick's Law, the Keystroke Level Model), permitting the use of existing models as starting points for analyzing and parameterizing minigames.

The time needed for minigame tasks involving cognition (e.g., calculation, reasoning, or mental rotation) will be more difficult to predict and will be subject to greater individual differences, and so are less useful for use within an ATM. However, even cognitive tasks could be modeled using empirical testing – that is, a mean time and a distribution around that mean can easily be found by asking a sample group to play the game during design and testing.

More complex adaptation

The minigames described here used a relatively simple mechanism for run-time adaptation – once the minimum time had been reached, players were allowed to complete the game at a high speed – but more complex strategies are possible. Adaptations should predict the actual time that a player will take with a given minigame, and also to be able to respond quickly to dynamic events. These goals must be balanced with the need to maintain a reward structure in the minigames – that is, there must be a reward for being skilled or exerting more effort in the minigame.

The dynamic time adaptation algorithm presented here was somewhat crude, but accepted by players. A more elegant solution would employ a slowly ramping, but still predictable dynamic speed increase so the player was less likely to realize that game parameters were being altered. In a more sophisticated variant, empirical testing could be used to record 'partial completion times' for the tasks in the minigame. In this approach, groups are asked to play the minigame and the time they need to achieve different milestones within the task are recorded (e.g., 90% of players might reach level three within 45 seconds). This information can then be used to adapt minigames for new players at a much finer level of granularity, exerting potentially tighter control over the completion times, while still maintaining a small step size.

6 Conclusion

In this paper we described a novel approach for balancing timing in multiplayer games using adaptive time-variant minigames. We balance timing in two steps: first, we draw from a set of minigames designed to have theoretically guaranteed minimum

completion times and that have linear difficulty / time-to-completion relationships in practice. These games are then tuned dynamically as the game is being played to attempt to reach a desired expected completion time. We demonstrate that these games can provide a compelling experience for balancing a mixed-reality game, a particularly difficult time-balancing problem since computer players must be balanced against those in the real world. In the future, we hope to explore the potential for balancing other game genres using this mechanism, and to examine different adaptive mechanisms including continuous feedback and predictive modeling.

Acknowledgements. We would like to thank our participants and NSERC and GRAND-NCE for funding.

References

1. Lankveld, G., Spronck, P., Rauterberg, M.: Difficulty scaling through Incongruity. In: Fourth Artificial Intelligence and Interactive Digital Entertainment Conference. Association for the Advancement of Artificial Intelligence (2008)
2. Vogiazou, Y., Raijmakers, B., Geelhoed, E., Reid, J., Eisenstadt, M.: Design for emergence: experiments with a mixed reality urban playground game. JPUC 11(1), 45–58 (2007)
3. Starner, T., Leibe, B., Singletary, B., Pair, J.: MIND-WARPING: Towards creating a compelling collaborative augmented reality game. In: IUI 2000, pp. 256–259 (2000)
4. Bateman, S., Mandryk, R.L., Stach, T., Gutwin, C.: Target Assistance for Subtly Balancing Competitive Play. In: CHI 2011, pp. 7–21 (2011)
5. Benford, S., Crabtree, A., Flintham, M., Drozd, A., Anastasi, R., Paxton, M.L.: Can You See Me Now? ACM ToCHI 13(1), 100–133 (2006)
6. Schneider, J., Kortuem, G.: How to Host a Pervasive Game Supporting Face-to-Face Interactions in Live-Action Roleplaying. In: UbiComp 2001 (2001)
7. Sweetser, P., Wyeth, P.: GameFlow: A Model for Evaluating Player Enjoyment in Games. ACM Computers in Entertainment 3(3), Article 3A
8. Andrade, G., Ramalho, G., Gomes, A., Corruble, V.: Dynamic game balancing: an evaluation of user satisfaction. In: AIIDE (2006)
9. Humble. R.: Inside EverQuest. Game Developer Magazine, 18–26 (May 2004)
10. Boll, S., Krosche, J., Wegener, C.: Paper Chase Revisited – A Real World Game Meets Hypermedia. In: Fourteen ACM Conference on Hypertext and Hyoermedia, pp. 126–127 (2003)
11. Andrade, G., Ramalho, G., Santana, H., Corruble, V.: Automatic computer game balancing: a reinforcement learning approach. In: AAMAS 2005, July 25-29 (2005)
12. Martínez, J., Delgado-Mata, C.: From Competitive to Social Two-Player Videogames. In: ICMI-MLMI 2009 Workshop on Child, Computers, and Interaction, Cambridge, USA (2009)
13. Demasi, P., Cruz, A.: Online Coevolution for Action Games. In: Intelligent Games and Simulation, pp. 113–120 (2002)
14. Spronck, P., Sprinkhuizen-Kuyper, I., Postma, E.: Difficulty Scaling of Game AI. In: Intelligent Games and Simulation, pp. 33–37 (2004)
15. Andrade, G., Ramalho, G., Santana, H., Corruble, V.: Extending Reinforcement Learning to Provide Dynamic Game Balancing. In: IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games (2005)

16. Hunicke, R., Chapman, V.: AI for Dynamic Difficulty Adjustment in Games. In: AAAI Workshop on Challenges in Game AI, pp. 91–96 (2005)
17. Lindt, I., Broll, W.: NetAttack – First Steps Towards Pervasive Gaming. ERCIM NEWS Special Issue on Games Technology 57, 49–50 (2004)
18. Tuulos, V., Scheible, J., Nyholm, H.: Combining web, mobile phones and public displays in large-scale: Manhattan story mashup. In: LaMarca, A., Langheinrich, M., Truong, K.N. (eds.) Pervasive 2007. LNCS, vol. 4480, pp. 37–54. Springer, Heidelberg (2007)
19. Benford, S., Flintham, M., Drozd, A., Anastasi, R., Rowland, D., Tandavanitj, N., Adams, M., Row-Farr, J., Oldroyd, A., Sutton, J.: Uncle Roy All Around You: Implicating the City in a Location-Based Performance. In: Advances in Computer Entertainment (ACE 2004). ACM Press, New York (2004)
20. Aspelin, K.: Establishing Pedestrian Walking Speeds. Portland State University (2005)