

Using “Tilt” as an Interface to Control “No-Button” 3-D Mobile Games

PAUL GILBERTSON, PAUL COULTON, and FADI CHEHIMI

Infolab21, Lancaster University, UK

and

TAMAS VAJK

BUTE, Budapest, Hungary

Mobile phones offer considerable challenges for game developers, and not least among them is the user interface, which is primarily optimized for number entry rather than for playing games. In fact, due to the limitations one of the most desirable criteria for mobile games has the design of games controlled by a one-button interface. However, this type of interface has only been seen as applicable for casual games, where mastering the interface is de-emphasized. As a number of mobile phones are starting to appear with 3-D accelerometers, game developers have the opportunity to investigate new interface mechanisms. In this article we illustrate how accelerometers provide the possibility of a no-button mobile game. While 3-D accelerometers offer a range of possible interface mechanisms, the one that requires minimal signal processing and no external references is *motion*, and in particular, *tilt*, and as such is eminently suitable for mobile phones. In this article we explore a tilt interface for a 3-D graphics first-person driving game titled *Tunnel Run*, and compare the user experience playing the same game with a traditional phone joystick interface and with a tilt interface in two different modes. The results show that the tilt interface was experienced as fun, and certainly seemed more attractive to players, who said they would not have played this type of game otherwise.

Categories and Subject Descriptors: D.3.2 [Programming Languages]: Language Classifications

General Terms: Experimentation, Design

Additional Key Words and Phrases: Mobile, Java, C++, mobile phone, motion sensors, games, Symbian, J2ME, usability

ACM Reference Format:

Gilbertson, P., Coulton, P., Chehimi, F., and Vajk, T. 2008. Using “tilt” as an interface to control “no-button” 3-d mobile games. *ACM Comput. Entertain.* 6, 3, Article 38 (October 2008), 13 pages. DOI = 10.1145/1394021.1394031 <http://doi.acm.org/10.1145/1394021.1394031>

Authors’ addresses: P. Gilbertson, P. Coulton, and F. Chehimi, Infolab21, Lancaster University, Lancaster, LA1 4WA, UK; email: {p.gilberston, p.coulton, f.chehimi}@lancaster.ac.uk, T. Vajk, Department of Automation and Applied Informatics, BUTE, Budapest, Hungary; email: tamas.vajk@aut.bme.hu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org. © 2008 ACM 1544-3574/2008/10-ART38 \$5.00 DOI = 10.1145/1394021.1394031 <http://doi.acm.org/10.1145/1394021.1394031>

ACM Computers in Entertainment, Vol. 6, No. 3, Article 38, Publication date: October 2008.

1. INTRODUCTION

User input has always played a major part in the development of games, and has recently entered a new innovative phase via WiimoteTM on the WiiTM. Indeed, companies such as Nintendo are increasingly seeing control as the most important aspect in user engagement. In an interview with Satoru Iwata on the Wii launch website, Akio Ikeda, who was responsible for the accelerometer hardware on the Wii, said

“Of course, when playing a game, the nearest thing to the player is the controller. The controller should therefore be regarded as an extension of the player rather than as part of the console. I always bear in mind the importance of the fact that the player will have far more contact with the controller and UI (user interface) than the console itself.”

While these comments were obviously made in relation to a console, the importance of interface is particularly relevant to mobile phones.

This is because mobile phones have an interface predominantly designed for number entry, and few have made any accommodation to the needs of game players. This has led to the emergence of the one-button game concept [Nokia 2006]. In a one-button interface the player does not need to focus attention on the controls or on the keypad but on the gameplay and the graphics [Wigdor and Balakrishnan 2003]. In these games the challenge does not come from mastering the interface, as in many console games, but rather from precise timing, fast reactions, and the rhythm of actions. Achieving a good rhythm or completing a series of difficult actions precisely on time is a key element in a good one-button game experience. However, the limitations of the one-button paradigm mean that not all game genres can be accommodated, particularly the 3D driving or first-person shooters (FPS) genres. This has led us to consider alternative forms of interface, hence in this article we address the emergence of 3D accelerometers on phones.

While the accelerometer offers a range of possible uses, being the simplest to implement and requiring minimal signal processing and no external references, it is undoubtedly the features of *motion* or *tilt*¹ that make it eminently suitable for implementation on mobile phones. Furthermore, tilt games in which manoeuvring a ball or marble within a maze have been around for many years, so the interface is familiar as a game input mechanism. A computer-generated example of such a game by Havok Xtra is shown in Figure 1.

Using tilt input from in-built accelerometers as an interface to mobile phone applications has been proposed in the past [Bartlett 2000; Fishkin et al. 2000]; however, these studies are related to conceptual or prototype devices that often bear little relationship to an actual mobile phone, and are primarily concerned with navigating around in the phone or as an alternative to text entry [Wigdor and Balakrishnan 2003], rather than an interface for playing games.

While using accelerometers for tilt-sensing is the most obvious and reliable method, there are alternatives available on mobile phones via the camera [Coulton et al. 2007]. The two alternatives fall largely into two categories: the

¹Defined by the Oxford English dictionary as to incline or tip.



Fig. 1. Marble maze game.

Fig. 2. The *Tunnel Run* game.

first where simple optical flow techniques are used to track the motion of the phone, such as employed by GestureTek™ for their EyeMobile™ engine; and the second where visual codes are used to provide a point of reference for movement [Bucolo et al. 2005]. However, using the camera can be problematic due to variations in optical quality, and particularly problematic in low-quality light levels, whereas 3-D accelerometers are more reliable and offer finer granularity.

Despite our emphasis on the fact that accelerometers are a better technical solution than using the camera, there are only a few phones on the market that contain 3-D accelerometers; we have some way to go before this technology is widespread. However, we believe this technology will become widespread and more devices will emerge, such as the Nokia 5500 used in this project, and more recently the Nokia N95 and N82.

In terms of games implemented using tilt on mobile phones, there have been numerous implementations for the camera and one for the Nokia 5500 using accelerometers, all of which were based on the marble types of games previously described. While these games are no doubt fun, they replicate the existing game input mechanics of the original games, and in our research we wanted to explore tilt in relation to other game genres. Therefore, for this project we implemented a 3D graphics first-person driving game called *Tunnel Run*, shown in Figure 2.

As tilt is not the natural output of 3-D accelerometers, in the following section we describe how tilt detection was ascertained. We then discuss the design



Fig. 3. Using gravity to ascertain tilt direction.

and implementation of the *Tunnel Run* game itself, which presented a number of technical challenges that are worth exploring. Finally, we will present results from our user experience trials in which different interface controls for the same game were compared. These user trials were built on previous ones for version 1.0 of the game [Gilbertson et al. 2007], which identified a number of issues regarding feedback and operating position. Finally, we explore the future of motion-controlled mobile game interfaces before drawing our overall conclusions on the practicalities of designing games using a tilt-based input.

2. USING ACCELEROMETERS TO ASCERTAIN TILT

Using tilt is the equivalent of using the earth's gravity field as a 1-g reference acceleration along the vertical (Z) axis (depending on location, g is approximately 9.8 m/s^2). The X and Y axes will each be at 0 g when the device is perfectly level. Figure 3 illustrates how this applies while using the accelerometers for the mobile phone in this research project, although it is valid for all such devices.

When the player tilts the controller, the game engine needs to ascertain if the amount of tilt has crossed a particular threshold. The method to measure tilt is an inherent trigonometric relationship. The X and Y outputs of an accelerometer as a function of the tilt angles, θ_x and θ_y (the angles that the X and Y axes make with the horizontal), are proportional to $g \sin \theta_x$ and $g \sin \theta_y$. In a marble maze type of game, where the player is trying to imitate the effects of real gravity, it's not necessary to know the actual angles (calculated by inverse trigonometric functions); the device's outputs physically model the forces affecting virtual marble movement. In this game we use the amount of tilt to control the speed of a racer in a particular direction.

3. GAME DESIGN AND IMPLEMENTATION

The choice of target platform when designing a mobile game is of paramount importance in terms of both sophistication and distribution [Coulton et al. 2007], and can have a huge effect on the game that is finally produced. Currently there are a number of platforms for developing mobile games, such as Symbian, J2ME, BREW, C#, WidSets Widgets, and Flash Lite, although in terms of market penetration J2ME and Symbian dominate game development [Coulton et al. 2005].

As previously discussed, our target device was the Nokia 5500, which supports games developed in J2ME, Symbian (S60²), WidSets Widgets, and Flash Lite. Flash Lite and WidSets Widgets were quickly discounted for this particular project due to their lack of 3D graphics support. In terms of accessing the 3-D accelerometers, the only solution was Symbian Sensor API, which is available from Nokia and requires the use of the Symbian S60 third edition SDK, which is similar in function to J2ME's Mobile Sensor API JSR-256, although, at the time of development there were no mobile phones that included JSR-256.

However, this did not rule out J2ME, as this problem was overcome by using a socket connection on the mobile phone to allow access to native services. The solution is achieved by opening a low-level socket connection in a S60 C++ application and then connecting to the defined port on the loop-back address from the J2ME application [Vajk et al. 2008].

J2ME was favored over Symbian for a variety of reasons, the first being the wider number of devices that support the Java platform, which means that a version of *Tunnel Run* with more standard controls could also be developed for commercial deployment. Further, the mobile 3D graphics API for J2ME (JSR-184) (M3G) offers better support for 3D gaming compared to OpenGL ES [Chehimi et al. 2006], which is implemented for Symbian, as it contains scene graphs and fast retained mode rendering support. Finally, the ease of rapid development when using J2ME and M3G warranted the exploitation of this unusual method of accessing the accelerometers.

Once the target platform and 3D library were defined, the 3D art assets could be developed. M3G supports a specific binary format developed for the platform M3D and an API for the construction of meshes from raw vertex data in source. This binary format contains a complete scene graph with camera, object, and lighting nodes. There are a number of tools available for the creation and export of M3D files, and for this project we utilized Blender, which supports a third-party Python plug-in for exporting scenes to both M3D and Java source code. For *Tunnel Run*, the Java source was chosen over the binary format to allow for simpler manipulation of meshes and the scene graph. In hindsight, this simplicity came with a penalty, as the speed of the API is much slower than loading a single M3D file.

On a resource-constrained device such as a mobile phone, it is vital to save resources, so in the following paragraphs we will highlight some of the choices made regarding 3D which could be used to achieve this aim.

²S60 (formerly Series 60) is a version of the Symbian OS for smart phones and has been developed by Nokia.

The actual number of art assets required for *Tunnel Run* is fairly low, as a simple tunnel was created out of three flattened cubes as shown in Figure 2. The floor cube extended slightly deeper than the two wall cubes, so that when two tunnel sections were connected end to end, it created a small gap in each wall. This means that as a player “travels” along the tunnel, these gaps provide the illusion of speed as the racer passes along the tunnel. Simple cubes provide the obstacles, and the player’s racer is represented by a flattened arrow-shaped solid.

For simplicity, the scene is lit by only two lights, coupled with a small amount of ambient light that provides sufficient lighting to all sides of the tunnel, the obstacles and the player, thus making the game easier to play. Shading is provided by a directional light located up and to the right of the player. The light position is locked relative to the player mesh, and moving the lamp with the player means that there is only one lighting node in our scene graph. Due to the simple vertex-based shading performed by the engine on meshes with large faces, the light casts an interesting pattern on the tunnel floor, which again heightens the sensation of speed.

The initial setup of the mesh is not a trivial task on a mobile phone, as each mesh requires the manipulation of vertex and index buffers, applications of transform matrices, and the creation and setup of basic materials. While the 3D assets would be considered very simplistic for a general 3D game, they still take a significant amount of time to set up on a mobile phone. An initial unoptimized version of the game had a setup time of three minutes during which the phone appeared to “hang,” which could be problematic for users.

Even with optimization of the graphics code, the setup time could only be reduced to about 10 seconds on a Nokia 5500, and 15 seconds on a Nokia E61. So the simple game developer’s trick of hiding the time on a thread that runs behind splash screens was implemented. Static graphics were utilized for splash screens, the high score table, and main menu, all provided as resources within the JAR file, which makes the game self-contained for deployment purposes.

Scoring is based on a simple countdown method, and each frame decrements the score by one. Although this means that the game differs from device to device, this was not considered too much of a problem for a number of reasons: the games’ main focus is one device, the Nokia 5500.

High scores are recorded in the phone’s Record Store service, which provides a simple flat database facility for applications. Records can only be stored as a single byte stream, but this was sufficient for our requirements, and the highest score for each level was recorded.

In the original version of the game [Gilbertson et al. 2007], the player moves at a certain minimum speed, except when colliding with an obstacle when the player is stopped. As soon as the player moves laterally to get around the obstacle, minimum forward speed is restored. This provides a clear and intuitive indication of when the player is clear of obstructions. The player may tilt the phone downwards to accelerate. Acceleration is a constant addition to forward speed up to a maximum speed. Tilting the phone backwards decelerates to the minimum speed. Lateral movement is at a single constant speed and is unrelated to forward movement.



Fig. 4. Game with speed bar and options screen.

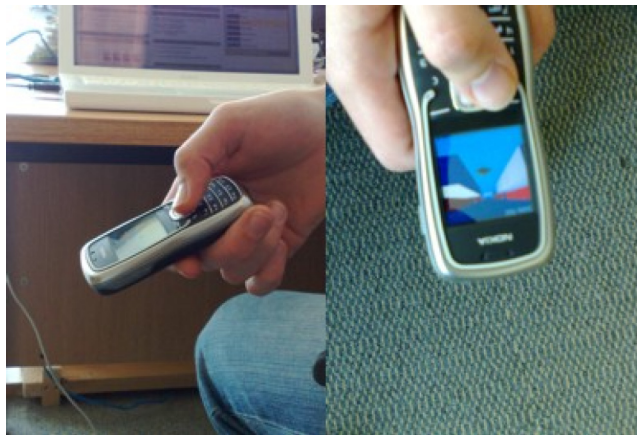


Fig. 5. Original gameplay.

However, there were a number of issues raised by the first user experience tests [Gilbertson et al. 2007], which needed to be reflected in a new version of the game. First, users asked for a visual indication of player speed relative to the maximum. Speed was indicated by adding a bar-style instrument to the upper-left corner of the game screen, as shown in Figure 4.

Second, some users had issues with tilting the phone downwards to accelerate, as they felt it caused reflections on the screen, as shown in Figure 5.

This was addressed in two different ways. An option was added to reverse the vertical tilt input, which now made tilting downwards the brake, while tilting upwards caused the player to accelerate. At the time we made the change, we thought that this interface was likely to be counterintuitive, so we implemented an alternative solution to provide the user with the option of recalibrating the “zero-input zone.” This zone represents the sensor readings when the phone is being held in a position of no tilt. In the original game this was hard-coded to the phone being in a horizontal position, as if placed flat on a table with the screen upwards, so that $1g$ would be sensed on the Z-accelerometer only. By

adding an entry to the new options screen, the user can hold the phone in a comfortable position and by selecting the “calibrate” option set the zero point to the new position. The input is then referenced as changes from the new values on the accelerometers.

We made one final addition: the ability to switch from sensor input to joystick input, as for the previous test we had two separate builds of the same game but the addition of this option allowed us to unify the code base.

4. USER EXPERIENCE

Mobile application development is still in its relative infancy, which is complicated by the fact that advances in hardware and software are accelerating at a rapid pace. However, as with other forms of games, when evaluating mobile game applications the intended user’s experience should be considered paramount [Bucolo et al. 2005]. In this game the main focus is on the experience of the tilt interface in relation to user feedback from the previous study [Gilbertson et al. 2007]. As before, a comparative study was undertaken in order to provide an effective measure whereby the users’ experience could be evaluated by allowing them to play the same game on the same phones using different tilt interface modes than the traditional joypad interface.

For the study we obtained 12 volunteers from around the university, and as with the previous study [Gilbertson et al. 2007], took care to avoid students studying technology, as they would bias the evaluation. Additionally, for this study we avoided using previous participants, as we felt they may have preconceived ideas due to their exposure to the previous version. The group consisted of seven males and five females; three members of the group were left-handed. The majority were in the 18 to 24 age group; all of them owned mobile phones and had previously played a mobile phone game; five of the twelve owned a game console. In terms of preferred game genres, the most popular was sports, which was chosen by five group members; three each chose puzzle and arcade games; and two preferred first-person shooters. While all members of the group played mobile games, only four had downloaded games onto their own phones, and of these only one had actually paid for a game. In terms of phone use, the entire group described themselves as predominantly voice and text users; only two had used data services.

For the study, we asked each member of the group a set of predefined questions related to games and mobiles (described above), before asking them to play with the traditional joypad; play *Tunnel Run* with the original accelerometer controls; and finally to play with reversed action on the vertical. At the end of this session, we also allowed the users to experiment with the calibration to see if they felt it improved the interface. We gave each of the group a basic overview of the game controls, that is, left right and forward back, before asking them to play. During the games we observed their actions and responses and recorded their first and then their best scores for *Tunnel Run*, before asking them to compare their experiences with the three interfaces at the end of the play session. Each user study lasted approximately 10 to 15 minutes; the results are presented in the following sections. Note that the



Fig. 6. Single-handed play.

structure of the analysis follows in part that of Bucolo et al. [2005] and that of our previous study, as there are no other studies of mobile phone games using 3D accelerometers. The work provides an interesting comparison to ours because Bucolo et al. used “tilt” as the primary input, but obtained a secondary effect from the movement of the camera.

4.1 Operating Position

While in our previous study we also considered the operating position, the main findings were related to single- or dual-handed gameplay [Gilbertson et al. 2007], which is seen as an important feature in a mobile game [Nokia 2006]. In this study all our subjects operated the phone single-handed; the most marked difference was in the position they adopted to play the game, as shown in Figure 6. When playing the game with the accelerometers using the original playing mode, players tended to be hunched over the phone as in the left-hand picture of Figure 6, while for the joypad and with the tilt control reversed, players adopted a more upright stance, shown on the right of Figure 6. This is probably why all players who preferred the tilt interface, which was 11 out of our group, also preferred its operation with the controls reversed from original game.

This was unexpected; during the redesign it was considered counter-intuitive, but this opinion may have been biased by the fact that while developing the game the researchers had become overly familiar with the original configuration.

In terms of the calibration mode, the majority of players felt it did not offer significant improvement, and indeed two players managed to calibrate the controls to a point where the game was unplayable. We would remove this feature if the game were to obtain commercial release.

4.2 Game Progression

The graph in Figure 7 plots the difference in scores (rather than the actual scores) between the first play and the best game for each of the three game

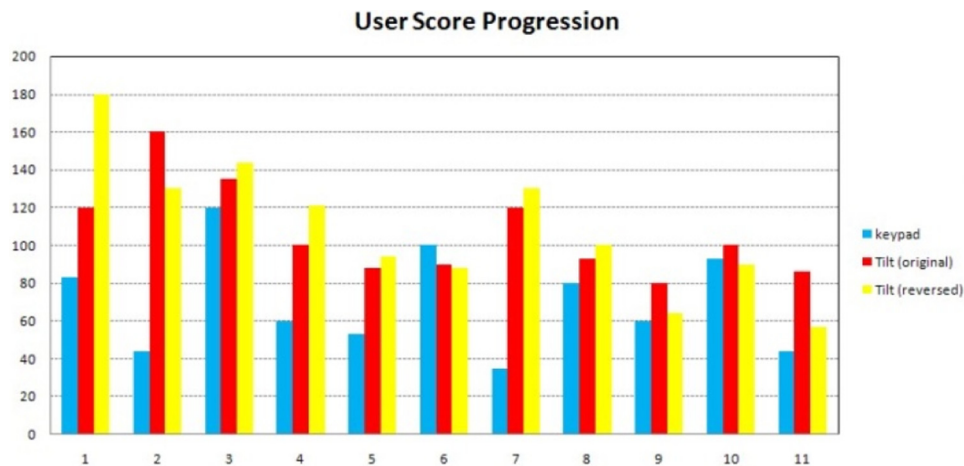


Fig. 7. Player scores.

modes. While in our original study it was inconclusive as to whether the tilt interface took more time to master, the results this time show both tilt interfaces generally resulted in the greatest improvement in scores. However, as previously, it is worth noting that the best score was not necessarily the last, and when questioned at the end of the session most users said they felt their scores had improved mostly because they had become familiar with the maze and not because they had gotten used to the interface [Gilbertson et al. 2007].

4.3 Improvements Suggested by Users

In the previous study the main problems, which we addressed in the redesign, were that users had to tilt the phone away from them to increase speed and the lack of a speed bar. In this trial the only real issues were calibration, discussed previously, and minor suggestions for improving sound and vibration. Hence we feel that the iterative design process has thus far produced very significant improvements overall.

4.4 Comparing Input Methods

During both trials, we recorded the users' comments and then got them to assess the merits of the tilt interface compared to the traditional keypad interface. The most typical comments made by users when first starting to play the tilt game were "that's so cool" and "this is more fun."

The almost unanimous opinion (one player preferred the joystick) was that the tilt interface was the most fun. Interestingly, a number of users asked whether "this was a commercial phone" and "whether they will put the sensors in all new phones," which, along with the comment that "this is great I don't usually play these types of games" should encourage manufacturers to consider incorporating this technology into a wide range of devices.



Fig. 8. Motion control of a 3-D world.

5. FURTHER WORK

Based on the work in this article, we have extended our research to the production of an application programming interface (API), to allow the use of the in-built accelerometers in mobile phones to manoeuvre in 3-D virtual or augmented reality (mixed reality) worlds with full 3-D motion control. The API, which has been termed MIRAGE-X, is currently implemented for the Symbian OS. It provides the user with a first-person view that allows for instinctive navigation through the 3-D world by simply moving the phone in the direction the user wishes to go. According to that movement the relative view direction in the 3-D environments gets updated and displayed on screen, as shown in Figure 8.

To demonstrate the API we have created a game called *Mirage Money*. The game is a flight-simulation with the phone screen being the window to the virtual, or augmented, world the player is flying through. The player's mission is to "fly" around in the 3-D world collecting floating silver and gold coins by colliding with them on the screen.

There are two modes for playing *Mirage Money*:

- (1) *Real Mirage*: the coins are augmented on images captured by the phone camera, as shown in the top image of Figure 9.
- (2) *Virtual Mirage*: the coins float in a virtual, sky-bounded environment, as shown in the lower image of Figure 9.

To gather feedback for future publications, we released the game on a number of free distribution websites; at the time of writing we've had over 10,000 downloads and extremely positive feedback. In the future we intend to create a multiplayer game using Bluetooth, a game where we use the internal GPS



Fig. 9. Mirage Money: the “real” and “virtual” modes.

to create a location-based augmented reality game, which would highlight the enormous potential of this type of interface.

6. CONCLUSIONS

The huge impact that the Wii™ has made on the traditional console market is evidence that many a fun and appealing interface can enable games to engage a much broader game-playing demographic. As mobile phones and mobile games are already impacting new game-playing audiences, it seems natural to explore new interface mechanisms on a device whose traditional keypad input limits the opportunities for gameplay.

In this research project we explored the use of in-built 3-D accelerometers as an interface for a mobile phone game. In particular, we avoided replicating a traditional tilt-based game but explored a 3-D graphics driving game instead.

The overall response was extremely positive, and gives us reason to explore the interface further across a broader range of games, and in particular to address a much broader age demographic, who were not represented in this study. Finally, we would say that placing 3-D accelerometers in phones offers a wealth of new interaction possibilities, and actually surpasses the design goal of single-button games by allowing the development of no-button games.

ACKNOWLEDGMENTS

Our thanks to Nokia for providing the software and hardware to the Mobile Radicals research group at Lancaster University for use in the implementation of this project.

REFERENCES

- BARTLETT, J. 2000. Rock 'n' Scroll is here to stay. *IEEE Computer Graphics and Applications* (May/June), 40–45.
- BUCOLO, S., BILLINGHURST, M., AND SICKINGER, D. 2005. User experiences with mobile phone camera game interfaces. In *Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia* (Christchurch, New Zealand, Dec. 8–10), 87–94.
- CHEHIMI, F., COULTON, P., AND EDWARDS, R. 2006. Evolution of 3D mobile games development. *J. Personal and Ubiquitous Computing* (Nov.), 1–7.
- COULTON, P., EDWARDS, R., BAMFORD, W., CHEHIMI, F., GILBERTSON, P., AND RASHID, M. 2007. Mobile games: Challenges and opportunities. *Advances in Computers* 69, Elsevier Press, New York.
- COULTON, P., RASHID, O., EDWARDS, R., AND THOMPSON, R. 2005. Creating entertainment applications for cellular phones. *ACM Computers in Entertainment* 3, 3 (July).
- FISHKIN, K., GUJAR, A., HARRISON, B., MORAN, T., AND WANT, R. 2000. Embodied user interfaces for really direct manipulation. *Communications of the ACM*, 43, 9 (Sept.), 75–80.
- GILBERTSON, P., COULTON, P., AND VAJK, T. 2007. Using tilt as the input for 3D mobile games. In *Proceedings of the 3rd International Conference on Games Research and Development* (CyberGames '07, Manchester UK, Sept. 10–11).
- NOKIA. 2006. Turn limitation into strength: Design one-button games. Version 1.0, May 15.
- VAJK, T., BAMFORD, W., COULTON, P., AND EDWARDS, R., 2008. Using a mobile phone as a "Wii-like" controller for playing games on a large public display. *Int. J. Computer Games Technology*, article ID 539078.
- WIGDOR, D. AND BALAKRISHNAN, R. 2003. TiltText: Using tilt for text input to mobile phones. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (Vancouver, B.C.), 81–90.

Received September 2007; revised May 2008; accepted June 2008