On Mapping Sensor Inputs to Actions on Computer Applications: the Case of Two Sensor-Driven Games

Seng W. Loke
La Trobe University
Australia

ABSTRACT

We discuss general concepts and principles for mapping of sensor inputs to actions on computer applications, as based namely on experiences with two implemented and trialed sensor-driven games. While our work is based on two games, the ideas can be generalized to computer applications where a fixed set of operations can be identified, and algebraic properties for the operations specifiable.

INTRODUCTION

There has been interest on using sensors to drive computer applications, including games. Recent developments such as GestureTek Mobile[1] allows one to rock, tilt or move the phone to control a mobile game. Sony's Wii aims to… One can control a game using the normal keyboard and mouse, but replacing the keyboard and mouse with sensor inputs (even for exactly the same game) adds a new dimension to the game in a number of ways: e.g., what was easy to do by pressing a key is now more challenging since some gesture is required (to be captured by sensors), what was boring to do with mouse clicks now involves whole body movements, and what involved finger coordination now involves hands and legs coordination (or even coordination of two or more persons). The use of sensors itself provides a new level of engagement, new challenges, and novelty of interaction, adding value to game playing. A mundane game can be made more interesting by changing the mechanism of user interaction – the sensor-based interaction is part of the fun itself. Given a fixed set of sensors, changing the mapping from sensor inputs to game actions produces another variant of the game – many variations of the same game can be produced by varying this mapping from sensor inputs to game actions. Two categories of sensor-driven games can be considered: games playable with ordinary keyboards and mouse but instead use sensor inputs, and games specifically tailored for the sensors to use.

Virtual reality and haptic interfaces can capture human gestures and movements and effectively replicate that in the virtual world, e.g., a grasping action is mirrored as a virtual grasping action, or waving a hand-held sensor corresponds to a racket swing. While virtual mirroring of physical actions, or exploiting *commonsensical coherence* between physical actions and virtual actions, is one way to employ sensor inputs, this need not be the rule – a waving of a hand-held sensor may be mapped to be simply firing a bullet, waving the sensor to the left may not result in a virtual swing to the left but a virtual swing to the right – the confusion adding a challenge feature to game play, or one sends an email by switching on a lamp.

In general, using sensors enables capturing of more of what we can do in the world and provides more degrees of freedom in interaction. It also enables more people to be involved at the same time, even on a single computer – e.g., multiple players controlling a tank.

In the rest of this paper, we describe two games which we have prototyped which relies on sensor inputs. We describe the interaction design of the games with the underlying philosophy, how we realize the two systems, and outline experiences in putting the games out for public trial and demonstration.

TWO SYSTEMS

The sensor-driven games were originally conceived for pedagogical reasons, as a means to demonstrate sensor technology to a non-computing literate audience and to get the audience to engage with the technology (and gain

[1] http://www.gesturetekmobile.com/

an appreciation for the technology itself), rather than games development. It was not adequately interesting if sensor inputs were simply used to drive cartoon figures on the screen, but a game can provide a storyboard or a reason for continuously interacting with the sensors - sensor-based interaction was the end, rather than the means, and the game was the means.

We worked with several Phidgets sensor toolkits,[2] and so, had an array of sensors to work with, including distance sensors, force sensors, sliders, motion sensors, RFID tags, etc, which means a vast set of possible combinations of sensor inputs can be generated. A game typically has a finite set of operations, and a design issue is how and what sensor inputs to map to each operation. A guiding principle is commonsensical coherence as mentioned (e.g., physical left wave corresponds to a virtual left swing) but we also considered *intentional incoherence* in two ways: *antithetical incoherence* (e.g., physical left wave corresponds to a virtual right racket swing), *orthogonal incoherence* (e.g., rubbing the hands to produce heat (as detected by a temperature sensor) corresponds to a virtual right racket swing) which adds a challenge feature to the game. Sometimes the coherence may only be *tangentia*l (e.g., given two distance sensors, simply placing the hand over the right sensor results in a virtual right racket swing). Also, the mapping from sensor inputs to operations can be *persistent* (fixed throughout the game) or *non-persistent* (does change during game play, each mapping triggered by some event in the game). Depending on the game, the set of operations possible might change at different stages of the game. It is also possible for more than one combination of sensor inputs to trigger the same operation or one combination of sensor inputs to trigger different operations at different stages of the game. The difficulty of the game can also be customized according to what combinations of sensor inputs are required to produce an operation, since in the physical world, certain combinations of physical actions are harder to produce than others.

Tank Warrior

Tank warrior is about a team of three people driving a tank through a hostile (in the sense that there are enemies firing at the tank) terrain, to rescue another tank. Figure 1 shows the screen (which we projected on a large wall), and the three stations (for three players) comprising:

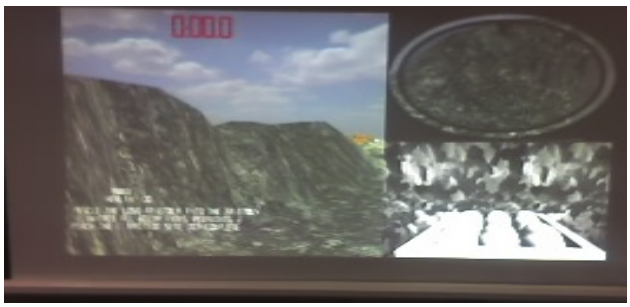
(i). two force sensors for steering the tank (forward, left or right) [left screen],

(ii). two sliders to control the tank's turret (and cannon) (up & down, left & right) and a touch sensor for firing the cannon [what the gun is currently aiming at is in the top right screen]; aiming the cannon is not easy if the tank is moving and so this player needs to coordinate with the tank driver in (i) above so that the tank slows down or moves in a consistent path when the tank is aiming at something, and

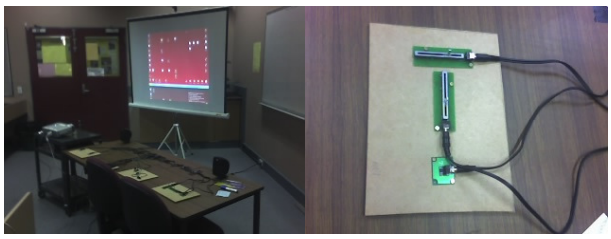
(iii). a distance sensor and a rotation knob to load the gun or cannon, together with a motion sensor to unjam the cannon (which becomes jammed if the cannon is fired unloaded, the 2nd player firing the cannon must coordinate with this player loading the weapon) [bottom right screen]. The player raises his hand over the distance sensor to "pick up" a shell; the rotation knob is used to shift between the shell store and a hole (in which the shell is loaded); the player unjams the weapon by waving his/her hands in front of the motion sensor rigorously enough.

The game is, thus, a team game where physical coordination among team members is manifested in the smooth operation of the tank. Because of the actions required for each station, it is not physically possible for one person to effectively play all three stations.

The game was exhibited and trialed by a group of about a dozen high school students (teenagers) (in turns), with very positive feedback about the way one interacts with the game and its novelty.

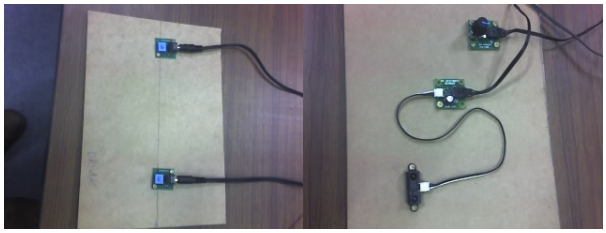


(a) projected screen for the tank controllers



(b) set up of sensors and screen; (c) the sliders and touch sensor

[2] http://www.phidgets.com



(d) the two force sensors; (e) the distance sensor, etc.

Figure 1. Set-up for Tank Warrior

In Tank warrior, generally, we used commonsensical coherence as a guide in mapping sensor inputs to tank controls, e.g., the harder the force sensors are pressed, the faster the tanks move and pressing only one of them will move the tank in one direction and pressing both will move the tank straight. However, to increase challenge, we also experimented with antithetical coherence, we swap the left button with the right button so that pressing left turns the tank right and conversely. The physical form of the sliders immediately suggest limitations on the extent the turret can moved whether vertically or horizontally and also the relative position of the cannon (i.e., how much further it can be moved up/down/left/right relative to its current position). Waving the hands "feverishly" over the motion sensor to unjam it is an example of orthogonal incoherence, but the rigorous action matches the excitement of the game since one cannot fire the weapon without doing this (even in the heat of being shot at by the enemy).

Pokemanz

Figure 2 shows the Pokemanz card game,the screen and the control sensors. A series of seven distance sensors is used, each for selecting one of the seven cards displayed on the screen – passing a hand over the 2nd sensor will cause the 2nd card (from the left) to be selected (this card is raised on the screen); passing a hand again over a selected card will put it down. In the game, each player, at each turn, selects a set of cards to play out of seven possibilities. Each card that is selected represents a weapon used against the other player. The play, is hence, similar to the typical Pokemon trading card game[3] and was, in fact, inspired by it. Each player selects his/her representative team characters via passing a selected set of cards over an RFID reader, each card embedded with an RFID tag, and each character being represented by a card (and its embedded tag with a unique ID). The figure below shows the kids at a Departmental Open Day playing the game after only a few words of instruction – hence, the game interface, though different from anything that the young kids (roughly in range 8 to 12) have ever seen, can be learned in a very short time. The kids then played unsupervised.

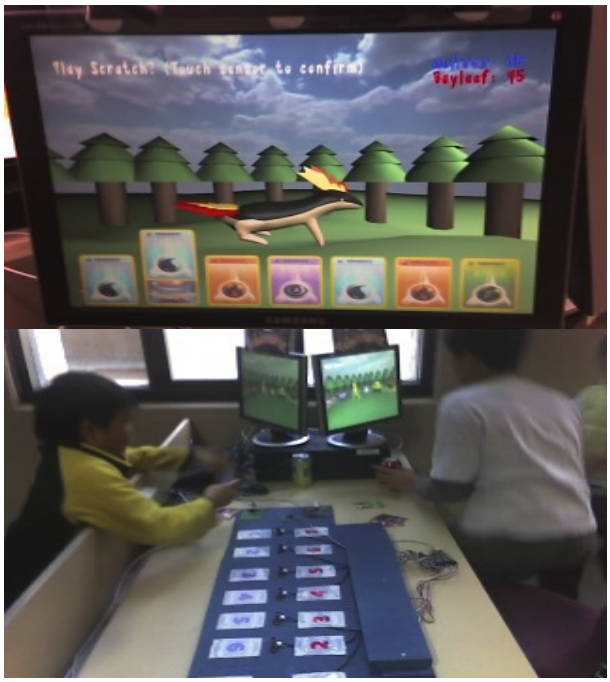


Figure 2. Pokemanz being played by two kids.

Moving the hand over the right distance sensor to pick up a card is an example of tangential coherence (to pick up a card physically, one has to not only move the hand over the card but also then pick it up – here, it is sufficient to simply move the hand over the right distance sensor). Selection of a character is by choosing the right card and passing the RFID embedded card over the reader. While this action has pedagogical value, suggesting to the kids that each card (or tag ID) uniquely identifies a character, the action is more than what one may do in making a selection (compared to simply pressing a key on the keyboard).

ALGEBRAICALLY FORMALIZING THE MAPPING

Inspired by [7], the set of operations in a game (whether controlling a tank or selecting cards) can be formalized into an algebra of operations, where two or more more operations can be composed or where certain operations have inverses (continuously effective actions can be discretized into a series of operations) so that the composition of an operation and its inverse results in an identity action. Some pairs of operations may commute and others may be idempotent.

Similarly, a set of sensor actions can be identified and their algebraic properties formulated – certain actions may cancel each other out (e.g. left and right force sensor actions leading to left and right tank movements),

[3] http://en.wikipedia.org/wiki/Pok%C3%A9mon_Trading_Card_Game

commute (it does not matter what order the actions are carried out) etc. The algebraic structure of the sensor actions (including the algebraic properties of sensor actions) can be induced by the algebraic structure of the game operations (since each sensor action map to a game operation).

The mapping from sensor actions to the game operations should be *exhaustive* (or *surjective*), i.e. there should be some way (some sensor action) to perform every game operation, and without clashes (i.e., a *bijective* mapping). Also, unless greater challenge is required, the mapping should be persistent so that the players are not too confused (we used persistent mapping in both games). Also, the mapping and compositions of should be *physically realizable*, in the sense that, for example, if a sequential (in time) composition of two game operations is required to perform a task in the game, then it should be physically possible within time constraints to perform the sensor action for the first game operation and then the next sensor action for the second game operation (e.g., one action is to pass a hand over the distance sensor and the other is to pass a hand over another distance sensor some distance away from the first, one should be able to do the first action, and then, within the allowed period, the second action; the period must not be too short for otherwise this task cannot be performed, unless another person is involved). Given a set of game operations, all legal compositions should be physically realizable. While there is no tool to automatically check for such properties, the designer would have to consider these issues.

RELATED WORK

Since their inception, there has been much work using Phidgets not only for tangible inputs to computer applications[4] but also for physically tangible outputs [1,2,3,4,5]. The notions of Tangible User Interfaces and Pervasive Gaming rely on sensors for their realization. Games have also been developed in [6,1]. However, we see that the mapping between sensor actions and operations on computer applications have not been comprehensively discussed in the literature as we do here.

CONCLUSION

We have discussed modes of coherency and incoherency in mapping actions in the physical world as picked up by sensors to operations in two game applications. We have also outlined an algebraic perspective in this mapping, characterizing, mathematically, favourable properties of such a mapping. From scratch, the Phidgets toolkit enabled three persons to develop the games (in a part time manner) over a total span of roughly eight weeks.

The space of possible user actions which can be picked up by sensors is remarkable – and limited only by the imagination. For example, one could kick a soccer ball to fire a cannon shot, albeit being purposefully orthogonally incoherent, this could add a different dimension to a game. Or, one could pedal a bicycle to keep the tank's fuel up – complementing the game with a fitness element, and making visits to a particular store (and these visits being permitted to be tracked by a GPS system) can enable ammunition to be added to a tank (adding an incentive to visit a store). Real world movements and interactions can be mapped to appropriate game operations, over long term or short term, but in a way that is commonsensically coherent, purposefully incoherent, and respecting the algebraic structure of the set of game operations.

ACKNOWLEDGMENTS

The author thanks Courtney O'Sullivan, Gordon Pedersen, Alister Smith and Brendan for implementing the games here and contributing ideas.

REFERENCES

1. Jung, B., Schrader, A. and Carlson, D. Tangible Interfaces for Pervasive Gaming. *Proceedings of Digital Games Research Association International Conference (DIGRA),* 2005.
2. Kimura, H., Tokunaga, E., Okuda, Y. and Nakajima, T. CookieFlavors: Easy Building Blocks for Wireless Tangible Input. *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, ACM Press, pp. 965 – 970, 2006.
3. Klemmer, S.R., Li, J., Lin, J., and Landay, J.A. Papier-Mâché: Toolkit Support for Tangible Input. CHI Letters, Human Factors in Computing Systems 6(1), 2004.
4. Koleva, B., Benford, S., Ng, K. and Rodden,T. A Framework for Tangible User Interfaces. Proceedings of the Workshop on Physical Interfaces, at the 5th International Symposium on Human Computer Interaction with Mobile Devices and Services (Mobile HCI, September, 2003, Udine, Italy
5. Mazalek, A. Tangible Toolkits: Integrating Application Development across Diverse Multi-User and Tangible Interaction Platforms. Proceedings of the Let's Get Physical Workshop, at the 2nd International Conference on Design Computing and Cognition, 2006
6. Rogers, Y., and Muller, H.L. A Framework for Designing Sensor-Based Interactions to Promote Exploration and Reflection in Play. International Journal

[4] http://grouplab.cpsc.ucalgary.ca/phidgets/

of Man-Machine Studies 64(1), pp. 1-14, 2006.
7. Thimbleby, H.W. User Interface Design with Matrix Algebra. ACM Transactions on Computer Human Interaction 11(2), pp. 181-236, 2004.