Classifying Input for Active Games

Tadeusz Stach, T.C. Nicholas Graham, Matthew Brehmer, Andreas Hollatz
School of Computing
Queen's University
{tstach, graham, 4mb14, andreas}@cs.queensu.ca

ABSTRACT
Active games are video games that involve physical activity. Interaction in active games is captured via a variety of input devices such as accelerometers, cameras, pressure sensors and exercise equipment. Although active games have become highly popular, the interaction styles they support are poorly understood, and largely driven by the capabilities of individual hardware devices. In order to allow for a standard development approach for active games, a better understanding of the interaction found in such games is required. We have investigated existing commercial and academic games in order to classify input for active games. Our classification abstracts input from hardware, providing a better understanding of the interaction itself. Based on our classification, we propose that active games can be developed independently of underlying input hardware. We illustrate this through our GAIM input framework for active games, and its application to the implementation of a device-independent car racing game.

Categories and Subject Descriptors
H.5.2 [User Interface]: Input devices and strategies, Interaction styles;

General Terms
Active video games, exercise video games, exergaming.

Keywords
Keywords are your own designated keywords.

1. INTRODUCTION
Active games, video games that involve physical activity, have become tremendously popular in recent years. Examples of active games include Wii Tennis, where players swing an accelerometer to control a tennis racquet [17]; Dance Dance Revolution, where players perform dance steps to music [8], and Frozen Treasure Hunter, where players pedal a bicycle while carrying out quests in a virtual world [21]. Nintendo's Wii, a console designed to support active gaming, has sold over 45 million units to-date, strongly illustrating the popularity of such games [16]. Recently announced motion sensing technologies, such as Microsoft's Project Natal and Sony's motion controller, have further increased the interest in active gaming.

Despite their commercial success, understanding of the interaction techniques underlying active games is immature. Most active games are designed for a specific hardware platform: Wii games are based on input from accelerometers and IR tracking; EyeToy games are designed around camera input [10], and PCGamerBike games are tied to pedal and steering input. This is analogous to the early days of graphical user interfaces, where programmers needed to deal directly with mouse input rather than using high level widgets for scrolling, text input and menu handling.

We address this problem by presenting a classification of input techniques in active games. The classification identifies six input styles, abstracting the details of hardware from the interaction itself. To develop the classification, we reviewed 107 active games, drawing from commercial, academic and "fantasy'' game designs, and extracted the common forms of interaction.

Each of the input techniques appears in several surveyed games. While some hardware provides better support for particular kinds of input, in general, the input styles crossed platform boundaries. We found that most games combined multiple forms of input, and that the interaction style of a game can often be captured by listing the forms of interaction it supports.

This high-level classification of input for active games has several benefits. It represents a starting point in standardizing the forms of input that active games provide. We show how this in turn can help in developing a toolkit that supports a wide range of devices, aiding the development of portable active games. Better understanding of active game inputs may help in the standardization of input devices, allowing different technologies (e.g., camera vs accelerometer) to provide the same programming interface.

This paper is organized as follows. We first review existing hardware used in active video games and the forms of input they capture. We then present our classification of input for active games, and provide illustrations of how the classification can be used to describe existing games. We then show how the classification can be used as the basis of a software framework for implementing active games over a variety of hardware devices. Finally, we summarize the implications our classification has for designers.

2. EARLIER INPUT CLASSIFICATIONS
To our knowledge, this paper represents the first attempt to categorize in a hardware-independent way the inputs used in active games.

Several frameworks have been proposed for specifying the capabilities of input devices in general. Notably, Card et al. presented a general model for input devices and showed how it could be used to analyze their effectiveness [3]. The development of direct-manipulation interfaces, has led to various specifications of abstract interactions afforded by a combination of mouse, keyboard and bitmapped display. Duke et al. illustrate the need for a framework to convey interaction in software applications, and propose the *interactors* model [5]. At the toolkit-level, the Garnet system encapsulates interactions into interactor objects [14]. This system allows for the development of interactive desktop application independently of input handling.

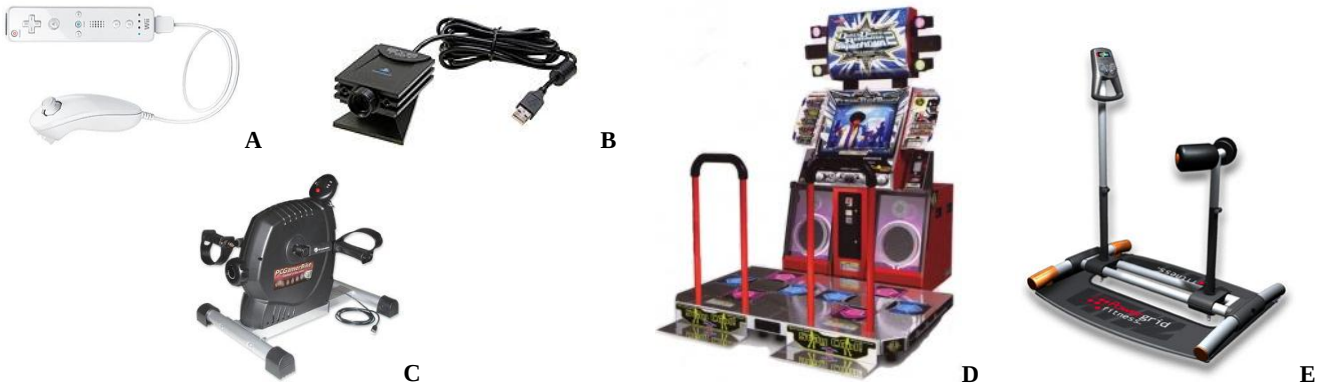

Figure 1: Active gaming hardware examples. (A) Nintendo Wii Remote and Nunchuck, (B) Sony EyeToy, (C) PCGamerBike Mini, (D) Konami Dance Dance Revolution, (E) Powergrid Fitness Kilowatt

Attempts to provide abstract classifications of input techniques have been performed for special domains such as multiuser tabletop surfaces [21], multi-modal interfaces [1], and augmented reality [4].

The concept of embodied interaction explores the intersection between physical and social computing. Embodied interaction has been used to distinguish between the low-level interactions traditionally used in games (such as button presses and joystick movements) versus physical movements more directly associated with real-life activity (such as swinging a tennis racket) [6]. While providing high-level motivation for understanding the differences between these kinds of interactions, this framework does not give a practical classification of the different kinds of input that might occur in a real active game.

3. ACTIVE GAMING HARDWARE

Although a relatively new form of entertainment, active gaming is supported by numerous special-purpose input devices. These range over devices designed to support a specific game (such as the Dance Dance Revolution game pad) through peripherals supporting a range of games (such as the Wii Remote). We now summarize the diverse ways in which physical movement is captured in video games. This summary complements that of Sinclair et al. [18].

3.1 Accelerometers and Gyroscopes

An accelerometer is a device capable of measuring changes in speed. Accelerometers can sense physical motion by measuring acceleration in multiple axes. A typical six degree of freedom accelerometer reports acceleration in the three spatial axes and rotation around those axes. Nintendo's Wii Remote and Nunchuck are popular examples of accelerometer-based gaming peripherals (see figure 1A), and are used in hundreds of games. Accelerometers suffer from drift resulting in cumulative measurement error, and so may be complemented with other technologies to improve their accuracy. The Gametrak Freedom is a hand-held device developed for the Xbox 360 which combines accelerometers with ultrasonic positioning. Similarly, Nintendo's Wii Motion Plus includes gyroscopes which more accurately measure rates of rotation, and when combined with accelerometer measurements provide more accurate motion control.

Accelerometers are used in games to detect and interpret many kinds of motion. In Wii Tennis, players perform backhand and forehand swings while holding the Wii Remote. Similarly, the Wii Motion Plus is used to track a player's golf swing in Tiger Woods PGA Tour 10. In Posemania, players wear several accelerometers that are used to detect when the player has achieved a dance pose [20]. Buttussi et al. use accelerometers to detect knee bends and jumping [2].

3.2 Vision

Computer vision systems typically use a camera to capture movement. Basic vision systems include the Sony EyeToy (see figure 1B) and the PlayStation Eye, which capture human motion by finding differences in consecutively captured frames. For example, in the EyeToy: Groove game, video of the player is overlaid on the game display. The player attempts to use physical punches to hit virtual targets shown around the edge of the screen. When a player moves her arm over a target, a "hit" is registered. Similar techniques are used in Body-Driven games [9], Breakout for Two [13] and Kick Ass Kung-Fu [7]. In all of these games, one or more cameras track player movement with the use of computer vision algorithms.

A related approach is to use a camera to track infra red (IR) tags. For example, NaturalPoint's TrackIR implements head tracking by using a camera to detect the position of head-mounted IR light-emitting diodes. A similar approach is used in the Dodge-It! game to allow players to physically dodge incoming missiles [22].

Vision can be augmented with other technologies to improve its accuracy. Sony's newly announced motion controller captures movement based on visual detection of a hand-held baton, using ultrasound to improve accuracy. Depth cameras can be used to provide 3D images of scenes, allowing tracking of the full body. Microsoft's Project Natal accomplishes this by shining pulses of IR light into the scene, and measuring the reflections of this light.

3.3 Exercise Equipment

Active games (or "exergames" [18]) are frequently designed to promote physical activity. Some exergames are based on sports club equipment, such as stationary bicycles or treadmills. This equipment is designed to provide cardio and muscular exercise, and to measure the work performed. Typically, active games using exercise equipment measure the raw power delivered by the player (e.g., based on the gear and pedaling speed of a bicycle) and translate this information into a game mechanism (e.g., avatar speed). For example, the PCGamerBike (see figure 1C) can be attached to a computer and used as an input device to control the speed and direction of the player's avatar in World of Warcraft. Other commercial systems include the Cateye GameBike, Fisher-Price Smart Cycle, and the Gamercize products. Stationary

bicycles have also been used in academic games such as Frozen Treasure Hunter [23] and Heart Burn [19].

3.4 Pads and Mats

Several very popular games use large pads or mats, placed on the floor or mounted on a wall, to capture user input. These peripherals typically include touch sensors to capture contact (or weight distribution) in a particular region of the device. For example, Dance Dance Revolution uses a floor mat to determine when a player steps on a particular square of the surface (see figure 1D). Other commercial pads and mats include the Nintendo Power Pad, the Wii Balance Board, and the XaviX J-Mat. The Remote Impact game uses a wall mounted pad to capture players' kicks and punches [12].

3.5 Special Purpose

Other active video game hardware has been created for unique game interaction. For example, the Powergrid Kilowatt game controller (see figure 1E) is a resistance training device which acts as an exaggerated joystick and is used in the Push'N'Pull game [13]. In Push'N'Pull, players capture virtual objects by pushing or pulling on the Kilowatt controller; the onscreen objects are easier to catch when more force is applied to the controller. The FlyGuy game [13] requires a specially developed hang-gliding harness to play. Specialized equipment is usually created for a single active game and may not be able to support the diversity of games of a more standard controller (e.g., Wii Remote or EyeToy).

In the following section, we describe our approach to classifying input in active games in order to allow interaction to be understood independently of the hardware used.

4. METHOD

We analyzed the input techniques used by 107 active games, and from these abstracted a set of input styles. We examined a broad spectrum of games drawn from three categories:

- Commercially available active games (including training systems such as Wii Fit and EyeToy: Kinetic).
- Research prototype games reported in the academic literature.
- "Fantasy" game designs elicited from students in a game development lab.

These categories allowed us to consider proven game designs, designs originating from research laboratories, as well as imaginative blue-sky designs.

The commercial games included both popular and lesser known games. Commercial games fell into three sub-categories: Wii, EyeToy, and "other." The Wii games cover a broad set of titles using the Wii Remote and Nunchuck, the Balance Board, and a combination of both. The chosen Wii games were those requiring active input as opposed to just the directional pad and buttons. The EyeToy category contains an exhaustive list of games using the EyeToy camera. The "other" games category is made up of active games tied to specific commercial equipment such as the PCGamerBike and the XaviX J-Mat, as well as demonstration games created for Microsoft's Project Natal.

The academic games we examined cover those active games that we were able to find in the literature. These include mixed-reality games and games designed to promote physical activity. We restricted this category to games that could be played in the living room, and therefore did not include ubiquitous games.

Game Type		Total
Academic		17
Commercial	Wii	35
	EyeToy	15
	Other	17
"Fantasy"		23
	Combined Total	107

Table 1: Total number of game types investigated.

Our set of "fantasy" games was provided by five researchers working in an academic video game lab. The researchers were asked to provide three to five active game concepts with complete descriptions of interaction and gameplay. The participants were not told what the purpose of the game concepts was, and were not instructed to consider specific input devices. Since the researchers did not have to actually implement the games, they were limited only by their imagination. These games were included in order to explore interactions that may not exist in currently available games.

In total we investigated 107 active games: 67 commercial systems, 17 academic and 23 "fantasy" games (see table 1). We described the input for each game in a hardware-independent fashion. From this raw data, we abstracted a set of general input types. Most active games require multiple forms of input and were therefore classified using a combination of several inputs types.

Three researchers evaluated the input of each game in the set of active games. Whenever possible, one or more of the investigators would play a game in order to explore the active input involved. However, it was not feasible to play all of the games in the set due to availability (e.g., access to all of the commercial games, and research prototype games using specialized equipment). In instances where it was not possible to play test a game, the investigators reviewed descriptions of the game and when available, examined videos of gameplay.

Our criteria for identifying input types were:

- The input type must be hardware-independent. We define this by requiring that it must be possible to capture the input with at least two existing input technologies.
- The input type must be seen in at least three games.
- The set of input types must be orthogonal, and must cover most if not all inputs seen in the examined games.

Our final input classification is presented and discussed in the next section.

5. INPUT CLASSIFICATION

After reviewing the input techniques found in the set of 107 games, six common forms of input emerged: gesture, stance, point, power, continuous control, and tap. Here we define each of the input types and describe their common usage.

5.1 Gesture

A *gesture* is a movement of the limbs, head, or body within a defined pattern. The location and orientation of the body is normally irrelevant to a gesture, but timing is important.

Game Type	Input Classification					
	Gesture	Stance	Point	Power	Continuous Control	Tap
Academic	6	1	0	2	8	1
Commercial	42	19	5	8	8	12
"Fantasy"	21	15	4	1	0	0
Total	69	35	9	11	16	13

Table 2: Input classifications found for each game type examined.

Gesture input is used in Wii Tennis. Players must hit a tennis ball using a forehand or backhand swing. A forehand gesture is performed by swinging a Wii Remote forwards and to the left; a backhand gesture involves swinging forward and to the right. The force of the swing determines the speed of the returned ball.

Gestures specify commands, not real-time control. When a gesture is complete (e.g., forehand/backhand swing), it is communicated to the application, which executes an associated command (e.g., avatar performs forehand/backhand swing). The position and orientation of the player have no effect on recognition of the gesture. Gestures may be captured by technologies as diverse as accelerometers and camera-based motion-capture devices.

5.2 Stance

Stance captures a player's physical position at an instant of time. A stance is not an action, but describes the placement of the player's feet, hands, and body.

Stance input is used in Wii Fit's Yoga game. Players use a Balance Board as an input device. Four pressure sensors monitor the player's center of mass, providing a coarse measure of the player's stance. In Wii Fit Yoga, players must complete a series of poses. Players stand on the Balance Board and hold their bodies in a required position, which is captured in approximate form based on their distribution of mass. Similarly, in Posemania, players are required to take on dance positions in time to music [20]. The player's position is determined from a set of accelerometers attached to her wrists, elbows, knees and ankles.

5.3 Point

Pointing requires players to direct attention to an on-screen entity. Players point by aiming a finger, hand, or hand-held device at a region of interest.

Pointing is used as input for the Secret Agent game found in EyeToy: Play 2. Players must point with their finger at a series of on-screen icons in order to collect them. In Call of Duty: World at War for the Wii, players aim their weapons by pointing the Wii Remote at the screen.

5.4 Power

Power represents the raw physical energy exerted by the player. Power is often tied to movement of the player's in-game avatar; for example, in Heart Burn, the more power that the player provides, the faster her car moves around a track [19]. Power input is typically captured continuously over a period of time.

A wide variety of input technologies can be used to provide power. The PCGamerBike is a compact exercise device equipped with a pair of foot pedals (see figure 1C). When a person is pedaling, the device is able to capture both intensity and direction. For example, the PCGamerBike can be mapped to a set of input controls for the World of Warcraft game. Intensity (measured as pedal speed in RPMs) translates into three possible speeds for a player's avatar: stationary, walking, or running. Conversely, in Heart Burn, power is measured using a heart rate monitor, where current heart rate (indicating how energetically the player is exercising) regulates in-game speed.

5.5 Continuous Control

Continuous control input slaves body movement to an on-screen entity. Typically, the whole or part of the body is used to guide an in-game object. With continuous control, it is possible to capture movement in two or three dimensions.

An example of continuous control is found in the Body-Driven Bomberman game [9]. In Bomberman, a player's character moves around a two dimensional maze while attempting to bomb other characters. A top-mounted camera monitor's players' positions as they move around in a physical space, and maps them to a virtual position in the maze. Thus, a player continuously guides her avatar as she walks/runs in the physical world. Similarly, in Microsoft's Burnout Natal, players steer a car by turning an invisible steering wheel with their hands.

5.6 Tap

A *tap* input requires a player to make contact with a particular object or location in the physical world, and is captured at the moment of contact.

For example, the Remote Impact game [12] uses a wall mounted pad to capture players' punches and kicks (i.e., taps). Two distributed players face their own individual pad reflecting a projection of their opponent. Each player uses her hands or feet to strike the projected image of her opponent. The location and intensity of each tap are captured. Location is used to determine if a strike is a hit or miss, while intensity determines how many points the player is awarded for a hit. Similarly, in Dance Dance Revolution, players use their feet (and hands!) to tap locations on the floor in time to music.

5.7 Summary

The six input types presented above describe all of the active inputs found in the 107 active games we investigated. Many of these games also use traditional (inactive) inputs, such as button presses; these were not considered in our study.

Table 2 summarizes the distribution of inputs over our three game categories. We see that in this particular set of games, gesture and stance inputs are particularly prevalent. This is because of the popularity of the Wii platform, whose hardware is particularly adept at capturing gesture and stance. Nevertheless, with a few exceptions, all identified input types occur in all three game categories, and are represented numerous times over all. We are confident that the classification will describe new games as they are developed, due to the wide range of games and game types that were consulted. For example, newly announced systems –

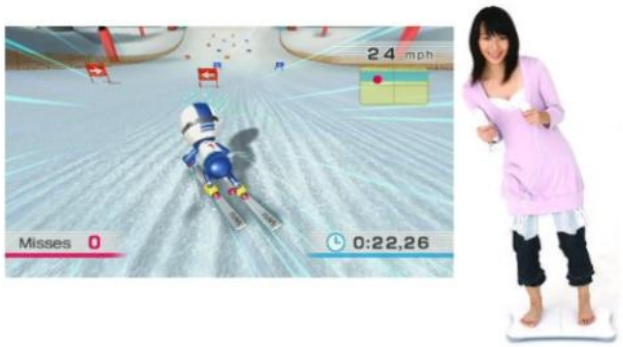

Figure 2: We Ski – player controls using Wii Remote, Nunchuck, and Balance Board.

Figure 3: Frozen Treasure Hunter – players control using recumbent bicycle, Wii Remote and Nunchuck.

such as the Wii Motion Plus, Microsoft's Project Natal, and Sony's PlayStation motion controller – offer novel active input controls. These new devices promise more accurate active control and one-to-one mappings of motion. However, games developed for these peripherals will simply allow for better continuous control, point, and stance inputs – all of which are captured in our proposed input classification.

In the following section, we use this input classification to demonstrate how active games can be designed independently of the peripherals used.

6. ILLUSTRATIONS

To illustrate its effectiveness, we use our input classification to analyze two existing active games. This analysis shows that typical games combine multiple input types, and that these types can be considered independently of the hardware used to implement them.

6.1 We Ski

We Ski uses the Wii Remote, Nunchuck and Balance Board for input (see figure 2). In the game, players guide their skiing avatar down a virtual slope. The player holds the Remote and Nunchuck as if they are the handles of a set of ski poles, moving them up and down to push the avatar forward, and rotating them to make the avatar tuck into a crouching position. When standing on the Balance Board a player is able to control the direction of her avatar by leaning left or right.

These interactions fall into the gesture and stance input types. The pushing motions and the wrist rotations performed with the Remote and Nunchuck are gestures, while a player leaning side-to-side on the Balance Board takes on a series of bodily poses which translate into a set of stances.

Although the gestures in We Ski are captured via accelerometers (i.e., Wii Remote and Nunchuck), and stances are delivered by pressure sensors (i.e., Balance Board), the game could be implemented using different hardware. For example, an EyeToy camera could be used to capture a player's stance based on the position of her head and body. Similarly, arm gestures performed by a player could also be interpreted using vision techniques such as provided by the Sony motion controller.

6.2 Frozen Treasure Hunter

Yim and Graham created the Frozen Treasure Hunter game in order to promote physical activity [21]. Two players share the control of an avatar as they collect virtual items. One player controls the forward momentum of the avatar by pedaling on a recumbent bicycle, and steers using a gamepad. The other player uses a Wii Remote and Nunchuck to swat away virtual projectiles thrown at the avatar (see figure 3). The pedaling of the player on the bike translates into power input, while the swatting motions performed by the other player are classified as gesture inputs.

In the current version of the game power is delivered using exercise equipment, and gestures are captured using Wii peripherals. However, these inputs could be delivered using a variety of other hardware devices. For example, heart rate monitors have been proposed as effective devices for measuring a person's physical effort in active games (e.g., [15], [19]) and therefore can deliver power input. The gestures used in Frozen Treasure Hunter could alternatively be captured using a vision based tracking system.

7. APPLICATION: THE GAIM FRAMEWORK

The core contribution of our input classification is that it helps identify the types of inputs that may be delivered to active games, independently of the underlying hardware that may be used to control the game. Beyond its contribution to the understanding of active input, the classification can be beneficial in the implementation of active games. We have illustrated this through the ongoing development of the *General Active Input Model* (GAIM) framework for handling input in active games. GAIM allows developers to program active games based on the six input types described in this paper. The framework then provides a variety of implementations for each input type, allowing transparent plug-replacement of input devices without requiring modification to the program code.

The framework is divided into three layers. The *input layer* is intended for use by application programmers, and provides access to the six input types identified in our classification. The *abstract input layer* provides interfaces to broad classes of devices (e.g., bicycles, heart rate monitors, accelerometers), while abstracting their differences. The *device layer* provides access to concrete devices. Classes at this layer interact with application programmer interfaces provided by the device's manufacturer or with independently developed interfaces.

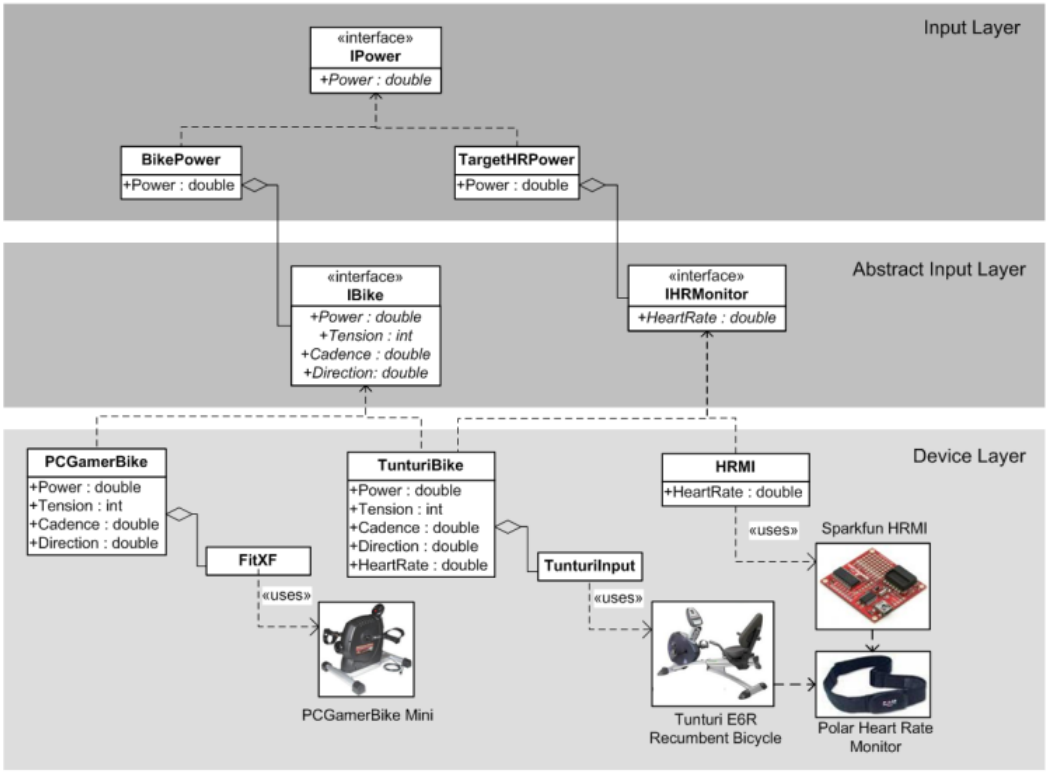

Figure 4: Class diagram of the *IPower* interface from the GAIM framework.

For example, figure 4 shows the classes making up GAIM's *IPower* interface. The interface provides a single property, *Power*, that reports the game player's current power output. The framework provides two implementations of power – one based on stationary bicycles (*BikePower*), and the other based on heart rate (*TargetHRPower*). As described in section 5.4, heart rate input bases the player's power on how close she is to her target heart rate [19]. These classes rely on interfaces provided by the abstract input layer. The *IBike* interface provides attributes capturing the current power, tension, cadence and direction of the bicycle device. The *IHRMonitor* interface reports the player's current heart rate.

The device layer provides access to the equipment itself. The *PCGamerBike* class implements the *IBike* interface, while the *HRMI* class implements the *IHRMonitor* interface. The Tunturi E6R is a recumbent stationary bicycle supporting both cycling and heart rate monitoring, and therefore the *TunturiBike* class implements both interfaces.

The challenge in designing these interfaces is that not all devices provide the same functionality. For example, as a full-featured exercise bicycle, the Tunturi E6R provides full control over tension and cadence, and reports power generated in Watts. As a less expensive gaming peripheral, the PCGamerBike Mini provides only cadence information. (Tension can be set manually, but cannot be read programmatically.) The PCGamerBike mini, therefore, cannot report true power values, since the tension value is required to compute it. The *PCGamerBike* class therefore estimates power from the current cadence and an average tension value. Additionally, tension can be set manually by an application programmer should it have better knowledge of the tension (e.g., via user input.)

7.1 GAIM Racing Game

To illustrate the effectiveness of the GAIM framework, we modified Microsoft's XNA Racing Game (available at www.xnaracinggame.com) to become an active game. The 3D racing game allows a player to race a selected car around several different tracks. In the original game, the car's speed is controlled by the keyboard's arrow keys or by the right trigger on an Xbox 360 gamepad controller; direction is controlled with the left analog stick. In our modified game, speed is controlled by the player's physical activity, as measured by the speed at which she is pedaling a bicycle, or how closely she is matching her target exercise heart rate. Figure 5 shows the active racing game controlled by three different input techniques: two types of bicycle, and via heart rate (elevated by jogging on the spot.) To steer the car, we continue to use the Xbox 360 controller's analog stick.

To modify the game, we removed 35 lines of code taking input from the mouse/keyboard or game controller, and inserted 11 lines of code to process power input. Since the game uses the *IPower* interface, no changes in code are required to change from one device to another. A simple text file is used to specify which devices are available to the application, allowing the framework to determine which class to use to implement *IPower*.

This example illustrates the practicality of basing input on high-level input types such as those described in this paper. Not only does the approach provide device independence, allowing radically different input devices to control the same game, but it

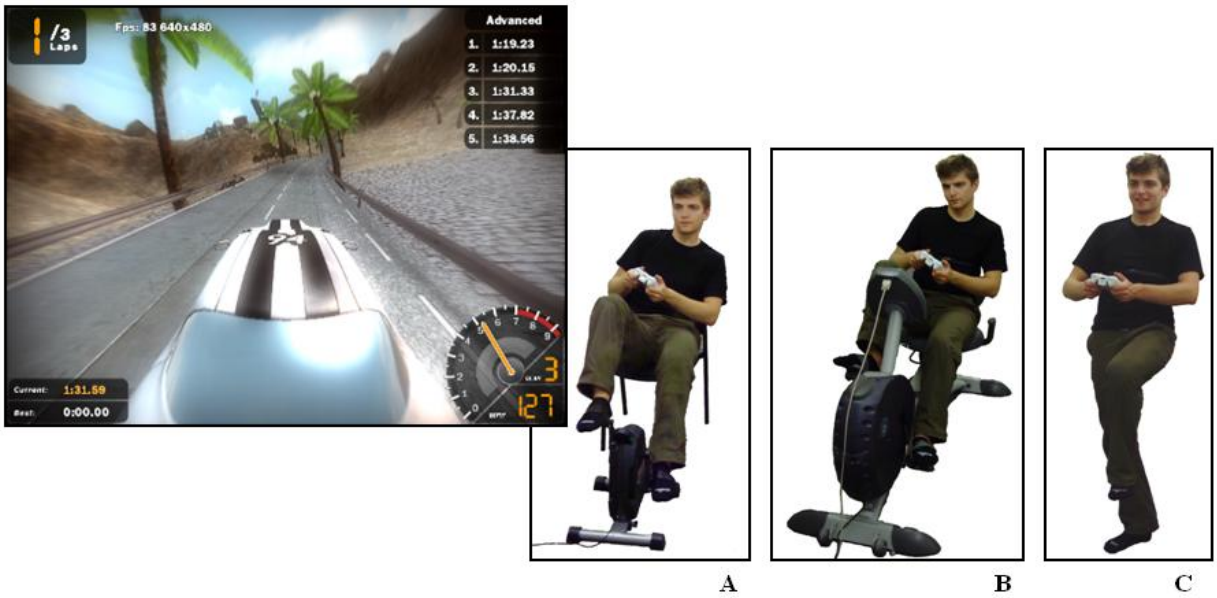

Figure 5: GAIM Racing Game (top-left). (A) PCGamerBike Mini input, (B) Tunturi E6R input, (C) Polar heart rate monitor input

(at least in this case) requires less code to process active input than was required to use traditional input devices.

8. DISCUSSION

Our analysis of existing active games reveals six unique input types: gesture, stance, point, power, continuous control, and tap. We found this input classification sufficient to describe the active inputs of the 107 games that we studied. Our survey is based on existing games (and "fantasy" designs), as well as technical demonstrations of the recently announced next generation of motion capture devices (i.e., Wii Motion Plus, Project Natal, and the Sony motion controller). Although these new peripherals promise a revolution in the design of active games, the input they provide is described by our classification. For example, demonstrations of Microsoft's Project Natal show players controlling a car with an invisible wheel and pedals or using their full body to deflect virtual balls, while early prototypes of the PlayStation motion controller allow players to directly control hand-held weapons (e.g., swinging a mace). These actions map to direct control, gesture, and stance inputs. Therefore, the next generation of active input controllers does not change the types of active input that are possible, but rather improves the accuracy of their detection. For example, Project Natal's recognition of the position of a player's body in space provides more accurate stance input than is possible with the Wii Balance Board's four pressure sensors. The upcoming generation of devices does appear likely to change the input styles that are most widely used. Our analysis of existing commercial games showed a high occurrence of gesture and stance inputs (due to the current capabilities of the Wii and EyeToy); however, we expect future games to include more direct control inputs as a result of improved motion capture technology.

Although we have illustrated the importance of abstracting input for active gaming, playing with different input devices can provide different gameplay experience. For example, the PCGamerBike Mini and Tunturi E6R bicycle used in the GAIM Racing Game provide similar styles of input, but subtly different performance. The Tunturi is more comfortable to sit on and has higher quality pedals. On the other hand, the PCGamerBike Mini is highly responsive to changes in pedal cadence, whereas the Tunturi bike takes upwards of a second to report changes in speed. A toolkit can allow development of games for a diverse set of devices, but ultimately cannot abstract all differences between those devices. This is analogous to the performance difference seen when playing a traditional computer game using a standard mouse versus a track pad.

When compiling our input classification, we focused on active forms of input and omitted traditional forms of input (e.g., analog sticks and buttons). Many active games use both active and standard input. The fusion of both types of input may involve surprising subtleties. For example, in the GAIM Racing Game, an analog stick is used to turn the car. However, the PCGamerBike Mini also allows players to specify direction by pedaling forwards or backwards. Therefore, when holding the stick to the right, pedaling forwards should move the car forwards and to the right, whereas pedaling backwards should move the car backwards and to the left. To solve this problem in the GAIM framework, we introduced a new *IDirection* interface that provides an abstract treatment of direction information, allowing input to be fused from different sources.

In determining the input classification, we explicitly excluded pervasive games [11], in favour of games that could be played in the living room. Given the advent of fast networks and portable devices with global positioning systems and accelerometers (such as the iPhone), it would be an interesting extension to the classification to include this style of game. We speculate that the main additional input would be locomotion, movement that takes a player from one physical location to another.

As we have shown, our active game input classification allows the development of portable games that are independent of specific hardware configurations. Games built around our abstract inputs can allow people with different hardware to play together. For

example, in a two player version of our GAIM Racing Game, one person could use a PCGamerBike while the other player uses a heart rate monitor as an input device. This raises the possibility that some input devices may confer an advantage in competitive games, analogous to the advantages of using a keyboard and mouse versus a game controller when playing a first-person shooter. An interesting avenue for further research will be to determine ways of detecting and compensating for such advantages.

Additionally, our input classification presented in this paper opens the possibility of developing active games for differently abled users. Input mechanisms providing the six input types could be custom-built for players with specific physical limitations.

In future work, we hope to extend the GAIM framework, by supporting additional input and peripheral devices, to allow programmers to express desired interactions independently of the underlying hardware. This will enable developers to more quickly create portable active games independent of particular hardware.

Over all, classifying input in active games opens the possibility of developing games independently of the underlying hardware. Designers should be able to create active games without having to consider the manner in which input is captured, allowing them to focus more on game content and story.

9. CONCLUSION

Currently, when designing an active game, developers must first consider what input hardware the game will utilize. Therefore, active game designers are limited to the capabilities of a specific hardware device. This situation limits creativity and the portability of active games. In order to address this problem we developed a classification of active gaming input.

In our development of the active game input classification, we examined interaction techniques in 107 active games. We were able to extract six major inputs types used in active gaming: gesture, stance, point, power, continuous control and tap. We believe that our input classification is the first approach at abstracting interaction in active games. With continued development of our GAIM framework, we hope to make it easier for developers to create active games without the need to implement low-level input handling for game peripherals.

10. REFERENCES

[1] Bastide, R., Navarre, D., Palanque, P., Schyn, A., and Dragicevic, P. A model-based approach for real-time embedded multimodal systems in military aircrafts. In *Proc. ICMI* 2004, 243-250.

[2] Buttussi, F., Chittaro, L., Ranon, R., and Verona, A. Adaption of graphics and gameplay in fitness games by exploiting motion and physiological sensors. In *Proc. Smart Graphics* 2007, 85-96.

[3] Card, S. K., Mackinlay, J. D., and Robertson, G. G. The design space of input devices. In *Proc. CHI* 1990, 117-124.

[4] Dubois, E. and Nigay, L. Augmented reality: which augmentation for which reality? In *Proc. DARE* 2000, 165-166.

[5] Duke, D., Faconti, G., Harrison, M., and Paternó, F. Unifying views of interactors. In *Proc. AVI* 1994, 143-152.

[6] Gregersen, A., and Grodal, T. Embodiment and Interface. *The Video Game Theory Reader 2*, ed. Perron, B., Wolf, M., 2008, 65-83.

[7] Hämäläinen, P., Ilmonen, T., Höysniemi, J., Lindholm, M., and Nykänen, A. Martial arts in artificial reality. In *Proc. CHI* 2005, 781-790.

[8] Hoysniemi, J. International survey on the Dance Dance Revolution game. *Comput. Entertain*. 4(2), 2006, 8.

[9] Laakso, S., and Laakso, M. Design of a body-driven multiplayer game system. In *Comput. Entertain.*, 4(4), 2006, 7.

[10] Larssen, A., Loke, L., Robertson, T., and Edwards, J. Understanding movement as input for interaction—a study of two eyetoy™ games. In *Proc. OzCHI* 2004.

[11] Magerkurth, C., Cheok, A. D., Mandryk, R. L., and Nilsen, T. Pervasive games: bringing computer entertainment back to the real world. *Comput. Entertain*. 3(3), 2005, 4.

[12] Mueller, F.F., Agamanolis, S., Vetere, F., Gibbs, M.R. Remote impact: shadowboxing over a distance. In *Proc. CHI* 2008, 2291-2296.

[13] Mueller, F.F., Stevens, G., Thorogood, A., O'Brien, S., and Wulf, V. Sports over a distance. In *Personal and Ubiquitous Computing*, 11 (8), 2007, 633-645.

[14] Myers, B. A. A new model for handling input. *ACM Trans. Inf. Syst.* 8(3), 1990, 289-320.

[15] Nenonen, V., Lindblad, A., Häkkinen, V., Laitinen, T., Jouhtio, M., and Hämäläinen, P. Using heart rate to control an interactive game. In *Proc. CHI* 2007, 853-856.

[16] Nintendo. *Consolidated Financial Highlights* (PDF).

[17] Parker, J. R. Games for physical activity: A preliminary examination of the Nintendo Wii. In *Proc. 6th International Symposium on Computer Science in Sport*, 2007.

[18] Sinclair, J., Hingston, P., Masek, M. Considerations for the design of exergames. In *Proc. GRAPHITE* 2007, 289-296.

[19] Stach, T., Graham, T.C.N., Yim, J., and Rhodes, R. Heart rate control of exercise video games. To appear in *Graphics Interface* 2009.

[20] Whitehead, V, Johnston, H., Crampton, N., and Fox, K. Sensor networks as video game input devices. In *Proc. of Future Play* 2007, 38-45.

[21] Wu, M., and Balakrishnan, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proc. UIST* 2003, 193-202.

[22] Yim, J., Qiu, E., and Graham, T.C.N. Experience in the design and development of a game based on head-tracking input. In *Proc. Future Play 2008*, 236-239.

[23] Yim, J., and Graham, T.C.N. Using games to increase exercise motivation. In *Proc. Future Play* 2007, 166-173.