A guide to input methods when considering portability to mobile devices.

COMP160 - Software Engineering Essay

Student number: 1607539

March 28, 2017

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 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 [1].

Introduction

The market for mobile games is both lucrative and in growth. Sensible console and PC game developers will consider the portability of their software to phones and tablets. This will quantitatively include; input methods, output capability, software compatibility, hardware limitations, and qualitatively; whether the games style is suitable for a mobile

device. This essay will examine the input methods and limitations of mobile devices, examine three games that have been ported from a console or PC to a mobile device, and propose a methodology to govern effective input design.

Proper Portability Practices

Given the dominance of the x86 architecture, common application programming interfaces (APIs) through widely used software like DirectX, and industry standard hardware: high definition monitors, qwerty keyboards, and twin stick console controllers, porting software between like machines can be cheap. However, with the market for mobile games worth nearly a quarter (£995.1 million) of the entire UK games industry in 2016 [2], console and PC game developers must consider allowing and easing portability to a mobile system. Bioshock, Grand Theft Auto (GTA), and Final Fantasy (FF) series can be examined as highly successful games that have been ported to smartphones and tablets.

As with many other proper programming practices, allowing flexibility with the hardware input of a game must be considered from the start. Most game building engines, including Unity and Epic, offer input mappings so that custom events can be created and then later mapped to controls. This is essential to the process of porting a game to another system whose control scheme is different, the software will obviously not automatically translate a keyboard's space bar press into a console controller's jump input, but were a jump event defined in the upper levels of the program, a developer can assign that event to any number of different inputs. This also encourages a level of abstraction when developing a controller program which separates the events from the processes. It is important to note that if no such input event customisation options are available, in PyGame for example, then one should be created, through a dictionary structure or similar.

Touch Input

While console conventions have led to hands and fingers assimilated to 16-button, dual analog console gamepads [3], smartphones and tablets can only be relied to have a touchscreen in common. Any buttons are proprietary and are used to change the volume or lock the device. A state machine that interprets touch on a touchscreen can have five different states; started, ended, moved, stationary, or cancelled (for instance, if a call were received on a smartphone mid game) and the developer must recreate complex and intuitive controls using some amount of simultaneous touch states. Accuracy is debatable as a single finger's touch can cover hundreds of pixels [4]. All three of the studied games relied upon a virtual joystick for movement mapped to the touch input on the lower left of the screen. It is an obvious solution given the standard controller convention, and that input mappings exist for joystick axis and amounts. Game developers must consider whether their control scheme can be replicated on a touch device, if it must be rewritten, or how the game can be balanced if either is inadequate. While this may work fine for the slow paced FF franchise; Bioshock and GTA rely on precision and reflexes. GTA has made some attempt to patch this inaccuracy with an enemy lock on and location specific pop up interfaces. FF makes little effort to adapt a control interface, merely replicating a console controller as an overlay. Although this may be to promote nostalgia.

Final Fantasy VII on IOS, on screen controls mimicking a console's controls.

When touch screen controls are being used, then nothing critical to the player experience should be placed on screen in that space. GTA mimics police car chases by showing the chasing car appear into the lower left or right portion of the screen. On a smartphone, this is where the player's hands must be to move the virtual joystick so the experience suffers.

Grand Theft Auto III showing how controls can hide potentially important areas of the screen.

Bioshock, like all first person shooters (FPS), concentrates the action around the crosshair in the middle of the screen, so controls around the edge are more acceptable.

Bioshock on IOS, plasmids.

What are the alternatives?

Different touchstate types [5] are being continually patented which look to add further options to developers. A good gaming experience requires a lot from the user interface (UI). It should be convenient, reliable, and usable so that the player can concentrate on playing the game instead of struggling with the UI [6]. Hardware abstractions facilitate the ability to play a game on different systems [7] but when the control hardware interferes with the UI, alternatives should be considered. Tilt [1], cameras, microphones, gestures [8], magnetic tags, accelerometers [9], and scroll [10] all serve to remove a player's hands from the play area as much as possible. An add on controller can be most effective although this incurs extra cost and size to the hardware, although front isometric joysticks are still preferred over a backmounted joystick [11] on a modified device to save size.

As a last resort, the game can be made easier to balance the frustration of missed information, but this is a workaround, not a meaningful solution.

Conclusion

The workflow should begin by examining where the important in game information is being shown, then to design any on screen controls around this. The controls should be intuitive, usually but not always optimally achieved by mirroring the original controller conventions. Players cannot be expected to always see information under the control interface, and should not be punished for not doing so. If there is no area of the screen that is not being used then the developer must create some by shifting the player's perspective, being mindful of the smaller screen, or explore non touch alternatives such as tilt controls.

References

[1] P. Gilbertson, P. Coulton, F. Chehimi, and T. Vajk, "Using tilt as an interface to control no-button 3-d mobile games," *Computers in Entertainment (CIE)*, vol. 6, no. 3, p. 38, 2008.

[2] L. Hebblethwaite, "Uk games market worth a record 4.33bn in 2016," 2017. [Online]. Available: http://ukie.org.uk/node/30445

[3] D. Parisi, "A counterrevolution in the hands: the console controller as an ergonomic branding mechanism," *Journal of Games Criticism*, vol. 2, no. 1, pp. 1–23, 2015.

[4] D. Schmidt, F. Chehimi, E. Rukzio, and H. Gellersen, "Phonetouch: a technique for direct phone interaction on surfaces," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 2010, pp. 13–16.

[5] S. M. Herz, R. G. Yepez, and W. C. Westerman, "Multi-event input system," Aug. 17 2010, uS Patent 7,777,732.

[6] H. Korhonen and E. M. Koivisto, "Playability heuristics for mobile games," in *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. ACM, 2006, pp. 9–16.

[7] A. BinSubaih, S. Maddock, and D. Romano, "A survey of gameportability," *University of Sheffield, Tech. Rep. CS-07-05*, 2007.

[8] G. Niezen and G. P. Hancke, "Gesture recognition as ubiquitous input for mobile phones," in *International Workshop on Devices that Alter Perception (DAP 2008), in conjunction with Ubicomp*. Citeseer, 2008, pp. 17–21.

[9] C. Harrison and S. E. Hudson, "Abracadabra: wireless, high-precision, and unpowered finger input for very small mobile devices," in *Proceedings of the 22nd*

annual ACM symposium on User interface software and technology. ACM, 2009, pp. 121–124.

[10] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan, "The smart phone: a ubiquitous input device," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 70–77, 2006.

[11] J. O. Wobbrock, D. H. Chau, and B. A. Myers, "An alternative to push, press, and tap-tap-tap: gesturing on an isometric joystick for mobile phone text entry," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 2007, pp. 667–676.