Creating and Designing customized and dynamic game interfaces using smartphones and touchscreen

Mateus Pelegrino Leonardo Torok Esteban Clua Daniela Trevisan
Universidade Federal Fluminense - UFF - IC - Medialab
Email: {mateuspelegrino, leotorok, danielatrevisan}@gmail.com; esteban@ic.uff.br

Abstract

When playing a game, the user expects an easy and intuitive interaction. Current controllers are physical hardware components with a default number, size and position of buttons. While in some cases a game uses a few controllers buttons, in others it requires all buttons and even a combination of them. Besides that, the use of few buttons in a physical controller (i.e. controllers that can't change buttons position) may not lead to a great usability. This work extends a previous research, where we developed an adaptive control using a touchscreen device with machine learning techniques to gather input, rearrange the size and position from buttons. In this work we propose the concept of context sensitive adaptability, which can add, remove, change position or size of a button, depending of a game context. This is a new paradigm to game developers, so they can create a controller that fits to his game and adapts to pre-defined situations, e.g. an enemy approaching might be a reason to increase one buttons size, or a mini-game inside the game might change all the buttons for a moment. To use this concept, we developed a game to deal with the controller, adapting to game context.

Keywords: Adaptive interfaces; adaptive game control; context sensitive adaptability; game context; game input; mobile; touchscreen.

1. Introduction

Gameplay experience is one of the most important aspects when playing a video game. This process includes several characteristics that will determine the final impression and engagement of the user. Among many aspects, the challenges must be correctly balanced or the user can get frustrated by excessive roughness or become bored because the game isn't challenging or compelling enough. The gameplay experience is implemented by the game rules, which is executed through the game interface. In this sense, this interface must be engaged with the user experience, stimulating the user to play and commands his in-game avatar. While several games became infamous because of unresponsive controls or clunky and unintuitive control schemes, others are remembered for the opposite: fast, responsive, intuitive, efficient and innovative control schemes, creating an enjoyable experience that made those games memorable. But the evolution of video game control schemes was not just motivated by the desire to improve the reliability of game input. With the evolution in computer graphics, games became more complex. While in the beginning of electronic games simply being able to control a large pixelated character on screen to accomplish a simple objective was the top-notch experience, the gaming experience evolved. Games started to narrate complex stories and present variated gameplay style and 3D environments to be explored. A more complex world needed a more complex control scheme to allow the player to perform a large range of actions. Unfortunately, an unwanted side effect became clear: the control schemes became less intuitive. An Atari 2600 game controller in the 1970s had a stick and a single action button. Nowadays, a modern game controller has more than 15 different buttons and it is not unusual to find games that use all of them, each one with a different control scheme. Even seasoned gamers would not be able to learn these new complex control schemes by themselves, a fact reflected on the long tutorial levels that most games present to the user: a first level, usually pretty long, that is entirely dedicated to introduce the user to the game controls, showing how to perform actions (that can depend on the current context) and navigate themselves in the game environment. As a comparison, that kind of level was extremely rare in older games, that simply relied on the user learning the controls simply by exploring the controller. Nowadays, trusting on the user learning by himself the controls would simply lead to frustration.

The lost of intuitiveness was perceived by game developers and companies behind video game consoles and major game publishers. To regain the simplicity that allowed a new user to simply "get a game and play it", new control methods were explored, like motion controls (Wii, PS Move, Kinect), sensors (Wii Balance Board) and others. In a similar time frame, the rise of smartphones started, quickly followed by tablets. The computing power on these mobile devices quickly improved, allowing the launch of several mobile games with complex visuals and refined interactions.

A major difference was that these devices rely primarily on touch input, done on a large screen that normally covers the entire device front. Besides that, they gained several different sensors, initially to perform specific objectives, like GPS (for map applications) and accelerometer (automatic screen rotation). Game developers soon started using these data sources as input methods, like using the player's location to trigger actions in games or using the accelerometer to control a game avatar using motion controls. These new technologies, both on mobile devices and in traditional gaming machines, were the starting point for a new wave of games using more intuitive control schemes, bringing millions of players that did not wanted to learn complex control schemes to electronic games. Unfortunately, the new control schemes have limitations. Traditional controllers, while complex, are reliable and precise. The new methods suffered with lack of precision in command input, slower reaction time and limitations when trying to map complex actions. The problem was clear in several mobile games, that started using virtual buttons displayed on its touchscreen to simulate a traditional controller and all of its input options. This solution had some problems. The biggest one is that a virtual controller on a screen lacks the tactile feedback of a real controller, where the user can feel the button. It's not uncommon to miss virtual keys when trying to input commands or having to look constantly to the virtual gamepad to confirm where the virtual buttons are. But a virtual controller has some advantages too. Every game on a mobile device can have its own virtual controller that will have the exact amount of buttons needed for that game. A traditional controller need to have all necessary buttons for any game, in a way that most of the games will use only a portion of these buttons, leading to an unnecessarily cluttered interface.

The above conclusions were the start of this work, a proposed new interface for electronic games, that has the large input options of a regular gamepad but with the flexibility of a virtual controller. This way, the new controller has all the varied input capabilities of the regular controller but is capable of having only the buttons that are necessary for the current game. This new input method aims to replace the traditional video game joystick with a touchscreen device with a virtual controller on its screen, creating a less cluttered and more intuitive interface to play video game. Beyond the fact of having only the necessaries buttons, the interface is capable to have only the required buttons for each moment. It can be exemplified by a game that initially use only some buttons and a long of the game, the character learns extra skills, and new buttons are needed. Instead of displaying a useless button, minimizing the useful portion of screen and probably minimizing the size of all buttons to fit on screen and even inducing to a bad usability, our work suggests a new paradigm that provide the controller necessary to a game in each situation. In this approach, the game developer is able to deal with the controller at any time, to change the interface to fit to the game, creating an adaptive interface related to the game context.

Contributions: This work proposes a framework which permits the game developer to construct and/or adapt the control of digital games on touch screen mobile dynamically. The framework is adaptable, allowing virtual keys to be proper adjusted in the device based on game context in order to change dynamically some control attributes such as create, delete, resize and/or define a new position for a button. The control and the game are implemented as a case study of this new way of interaction based on game context.

Paper summary: the remainder of this paper is organized as follows. Section 2 presents some related works on mobile devices as inputs and adaptive interfaces and Section 3 describes the framework architecture. Section 4 presents the case study. Finally, in Section 5, the conclusions and future topics to research are discussed.

2. Related Works

As this work use a mobile device as input to games that has the controller adapted dynamically, the related works are subdivided in two subsection : Mobile devices as input and adaptive interfaces.

2.1 Mobile devices as input

Mobile phones have specific hardware (camera, accelerometer, GPS, bluetooth and so on) with lots of them being different from the ones found in traditional game platforms, like video games and PCs. For this reason, these devices bring new forms of user interaction [Joselli et al. 2012a] and some of them will be detailed in this section to provide an useful background for this work.

Nowadays, smartphones have reliable speech recognition which can be used for game input. This type of voice recognition is still a processing expensive task in this kind of devices. However, there are other forms of speech recognition in mobile phones, like the use of a speech server using voiceXML. The use of voice in mobile games, and in games in general, has been widely explored. In Zyda et al. [2007] voice in mobile games is explored using a voiceXML server. Zyda et al. show the application of voice in different aspects of the game: for game command input, for chatting in a multiplayer game and for accessing exclusive content in a game.

Mobile phone touch screen devices are very common. Most devices that have such features possess few buttons, with almost all input made by touch. This way, mobile touchscreen games must be designed to accept most of their input by touch, that will be be used in games in a similar way to mouse clicks on a regular computer, allowing developers to make different types of mobile games based on virtual buttons.

Consequently, the screen can draw buttons and use it to simulate button input [Joselli et al. 2012b]. Another type of input is through touch gestures, as presented in a First Person Shooter mobile game [Wei et al. 2008].

Several researches in the field of human interaction have been developed to use the mobile devices as well as their sensors. As a result, game applications are employed to help rehabilitation or as a complementary therapy, instead of being used for fun [Golomb et al. 2010; Burke et al. 2009; Laikari 2009; Brandao et al. 2010]. In human interaction research field, two groups are well explored: sensors and computer vision. In Stenger et al. [2010], computer vision is adopted, so that it can recognize a pattern to be employed as a player's input.

Mobile phones are already been proposed as input for simulations. The Poppet Framework [Vajk et al. 2008] can be used as a simulation control that can send and receive data information to and from simulation, respectively. To perform this, the framework uses Bluetooth communication. The movement applied to the mobile phone, detected by its accelerometer, is sent to a central control server that does the player’s avatar movement.

Malfatti et. al [2010] propose the BlueWave framework for using mobile phones as simulation input through Bluetooth technology. BlueWave is implemented using Java technology, requiring a java enabled mobile phone. Unfortunately, the disadvantage of the proposed framework is that it uses the mobile phone hardware keypad for control input, which can be completely different from a mobile phone to another (and is not valid for more modern devices, that does not have physical keys). In this case, a key layout that could works well for a mobile phone could not work appropriately for another one, causing frustration for the player.

2.2 Adaptive Interfaces

An adaptive user interface is an interactive software system that improves its ability to interact with a user based on partial experience with that user [Langley 1997]. Adaptation of interactive systems describes changes of the interface that are performed to improve the usability or the user satisfaction.

The observation of user behavior is a prerequisite for performing adaptations. Based on an observation of basic events, such as button presses, speech input, or internal state changes, user preferences are derived. Different algorithms extract information from these basic events, such as preferences of the user or a prediction of the most likely following user action.

As computer systems are getting more and more used by different type of people, considering age and experience in this field, researches about interfaces that try to help the users and improve the usability are always on focus. So, Rogers et al. [2010] developed models that treat uncertain input touch and use this to deal with the handover of control between both user and system. They demonstrate a finger map browser, which scrolls the map to a point of interest when the user input is uncertain. Keeping the same goal, but in a different way, Weir et al. [2012] used Machine Learning approach for learning user-specific touch input models to increase touch accuracy on mobile devices. They proposed mapping data or touch location to the intended touch point, based on historical touch behavior of a specific user.

Bi and Zhai [2013] conceptualized finger touch input as an uncertain process, and used statistical target selection criterion. They improve the touch accuracy using a Bayesian Touch Criterion and decrease considerably the error rate. But, even improving the accuracy in a higher level, the user keeps missing the buttons and, the interface needs to calculate the intended target.

Personalized inputs have been commonly used, for example key-target resizing on soft keyboards [Baldwin and Chai 2012], but this type of adaptability has some disadvantages, as the needs of using only on meaning of process language, and it just adapt according to model of language and of user typing behavior. Our work provides a framework for adaptive controller, which allows game developers to design a game controller that best fit to each game's context. The developer, in addition to create the game, can change the controller. These changes are executed in real-time, due some moments, like a mini-game inside the game(e.g. Mario Party) or an enemy approaching. So, from these contexts, the developer can decide what the controller might looks like, for example, he can decide to change completely the layout, add, remove, increase or decrease some buttons.

This work is an extension of Zamith et al. [2013] where a framework for building mobile game controls adaptive is presented. In this previous work, the adaptability was related to machine learning and statistical analysis over the inputs gathered by the client. To realize this type of adaptability, the client sends the buttons pressed and the mistaken buttons. With these data, the server ran the k-means algorithm [Smola and Vishwanathan 2008] and suggests where each buttons should stay to increase the button's hit rate. The major difference between this work and Zamith et al. [2013] is in how the adaptability works, one in the game and other in the client/server, which the adaptability is extremely different because one uses the game context and the other uses history of touches on buttons and wrong touches.

3. The Proposed Framework

The proposed framework can be divided in three main blocks: client, server and game, as presented in Figure 1. In this division, each one has a particular and necessary main functions. The client is responsible for all data acquisition, which in our proposal corresponds to a mobile device. The data can be acquired by accelerometer and touch in buttons that are, first of all designed by the game developer and then dynamically drawn at the touch screen, through the game context. The server is responsible for delivering data between client and game. To perform its function, the server synchronizes the game and the controller state. And lastly, the game is connected with the server in order to update the current state and to inform its new state to the server. Fig. 1 illustrates our proposed framework.

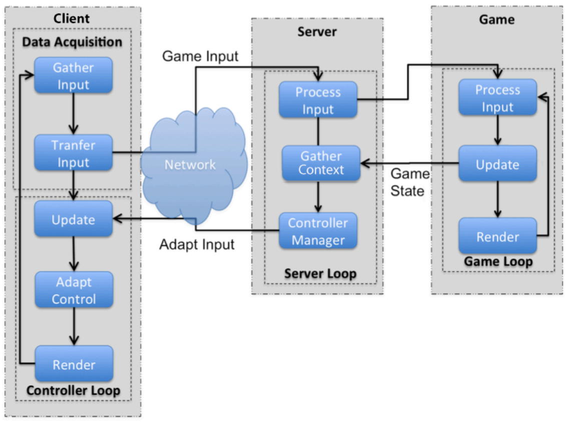

Figure 1: Framework Architecture

In this work, the client is based in Zamith et al. [2013], which is a software that runs on the device used by the player to interact with games, but the connection method used differs from the previous, where it was TCP/IP and Bluetooth. This framework, provides a connection between client and server only using Bluetooth. Since the mobile device is able to play sound, vibration as well as render, these features are also employed by client component in order to provide feedbacks to the player.

The server component is responsible for processing the user connection requests and the data input. It works as a layer between the game and the game control. In this architecture, these data are received by the server and sent to the game. The server synchronizes, for each frame, the game state with the controller state.

The game, in this case, is the most important component. Inside the game loop, when the game updates, it is able to send it's context to the server. So, for each frame, it can send or not the context, asking the client to update or not the interface. We pre-defined four methods that the game is capable to call according to its context. The methods are: create, remove, change size and change position of a button. So, in one step, the game is able to call all four methods and multiple times of them, allowing to change the role interface.

In order to make all methods to work in any device, the game does not know the screen resolution, working only with percentage of resolution and the client apply this value to it's screen. The position and size of the game cannot extrapolate screen size, causing loss of useful area. So, we estimated and do not permit that a button exceed the size of 25% of the screen. This value is very high and was empirically estimated to prohibit that a button gets too large and then cause an usability problem. If a button stays extremely large, the other ones that borders it may not increase their size in case of this large one does not decrease. This rule is important because a button cannot overwrite the other ones, so it can just increase size, change position or even create a new button if it won't overwrite any part of some button.

The client, as mentioned above, is responsible for sending input data to the server, which transfers it to the game. The game executes its game loop and when updating the state, if it is in context of modifying the controller's interface, the server receives this information from the game and transfer to the client. Then, the client analyze if it is possible perform this change, and do it or don't, according to the established constraints. After rendering the new interface, the client can also present a feedback, such as a sound or vibration. This loop remains until the client ask for disconnect .When this happens,the server informs to the game, which closes and disconnect. This sequence is made in every game that is developed with our framework. The unique deviation is if the interface should or not be modified on both game(when it request) and on mobile(when it can or not adapt).

Each button of the interface is represented by a finite-state machine (FSM), according to Figure 2. There are four states, which coincides with the four methods that our framework implements, and they are C, R, P and D. The initial state is C, which means the state of button creation. There are also the state R, that represents resize, P that is the position change state, and D the delete state. As their names suggests, every button begin as a C and can stay forever as C, but if the game asks for resize the button, state R is achieved, even as the state P can be achieved if the game asks for change the current position of this button. Even in cases that a same button comes back again to the interface, we assume that a new button is created and then it's state is C.

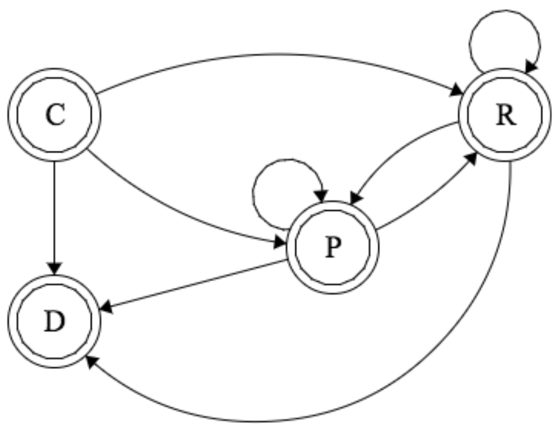

Figure 2: Button FSM

3.1 Framework Application

This framework is used to give the game the chance of "preview the future" and then help the gamers to achieve more points, increase accuracy, provide a better use of the mobile screen showing and using only buttons that has meaning in the immediate situation of the game, beyond others advantages. This might be a great tool to some specific group of users, for example old adults and users impaired disabled.

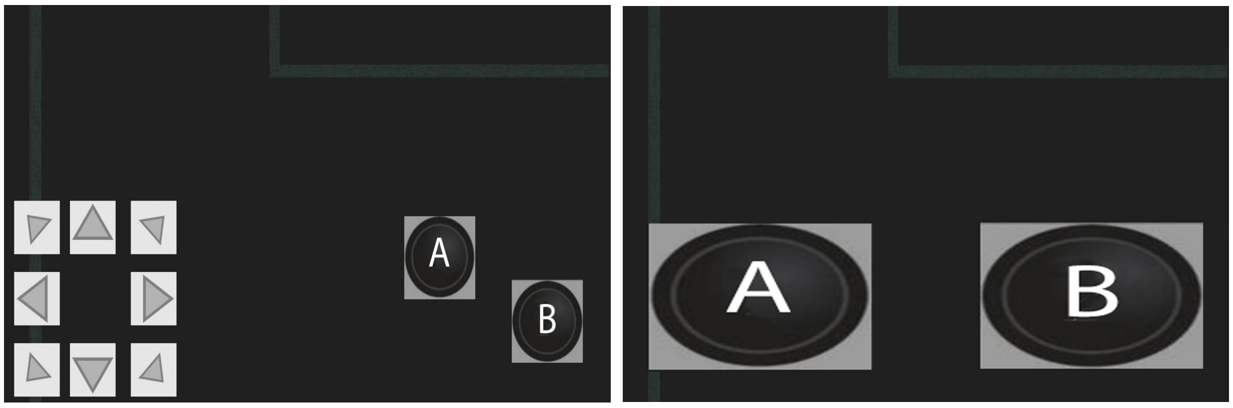

Figure 3: Basic interface on the left and the interface adapted when just buttons A and B are required

The feature that allow a button to increase the size some seconds before it is needed to be pressed, might help users that want to play, but for some disability takes longer to recognize what action is needed or even knows what have to press but click outside the button due the small button size. The features of create and delete a button help to construct a succinct interface, due the ability to show just the useful buttons and hide them when not necessary anymore. The repositioning is good to associate the game to the controller more intuitively, for example games that the user needs to press the button in a sequence as the Genius or mini games like Mario Party, which the gamer often needs to press two buttons (e.g. A and B), he could press each button with one hand's finger instead of the same finger pressing two buttons, improving his result and the game playability. Figure 3 illustrates this example, where the left interface is the usual, but as the game requires just A and B,in a specific moment, all the unecessary buttons are excluded, creating a more a intuitive and clean interface.

4. Case Study

The proposed framework was implemented using the Java language, being developed as a library that can be added to any game project and allows the game to collect the input data from the joystick and to perform modifications in the controller's layout dynamically during the gameplay interaction. All requests for controller changes will be passed to the mobile component that will evaluate if the change is possible and will try to perform it as best as it can. This kind of decision is made in the controller to guarantee that the layout won't achieve an inconsistent configuration like buttons that overlap with each other or that are rendered out of the screen. An example of this kind of intelligence would be if the game requests that a button increases its size by a factor of 2. Instead of denying the request, the controller will grow the button as much as it can without causing the overlap, effectively trying the best it can to follow the commands given by the game.

To demonstrate our work and generate a real study case, it was developed a game to make use of the proposed framework, called Aliens Front. This game is a 2D adventure, shown in Figure 4, where the player controls a warrior and has to pass through a horde of enemies. As usual, the player is capable of hitting with a sword to destroy enemies, that will be trying to destroy him with their weapons as well. The player will have to react fast to avoid threats and hit enemies. Special power-ups will be available in some parts of the games to perform a stronger attack that is capable of destroying several enemies. These kinds of characteristics allows several possible improvements as we will see next.

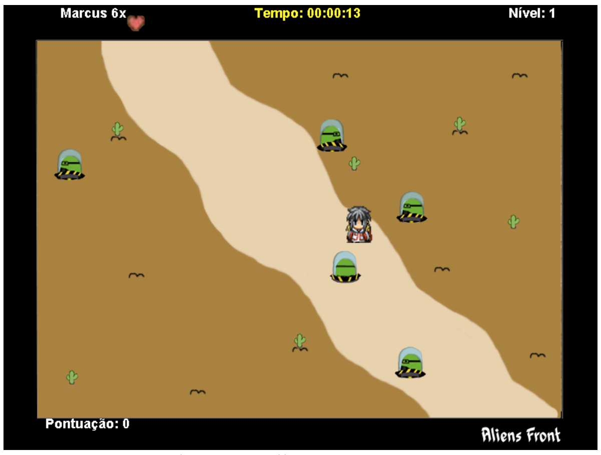

Figure 4: Aliens Front game

In this game, the control scheme uses a directional button or D-Pad, an attack button and a power-up button, used to activate the special power-up when it is available, positioned following a traditional joystick layout, with the D-pad at the left side of the screen and the action button in the right side, all of them positioned close to the bottom of the screen. The game uses the framework, calling the libraries methods to control and communicate with the virtual joystick. The first step that the game accomplishes is to initialize the Bluetooth library and wait for a controller. After the virtual controller, running on the mobile device, successfully connects to the PC, the game is notified by the framework. With the connection established, the game can define, using the proper classes Joystick and Button provided by the framework, the position and amount of buttons on the controller. The mobile device will receive this data and display the buttons in the correct positions. It's important to remember that the game can request at anytime that the controller wants to remove or add new buttons or change the position or size of any of them. This kind of flexibility allows the game to not only create an ideal layout but to change it at any moment to adapt the controller to the challenges on the game.

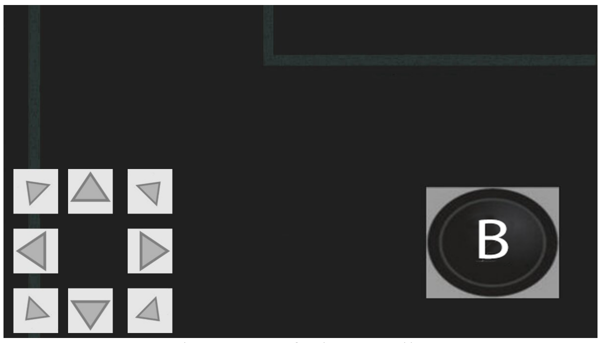

Figure 5: Default controller

Aliens Front makes use of this characteristic by initializing a normal controller with a D-Pad and an attack button, presented in Figure 5. In this game, it's very important to be able to quickly evade enemy fire and contact. To help the player with this task, often complicated on sections full of enemies, the game determines the best escape route for the player and grows the buttons on the D-Pad that represents the best directions to avoid the hazard. The decision is simple: if the hazard comes from the right side of the screen in the player direction, he will need to evade vertically, so the D-Pad will present bigger buttons for "Up" and "Down", as shown in Figure 6. If the danger comes from behind or above, it will be necessary to move the character horizontally, resulting in bigger buttons for "Left" and "Right" like Figure 7. These adaptations will probably happen without the player even noticing, but they demonstrate how the developer can use the ability to change the controller to improve chances of success for the player.

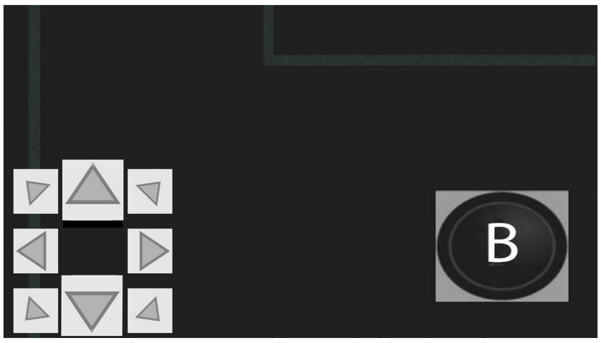

Figure 6: Controller vertically adapted

However, there is another capacity of the controller being actively used by the controller. When describing the control requirements for this game, it was necessary to use a D-Pad and two action buttons: attack and power-up. But the game actually initialized the controller with a single action button used to attack with the sword. That's because while the attack button will be used pretty much during the entire game, the power-up button only is useful when the player gets one and, after using its single powerful magic, it won't be needed until another power-up is available. So, the game will only add a power-up button when the player can use it, as Figure 8, and will remove the button as soon as it is no longer needed. A simpler interface, that only contains the necessary buttons for the given gameplay section, results in a more intuitive control scheme and a more casual-friendly game. Of course, this is a simple example, but in more complex games that use sometimes 8 or 10 buttons the capacity of displaying only the necessary buttons for a section can simplify considerably the interface and provide more screen space for the buttons that will be really used in a given moment.

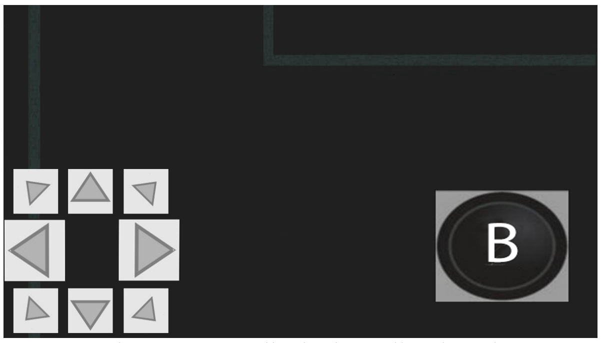

Figure 7: Controller horizontally adapted

All of these changes are simple to be implemented on the game. The only work needed is to call the methods from the framework to request a change in the controller. The bluetooth communication, serialization of data and all the necessary steps to the pass the changes to the controller are handled in the framework, allowing the game developer to concentrate on its own logic and in determining how it wants the controller to be in order to provide the best gameplay experience customized for its needs.

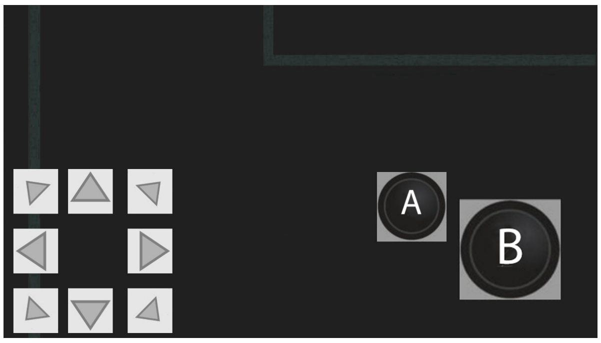

Figure 8: Controller with all buttons

5. Conclusions and Future Works

Nowadays, several ways of user's input are being presented by the game industry to attract more players enhance the immersion during the gameplay. And actually, some games have a complex input system, which may avoid users to play. As this work presents, using a mobile device as the game input, which is a hardware that is known by the user, and making it the most simple possible at the game context and even helping the gamer in specific moments, can give opportunity to users that may avoid complex input system to try and possibly enjoy playing games using their mobile devices. And, this could eliminate the enter barrier found in novice game players.

This work shows how the game developer can create the specific controller to his game, and then take advantage of that, modifying the layout according to the game context. This feature can be used in different ways, focusing in help users and change buttons' layout. And this is a property that, is not seen in traditional game controllers and many related works on game input.

The authors would like to investigate in the future, how this framework behaves in different kind of users applying an usability test varying the age, sex, experience with games and mobile devices. In the field of accessibility, an investigation of how this kind of control could improve the game interaction of older adults and users with disabilities that have some motor skills limitations or even by augmenting the gameplay experience of such users.

As this work is an extension of an previous work, this framework might be unified with the previous to originate a better game input, taking advantage not only by game context but also by history of touches, using machine learn techniques.

The authors also pointed the study of other ways of adaptability, changing not only the button size and position, but also the shape. This would be a great advance, including the study about user and interface behave after modify shapes of buttons.

Acknowledgements

The authors gratefully acknowledge NVIDIA, FAPERJ, CNPq and CAPES for the financial support of this work.

References

BALDWIN, T., AND CHAI, J., 2012. *Towards online adaptation and personalization of key-target resizing for mobile devices*. In IUI '12, 11–20.

BI, X., ZHAI, S., 2013. *Bayesian Touch – A Statistical Criterion of Target Selection with Finger Touch. ACM UIST '13* 51-60.

BRANDAO, A., TREVISAN, D., BRANDAO, L., MOREIRA, B., NASCIMENTO, G., VASCONCELOS, C., CLUA, E. AND MOURAO, P., 2010. Semiotic inspection of a game for children with down syndrome. In: Games and Digital Entertainment (SBGAMES), 2010 Brazilian Symposium on, nov. 2010, 199 –210.

BURKE, J. W., MCNEILL, M., CHARLES, D., MORROW, P., CROSBIE, J. AND MCDONOUGH, S., 2009. *Serious games for upper limb rehabilitation following stroke. In: Proceedings of the 2009 Conference in Games and Virtual Worlds for Serious Applications, ser. VS-GAMES '09.* Washington, DC, USA: IEEE Computer Society, 103–110. [Online] Available from: http://dx.doi.org/10.1109/VS-GAMES.2009.17 [Accessed 20 July 2014].

GOLOMB, M. R., MCDONALD, B. C., WARDEN, S. J., YONKMAN, J., SAYKIN, A. J., SHIRLEY, B., HUBER, M., RABIN, B., ABDELBAKY, M. E. NWOSU, M., BARKAT-MASIH, M. AND BURDEA, G. C., 2010. *In-home virtual reality videogame telerehabilitation in adolescents with* *hemiplegic cerebral palsy. Archives of Physical Medicine and Rehabilitation,* vol. 91, no. 1, 1 – 8.e1. [online] Available from: http://www.sciencedirect.com/science/article/pii/S000399930900817X [Accessed 21 July 2014].

JOSELLI, S., JUNIOR, J.R.S., ZAMITH, M., CLUA E. AND SOLURI, E., 2012, *A content adaptation architecture for games. In: SBGames. SBC.*

JOSELLI, M., SILVA JUNIOR, J. R., ZAMITH, M., SOLURI, E., MENDONCA, E., PELEGRINO, M. AND CLUA, E. W. G., 2012. *An architecture for game interaction using mobile. In: Games Innovation Conference (IGIC), 2012 IEEE International*, August 2012, 73–77.

LANGLEY, P. Machine learning for adaptive user interfaces. 1997. *In KI-97: Advances in artificial intelligence, Springer Berlin Heidelberg*. 53-62.

LAIKARI, A., 2009. *Exergaming - gaming for health: A bridge between real world and virtual communities. In: Consumer Electronics. ISCE '09. IEEE 13th International Symposium on, may 2009*, 665 –668.

MALFATTI, S. M., DOS SANTOS, F. F. AND DOS SANTOS, S. R., 2010. *Using mobile phones to control desktop multiplayer games. In: Proceedings of the 2010 VIII Brazilian Symposium on Games and Digital Entertainment, ser. SBGAMES '10*. Washington, DC, USA: IEEE Computer Society, 74–82.

ROGERS, S., WILLIAMSON, J., STEWART, C., AND MURRAY-SMITH, R., 2010. *Fingercloud: uncertainty and autonomy handover in capacitive sensing. In ACM CHI '10*, 577–580.

SMOLA, A. AND VISHWANATHAN, S. *Introduction to Machine Learning. Cambridge University,* UK (2008) 32- 34.

STENGER, B., WOODLEY, T. AND CIPOLLA, R., 2010. *A vision-based remote control. In: Computer Vision: Detection, Recognition and Reconstruction,* 233–262.

VAJK, T., COULTON, P., BAMFORD, W. AND EDWARDS, R., 2008. *Using a mobile phone as a wii-like controller for playing games on a large public display. Int. J. Comput. Games Technol., vol. 2008, January 2008, 4:1–4:6.* [Online]. Available from: http://dx.doi.org/10.1155/2008/539078 [Accessed 21 July 2014]

WEI, C., MARSDEN, G. AND GAIN, J., 2008. *Novel interface for first person shooting games on pdas. In: OZCHI '08: Proceedings of the 20th Australasian Conference on Computer-Human Interaction, OZCHI.* New York, NY, USA: ACM, 113–121.

WEIR, D., ROGERS, S., MURRAY-SMITH, R., AND LÖCHTEFELD, M., 2012. *A user-specific machine learning approach for improving touch accuracy on mobile devices. In ACM UIST '12,* 465-476.

ZAMITH, M., JOSELLI, M., SIVA JUNIOR, J., PELEGRINO, M., MENDONÇA, E., CLUA, E. *AdaptControl: An adaptive mobile touch control for games. In SBGames 2013,* 137-145.

ZYDA, M., THUKRAL, D., JAKATDAR, S., ENGELSMA, J., FERRANS, J., HANS, M., SHI, L., KITSON, F. AND VASUDEVAN, V., 2007. *Educating the next generation of mobile game developers. IEEE Computer Graphics and Applications,* vol. 27, no. 2, 96, 92–95.