Application of Motion Detection and its Design Principle in Next-generation ACT/ARPG Games

Yinchen Xu
Stanford University
Department of Computer Science
yxu72@stanford.edu

Wanling Liu
Stanford University
Department of Electrical Engineering
liuwl@stanford.edu

Abstract

With the development of sensors and tracking technique, the cost of detecting users certain action in VR (Virtual Reality), AR (Augmented Reality), or traditional-display devices becomes lower, which makes it possible to fuse action detection into the control of different software. On the other hand, the design of VR environments always pursues a high degree of immersion for users, which can be perfectly achieved by exploiting action detection in the design of user interactions. In this project we explored what we should mention when applying some common types of action detection in general, provided some examples in the form of an ARPG (Action Role-Playing Game) game, and applied the methods on a non-virtual-reality game, Sekiro Shadows Die Twice, to show the potential of game design.

1. Introduction and Motivation

Recent development of sensors and tracking techniques makes the detection of simple action made by the users easier and more efficient. Oculus Rift, HTC Vive, and many other virtual reality devices become more and more suitable for the users' natural gestures, in order to feasibly gather the motion of the user accurately and fast and also improve the user's experience. Based on the development of VR technologies, it must be true that, there will be many games in traditional gaming genres being modified, improved and fit into the VR environment. ACT/ARPG game is a perfect example.

1.1. Design Principle of VR Environment

The design of VR environments requires designers to think comprehensively about immersion and interaction. In detail, the sense of immersion is required to attract users' attention, and the design of interaction is required to physically complete the goal of the corresponded software. Both properties should be guaranteed for a good VR environment [4]. Designers can exploit all possible aspects that would interact with the users, including but not limited to visual rendering, audio design, users input in the virtual world, and users movements in the real-world.

1.2. Design Principle of ACT/ARPG Games

The design of ACT and ARPG games pursues the sense of hitting and player's high intensity interaction as the keys to the quality of the game. Therefore, player's sense of immersion can improve their experience playing these games. The design of good ACT/ARPG games can involve aspects including but not limited to the design of input settings, the animation of actions, the audio and visual effect.

2. Related Work

There are many existing researches about motion detection (e.g. [3], [1]) and its application in software, and even games. Nintendos Wii and Kinect [5] are good examples and the combination of motion detection and the computer is efficient for general-purpose software. However, there are few attempts to apply motion detection techniques to ACT/ARPG games, partly because it is probable that the resulting products would be tiring and inefficient.

The property of ACT/ARPG games determines that the input design should preserve low latency, high immersion and independence of each input [2], which should foster new ideas in the application of motion detection. It is noticed that with the further development of motion detection techniques, it is possible to solve this problem with a good design of input semantics, which will be the concentration of this project.

With the development of VR devices and developing tools, there already exist several famous and well-designed action games. *Beat Saber* is an action game based on music and player's action. The game fuses the element of music and player's action by generating cubes which would reach player's location following the tempos of the playing music.

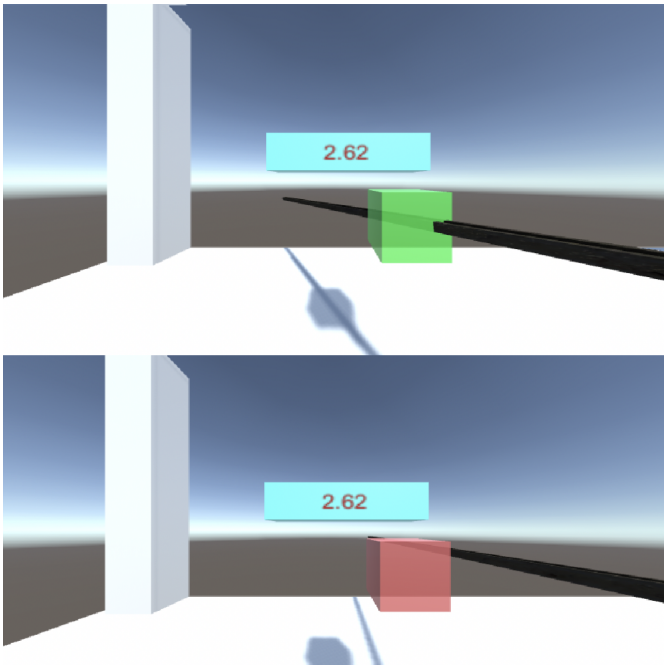

Figure 1. The scene used for testing the users' response of hitting a virtual object when the counter counts down to 0: the green box denotes that the box is hit, while the red box denotes that the box is not hit by the participant.

3. Content

Our work contained both research and real-world implementations. More specifically, our work can be divided into three parts. In the first part, we conducted experiments of user response accuracy for several kinds of VR user inputs. In the second part, we used Unity to implement a VR game. In the last part, we mapped the inputs of a non-VR game, Sekiro - Shadows Die Twice, to make the game more immersive for users and also explore the potential of extending non-VR game to VR game. We used the Oculus Rift with Oculus Touch Controllers as our hardware (refer to Figure 2 for usage explanation of the controllers).

3.1. User Test

In the first part of our work, we conducted several kinds of experiments to do research on the user response of different VR inputs. More specifically, we tested the response accuracy of different kinds of motions: hitting a specific target in the virtual scene, squat, swing, and pressing a button. In the pilot testing, we gathered information from 10 users, collecting and evaluating their results.

We designed a simple system to let the participants to interact with the virtual scene (refer to Figure 1 for an example scene for the participant). We set a counter for the participants and asked them to do some actions when the counter counts down to zero and they would react accurately as possible as they can. We chose four actions to test: hitting a specific target in the virtual scene, squat, swing, and pressing a button, as we mentioned before. Those tests are to experiment with the users' response of different actions in VR and hopefully we can get some useful findings for future game design.

3.2. Game: Battlefield

In the second part, we realized a game using Unity and we named it Battlefield. The scene of our game was made based on an existing scene from the unity assets, and we combined materials downloaded from the internet (bombs, firing effect) and our self made components (snowballs, the rod, the scoring display and some interactive buttons).

We composed the game controller that controls all the elements in the whole scene, which works as the hub of the game. We also set the mechanisms for the interactive elements in the game, defining their behaviors including the interaction with the player's controller configuration and phase. We set a separated collision layer in Unity to make sure the game mechanism does not conflict and is not influenced by the surrounding objects and the OVRController-Prefab object.

In this game, the player would be equipped with a pair of controllers (for the left and right hands) of Oculus Rift (refer to Figure 2 for the usage of the hardware). For the game setting, the player is a "defender" who will hold a shield with his/her left hand and a rod with his/her right hand to defend potential several kinds of attacks. The game is set in a "battlefield", where there would be bombs and snowballs coming from random directions to hit the player. Additionally, we set a scanner to scan the player to increase the difficulty of the game. The game mainly contains three systems: shielding system, hitting system, and hiding system. The goal of the game is to "live" longer in the game. Each player in each round would be given four chances to keep themselves from attacking. Players need to raise their left hand in order to utilize the shield (open the shielding system) to prevent themselves from the attack of the bomb, use their right hand with which they hold a rod to hit the snowball. To make the situation more realistic, we decided to ask the user to hit the snowball with some velocity above a threshold or their action is ineffective. Additionally, the player should protect themselves from being scanned, so the player need to cross their hands (put the left and right controller within a close distance) to activate their "hiding mode" to hide from the scan.

3.3. Sekiro Input Mapping

In the third part, we mapped the VR device inputs with the inputs of a non-VR game, Sekiro Shadows Die Twice.

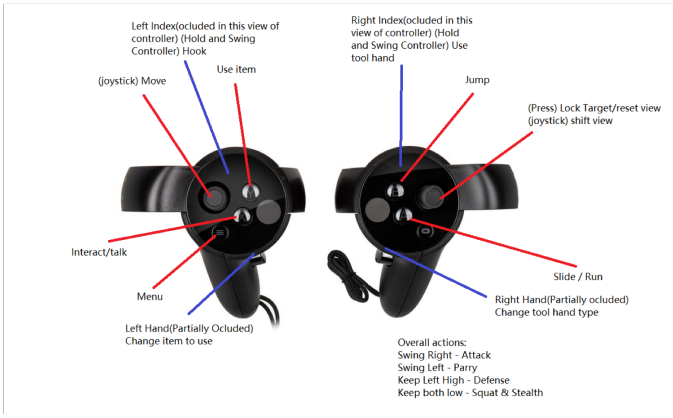

Figure 2. Controllers of Oculus Rift

Our design is to use a combination of buttons and actions of the Oculus Touch controllers instead of the traditional mouse-keyboard or game controller, which creates higher degrees of freedom for users to interact with the game mechanics and introduces a higher extent of sense of immersion when playing the game.

The pipeline of the remapping involves three parts. First we detect the action of the player with the tracking sensors of the controllers and the headset, which provides us with the relative positions and the velocities of the controllers in the local coordinate. Combining with the key press configuration at the frame, we determine what kind of action is player trying to perform.

Second, based on the expected action we justified, we deduct the corresponded signal needed to be sent to the computer. The computer recognize the generated signals as virtual key press on the keyboard and if we use the original hex scan key code, those signals can also be recognized by DirectX, the module used to run most of the large-scale game in Windows.

Third, since the needed type of key press is different in duration and key release also requires an additional signal, we use the Invoke() function and startCoroutine() function in Unity to set delayed function calls. In general there are two types of timing structure, the first type requires a key press for a relative short but longer than 1 frame time. For this type of key press, we set a fixed release function and a corresponded flag variable used to freeze the detections lead to a result with pressing this key. The second type requires a continuous key press. For this type, when we detect the start signal of action, we start the key press, start the detection of the ending action, and freeze the detection of duplicated start signal which may disrupt the virtual input configuration. When we detect the ending action, we release the corresponded key and switch back the action detecting target actions.

4. Results and Evaluations

In this part we would give both qualitative and quantitative evaluations of the results gathered from our user test. We would also show our games and evaluate and discuss the our game implementations.

4.1. User Test

As for the user test part, we conducted experiments to explore the accuracy of users' response of different actions as we have explained in Section 3.1.

In Figure 3, we show the distribution of difference in time to the exact $t = 0$ of each user's attempt. We can get the mean value and standard deviation as shown below in Figure

From the statistical result (mean value: refer to Figure 6, and standard deviation: refer to Figure 7) below, we can see an obvious tendency that it is harder for a user to perform accurately in time if he/she tries to swing the controller at a speed or squat with the controllers, also the performance from different users varies more for those actions. Controlling a rod in the virtual environment with the right hand controller to hit a box in the virtual environment leads to less inaccuracy, yet pressing button A gives the best result.

The high inaccuracy of swing and squat may be the result of multiple reasons. Possible reasons include the limitation that the user cannot clearly see the velocity threshold or the displacement threshold directly from what they see, the fact that these operations require the users to continuously posing their actions, and that some of the testers are not familiar with the controllers. These statements can also explain the relative better accuracy of hitting the box and the best accuracy of pressing the button: User can directly see the distance between the rod and the trigger region in the virtual environment; pressing the button only involves an instant action and since users were notified with the upcoming action, they do not need to see the location of the button and find it before action.

It is also noticed that the learning effect of more complicated action is less obvious than that of simple actions. The quantitative comparison can be more obvious if we let users test on the same actions for even more times within one evaluation. The graphs below45 record the three attempts of a user performed in one user evaluation. We can see that the accuracy of pressing the button increases dramatically yet there is no obvious change in performance when he was performing swing.

4.2. Game: Battlefield

Figure 8 to Figure 13 show our scenes of the game. In Figure 8, we can see the general scene of our game: a player would be equipped with a shield in the left hand, a rod in the right hand. Bombs (the purple one) and snowballs (the

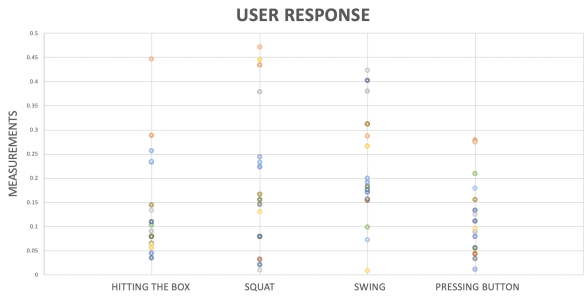

Figure 3. Users' response of different inputs.

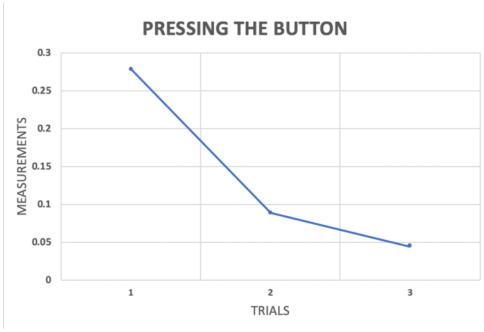

Figure 4. An example of the user's response to pressing the button.

Figure 5. An example of the user's response to swing the controller.

white one) would come to the player from random directions and the player need to raise his/her left hand to activate the shield to protect himself/herself from being attacked by the bomb (refer to Figure 10 and Figure 9 for what the different states of the shield would look like). As for the snow ball, the players need to use the rod in their right hand to hit the snowball with a certain velocity to break down the snowball. If the player misses one bomb or snowball, he/she would lose one chance, and the number of total chances is four, which is denoted by four red squares floating in the upper left part of the scene (those squares can be seen from

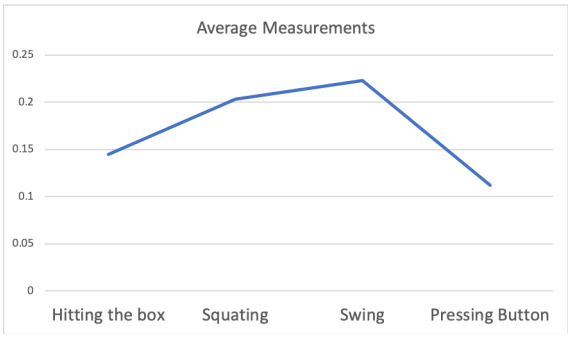

Figure 6. Average measurements of four inputs.

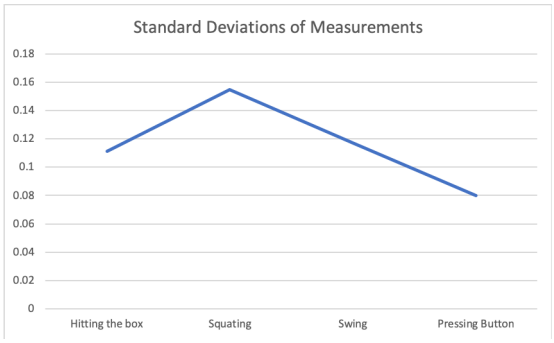

Figure 7. Standard derivation of four inputs.

Figure 9). We also set up a scanner to make the game more difficult (refer to Figure 11). The scanner would scan the scene and the player needs to put his/her hands close to activate the hiding state, which allows the player to escape from scanning. If the scanner detects the player, the player would lose one chance. To make the game more interesting, we added some special effects of the game: the explosion of the bomb and the collapse of the snowball (refer to Figure 12 and Figure 13).

We tested our game thoroughly, and from the feedback of the users, we think we have realized the full desired functionality of this Unity game.

4.3. Sekiro Input Mapping

Since the program need to be run in background, we can fetch the latency of the whole detection procedure easily. When there is no input given to the Touch controllers, the update rate of the program is near 100 fps, which is faster than the 60fps game setting. In most of the cases, When the player gently posing action to the program, due to a scan of some keys involved in continuous pressing, the update rate of the program can dropped to near 50fps, which is also a good result. In the worst case, when the player tries to input all possible instructions to the avatar in the game through

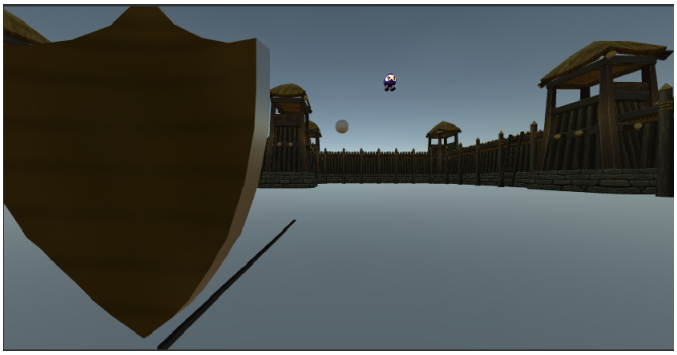

Figure 8. The scene of our game.

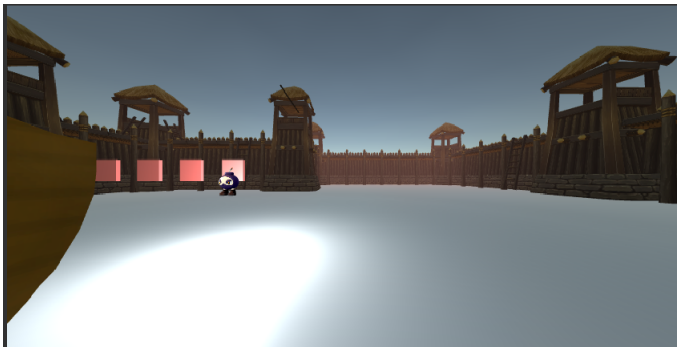

Figure 9. Shield state: inactive

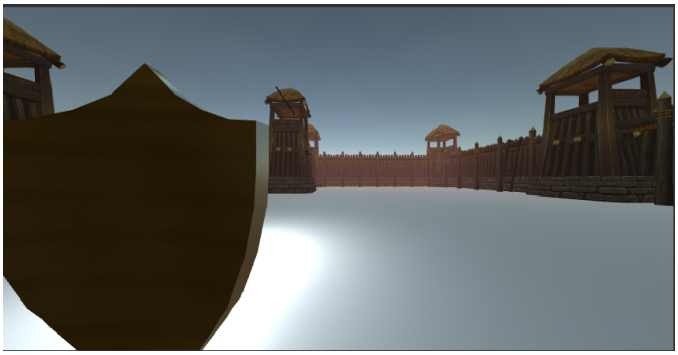

Figure 10. Shield state: active

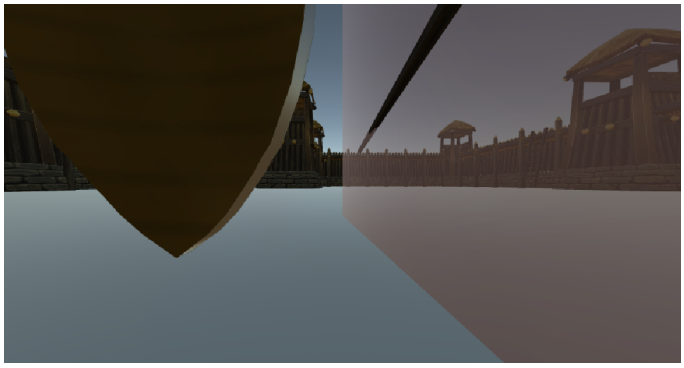

Figure 11. The scanner in our game.

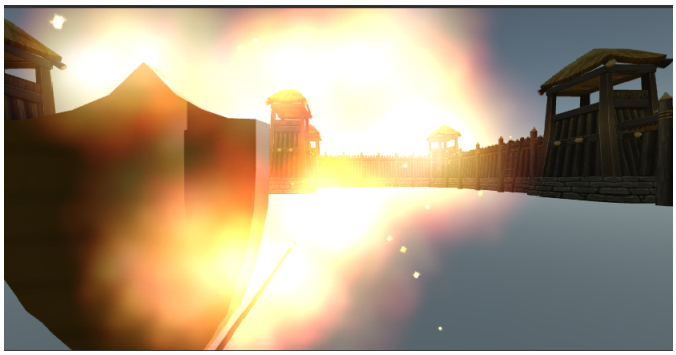

Figure 12. The effect of bomb explosion.

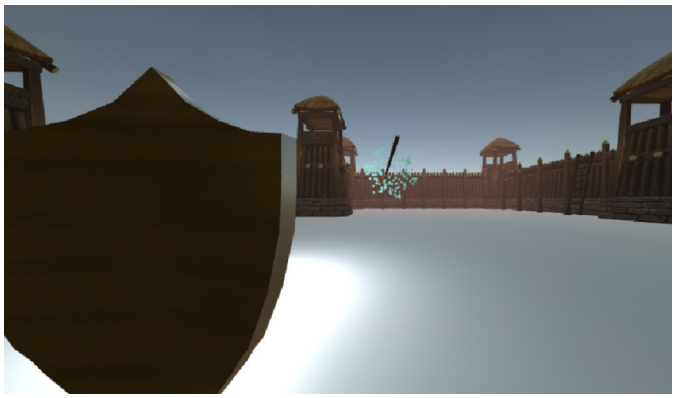

Figure 13. The effect of snowball collapse.

satisfying multiple action detection rules in same frame, the latency would drop to between 36 to 40 fps, but since this latency mainly occurs after the key press system call and the frame rate of the game is still preserved, it is still acceptable.

From the perspective of the sensitivity, in the demo session the first-time players can also complete the expected operations in the game with no missed attempts and we successfully beat the opponent which is designed to be a result of multiple times of attempts involving more than three quarters of the mapped actions, which shall prove the reliability of this detection procedure.

Figure 14 to Figure 16 are some real demos of Sekiro with our implementation.

Figure 14. Use tool. Triggered by Player's right index button and right controller swing.

Figure 15. Defense or Parry. Triggered by Player's left raise or left controller swing

Figure 16. Hook to landscape. Triggered by Player's left index button pressing and left controller swing

5. Conclusion and Future Work

In conclusion, our research part and implementation parts work as desired. The user test can provide effective data in user evaluation. The difference in accuracy of actions satisfies our initial assumption and provides more insights than expected. The performance of the input mapping for Sekiro is also highlighting.

In the future, for the research part, we aim to conducted experiments on more participants to collect more accurate data and make more comprehensive and thorough evaluations. User Statistics also would have higher reliability with the test numbers grow. And for the game implementation parts, we desire to improve the complexity of our Unity game, set different levels of difficulty and design a more intriguing reward system to make the game more attractive to users. The action detection and input mapping pipeline can work to many 3A games run on Windows, which can be used as a platform to explore the VR development of other types of games.

The result of the user test is a good reference for VR environment design. If a game or software pursues high extent of sense of immersion and requires users to actively pose actions to interact with the system, the input should be collected within larger time frame in case of trade-off in efficiency of the interactions. On the other hand, for some core games which attract players with high difficulties and accuracy, involving action detection with the input system may be a good choice of design.

References

[1] R. Cutler and L. S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, 2000.

[2] S.-C. Ho and B.-Y. Li. Improve the design of human-computer interface in digital games. In *2013 Fifth International Conference on Service Science and Innovation*, pages 174–177. IEEE, 2013.

[3] S. B. Kang. Hands-free interface to a virtual reality environment using head tracking, Dec. 28 1999. US Patent 6,009,210.

[4] E. Kruijff and B. E. Riecke. Navigation interfaces for virtual reality and gaming: Theory and practice. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, page C11. ACM, 2018.

[5] F. Madera-Ramirez, E. Segura-Pool, J. Rios-Martinez, and L. Basto-Diaz. A computer graphics program to make exercises using the kinect device. In *Proceedings of the 7th Mexican Conference on Human-Computer Interaction*, page 3. ACM, 2018.