Project Initial Design

Nicholas Muszynski

October 3, 2017

VR Fencing

This project aims to replicate the sport of fencing within virtual reality, to provide players with an immersive experience as they spar against an AI opponent. To heighten the player's immersion in the game, movement of the player's body and control of their sword will be instrumental in playing and succeeding. The goals of this project are to deliver as immersive an experience as possible, with special focus on haptics (and the illusion of haptics) to provide satisfying feedback to the player. Likewise, this will be the main problem of the project, as it is paradoxical in nature: a primary concept of fencing is feeling the pressure of the one's blade against the other, and as such must be creatively approached to achieve this sensation in a virtual environment.

The user experience is intended to be simple: To move, the player physically moves. To attack or defend, the player holds their controller as they would a stick or other long object, and will swing to attack or put it in the way to defend themselves. By eliminating the need for any button or analog presses (except for menu interaction), it will encourage a natural flow of movement that is easy to understand. The user interface will also be minimalistic, only displaying necessary information when needed and keeping the player's view free of distractions and clutter.

Milestones:
1 - Basic compatibility/communication between software and hardware.
2 - Registration of physical movement and controls.
3 - Establish basic sword binding mechanics between physical and artificial players.
5 - Configure 'hitboxes', which determine when a player is hit.
5 - Configure AI for basic movement and strategies.
6 - Basic skeletal mesh animation systems.
7 - Improve AI and mechanics.
8 - Polishing any remaining minor features.

Figure 1: Program structure concept

SOFTWARE

Main Menu
(choose to play/exit)

Constant I/O
between

VR
Hardware

software and
hardware

Main
Loop

- Check if player has been hit
- If they have, add a strike
- If 3 strikes, game over, return to menu
- Check if AI has been hit
- If they have, add a strike
- If 3 strikes, player wins, return to menu
- Update player/controller coordinates
- Update AI's current logic
- Update player view