Barrier Mage: A Gestural Game Mechanic for VR

P. Douglas Reeder
Hominid Software
doug.reeder@hey.com

ABSTRACT

This work implements gesture-based Virtual Reality game play. The user draws mystic symbols with 6-DOF VR controllers. Each symbol has a different effect. This novel game mechanic eliminates an abstraction layer usual in fantasy games, making this game more immersive. Potentially, many more game functions than the number of controller buttons could be at-hand, without immersion-breaking GUI.

KEYWORDS
gesture-based UI, Virtual Reality, immersive gameplay

1. INTRODUCTION

Two decades passed between the invention of the video display terminal by Bell Labs in 1964 and the popularization of the Windows, Icons, Menu, Pointer (WIMP) paradigm by the Macintosh in 1984. Handhelds and tablets passed from hardware keyboards in the 1980s and early 90s, through pen-based interfaces (Palm Graffiti) in the 90s and early 00s, to the swipe software keyboards dominant today. Although VR dates from at least 1968, user interface paradigms have not yet converged.

Gestural interfaces have long been experimented with. Palm webOS made standard gestures (fingers on a capacitive screen) central to the UI of the Palm Pre. iOS and Android support various gestures as shortcuts, for advanced users. Ultraleap (formerly Leap Motion) sells PC peripheral hardware to track hand and finger motion. Hardware and software keyboards are inconvenient in VR, leaving room for a gestural UI.

2. DESIGN AND IMPLEMENTATION

Web technologies were chosen to allow the software to run on a wide variety of headsets, and for familiarity to the developer. APIs for VR on the Web have been in flux - a series of proposals culminating in "WebVR 1.1" was implemented in several browsers, but the growing importance of Augmented Reality (AR) triggered the switch to combined API called WebXR (https://immersive-web.github.io/). Several libraries and frameworks abstract away the API details, and allowed a relatively easy transition from browsers supporting WebVR to WebXR (https://caniuse.com/?search=webxr). The A-Frame framework (https://aframe.io/) was chosen for its extensive community and library of third-party components (https://www.npmjs.com/search?q=aframe-component).

Gestures can encompass both shape and motion. This project used a subset of historical mystic symbols - line shapes in two dimensions, the elements of which can be drawn in any order. Symbols were further restricted to those which could be drawn using only linear segments, arcs and circles.

Six degree of freedom (6-DOF) controllers return position and orientation over time. Standard writing uses a stylus, pen, pencil or brush to transform hand and finger position and orientation into a single 2-D position; drawing and painting in VR use a similar paradigm. As the game rewards symbols drawn large, the user is equipped with a staff, the end of which is 1.09 m from the center of the hand. (Paint and productivity apps would use something much shorter.)

Two modes of drawing were allowed. Pressing the trigger starts a line segment at the current tip location. Holding down the trigger, the user moves the tip to the location of the other end of the segment, and releases the trigger. Pressing and holding a thumb button "paints" a free curve. When the thumb button is released, if the end position is close to the start, the user is presumed to be drawing a circle; otherwise an arc. The painted curve is redrawn as a perfect arc, using the endpoints and midpoint, or a perfect circle, using the start/endpoint and points one- and two-thirds of the way around.

Start and end positions "snap" to existing points, if closer than a threshold.

Symbol templates are rotated to the plane of the drawn symbol and scaled to match, to allow comparison. Drawn segments are compared to template segments using the endpoint positions. Drawn arcs are compared to

template arcs using the positions of the endpoints and midpoint. Drawn circles are compared to template circles using the center, radius, and normal vector.

For each template element (segment, arc or circle), the drawn element with the smallest sum of root-mean-square differences is considered to match. (Thus, a single drawn segment might be the "best" match to two template segments, but the match would be poor.) The elements of a symbol may be drawn in any order. Extra elements may be present, but no match can occur with all of the elements.

For a drawn symbol, the sum of the RMSDs is calculated for each template. The "score" is the inverse of the RMS sum, less a heuristic threshold. The best scoring template, if any are greater than zero, is returned as the matched symbol.

For training, on a successful match, the symbol template is briefly shown - positioned, rotated and scaled to match the drawn symbol - allowing the user to improve future drawing.

Symbol matching is done each time a segment, arc, or circle is completed. This algorithm is O(n). This is done in a single thread on the main processor (a Qualcomm Snapdragon 835 on the least-powerful device on the market) in JavaScript.

Most symbols form barriers, constraining the motion of attacking creatures or damaging them. Others have fantasy effects like illumination or portal travel.

3. DISCUSSION

Users typically learned to draw most of the half-dozen symbols well enough for the code to recognize them within ten minutes, though no rigorous data was collected. They also report the game being fun and engrossing. The algorithm's discrimination between the six symbols was acceptable for game purposes.

Current VR interfaces limit the number of functions to the number of buttons on the controllers, or to the amount of in-world GUI that can be managed. Recognizing a half-dozen symbols does not increase productivity by itself, when a pair of controllers have ten buttons available. This gesture mechanism can support many distinct symbols, and thus functions. The cognitive limit is presumably something like the number of distinct characters in East-Asian character sets. The technical limit is governed by both the precision humans can draw (without force feedback from pressing a drawing implement against a surface) and the algorithm used to distinguish symbols. RMSD is demonstrated to be sufficient to distinguish half a dozen symbols.

4. FUTURE WORK DIRECTIONS

Future work could explore algorithms that use more points to find the best-matching arc or circle.

The symbols are planar, so to rotate the template to match the drawn symbol, the plane best fitting the 3-D points of the drawn symbol must be calculated. The singular value decomposition (https://mathworld.wolfram.com/SingularValueDecomposition.html) would be best for this.

Dots could be drawn using a third controller button, which would allow more historical symbols. Note that many existing symbols still could not be drawn; spirals cannot even be approximated. A simplified version of most symbols could be drawn, akin to the simplified letter forms used in the Graffiti handwriting recognition system on Palm OS.

The number of symbols that the existing algorithm could distinguish should be explored. The task is highly parallelisable, so multiple processor threads could be used. Algorithms from handwriting recognition should also be explored. Productivity apps could also benefit from having many function at hand, without a complex UI.

ACKNOWLEDGMENTS

The Immersive Web Working Group (https://immersive-web.github.io/) and its predecessors have spent six years refining the WebVR and now WebXR APIs, with at least 70 contributors. Mozilla, Google, Samsung, Microsoft and Oculus have implemented them on various platforms.

A-Frame was developed by Diego Marcos, Kevin Ngo and Don McCurdy, with at least 342 contributors.