VRAM/G – gesture enabled VRAM
Technical Report

Hartmut Seichter, hartmut@technotecture.de

Juni 2001

Professur InfAR, Prof. Dr. D. Donath

für den internen Gebrauch an der Bauhaus-Universität Weimar

Vorwort

VRAM wurde 1998 durch Dr. Holger Regenbrecht als Applikation zum Testen und Modellieren von VRML-Welten ins Leben gerufen. Als Eckpunkte für die Entwicklung wurden Portabilität und leichte Bedienbarkeit auch durch Nicht-Informatiker gesetzt. Die Anbindung der Gestenerkennung soll schnelles Skizzieren im virtuellen Raum erlauben.

Technologie

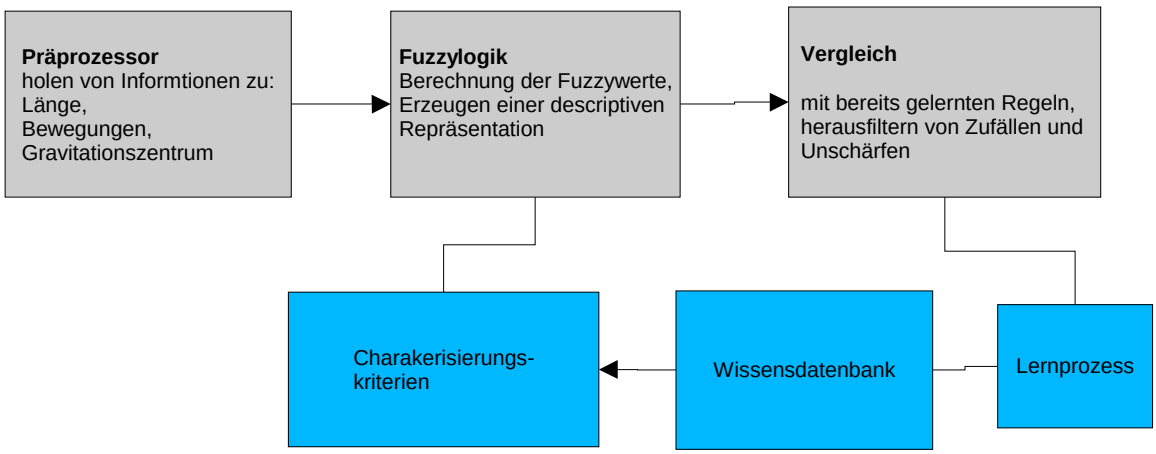
Die vorrangig unterstützten Plattformen sind Microsoft Windows (Win32-API) und SGI IRIX 6.5.x. Dies resultiert aus der verwendeten Scenegraph-API SGI OpenGL Optimizer/SGI Cosmo3D. Dieser unterstützt Windows NT 4.0/2000 und Windows 98 aufwärts (ab VRAM beta 1.3). Wie der Name schon sagt, setzt OpenGL Optimizer auf OpenGL auf und kann somit hardwarebeschleunigte System besonders ansprechen.
VRAM ist multithreaded und optimiert für Multiprozessorsysteme. Als multithreaded Prozesse sind zum Beispiel das Laden und Entladen von Dateien und eben auch die Erkennung der Gesten ausgelegt.

Gestenerkennung

Die Gestenerkennung IMGR (Interactive Multi Gesture Recognition) wurde von Oliver Bimber [1] von der FHG-IGD Rostock entwickelt. IMGR wurde mit der Intention eines 6DOF-Gestenerkennungssystem entwickelt und vereint grundlegende Techniken wie einen Perceptron-Analyser und eine Fuzzy-Logik.
Die Fuzzy-Logik wurde 1965 an der University of California – Berkley von Prof. Lotfi. A Zadeh erfunden. Wesentliche Teile von IMGR stützen sich auf diese Erkenntnisse und wenden sie auf den dreidimensionalen Raum an.

Die Erkennung von Gesten läuft wie folgt ab:

Abb. 1) System der Gestenerkennung aus [2]



Aufgrund der unscharfen Erkennung muss anhand von Wichtungen von Ecken und Geschwindigkeiten die jeweils richtige Geste herausgefiltert werden:

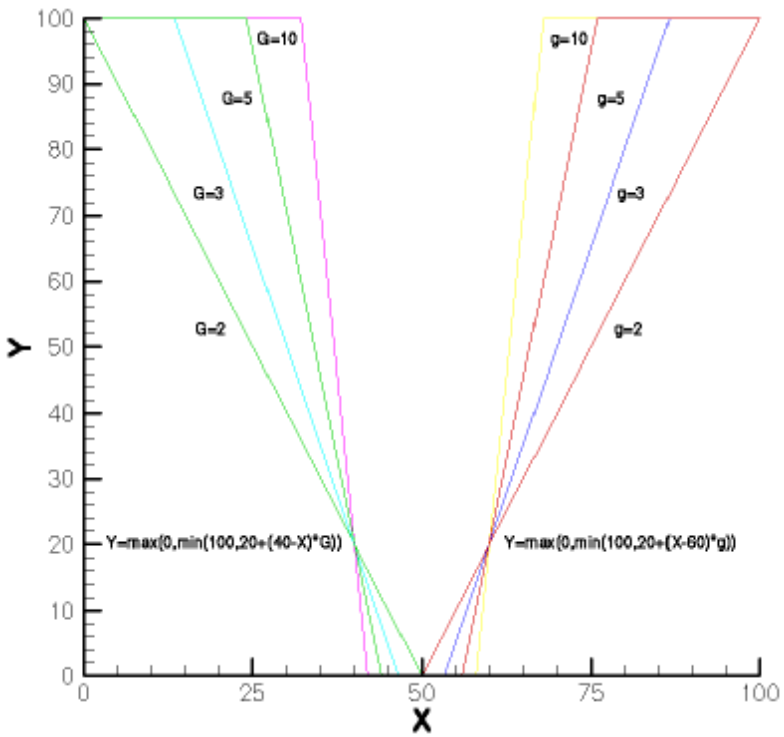


Abb. 2) Bewertungen einer Rotation und ihrer gewichteten Abbildungen
Die erkannten und als richtig bewerteten Gesten werden in einer Datenbank abgelegt. Als

Folgeschritt werden die Gesten dann in einem Abweichungswerte-Pool abgelegt – die Wissensdatenbank des Systems.
Wie bereits erwähnt, kann man innerhalb der IMGR-API zwischen Fuzzylogik-Analyse und Perceptron-Analyse umschalten. Im Fall von VRAM wird ausschließlich die Perceptron-Analyse benutzt, da sie wesentliche Vorteile in Bezug auf Performance hat.

Integration in VRAM

VRAM ist modular aufgebaut. Auch die Gestenerkennung wird extern als Bibliothek mit eingebunden. Das auch nach wie vor eine Version von VRAM ohne Gestenerkennung erzeugt werden kann, wird durch das Compilermakro

```
VRAM_GESTURE
```

gesteuert. Wird dieses auf TRUE gesetzt, hat die kompilierte Version eingebaute Gestenerkennung.

Im Hauptteil von VRAM root.cxx wird ein eigener Thread für die Gestenerkennung angelegt und läuft somit unabhängig vom Programm.
Jede Hand bekommt einen eigenen Thread (hier als Beispiel, die rechte Hand):

```
csThread* GloveRGestureThread;

...
csThread* GloveRGestureThread = new csThread();
GloveRGestureThread->start(GloveRightGestureThread, NULL);
```

Der Gestenthread fängt ab dieser Initialisierung alle Stylus-Klick-Events ab und schickt sie in eine spezielle Hauptprogrammroutine, in der sie dann verarbeitet werden.
Gesten werden momentan innerhalb von VRAM nur als 3DOF (Position) erkannt und bewertet. Die Geste wird anhand ihrer Mittelpunktvektoren bewertet:

```
__gesture_center[0] = g_prop_start_x;
__gesture_center[1] = g_prop_start_y;
__gesture_center[2] = g_prop_start_z;
```

Um nicht zufällige Gesten oder kurze Klicks durch den Recognizer erkennen zu lassen oder die Wissensdatenbank zu „verunsichern“ (siehe auch „Probleme“), werden nur Gesten ab einer bestimmten Länge erkannt:

```
g_prop_number_samples = pl.look_up(PROPERTY_NUMBER_OF_SAMPLES).get();
if(g_prop_number_samples < 35) { ....
```

Danach wird über den dementsprechenden Analyzer die Geste erkannt:

```
gesture_id = rgnzr.recognize(fl);
...
```

und an das Hauptprogramm zurückgemeldet. Dieses verzweigt dann in die Komponente vramManipulation() und ruft dort:

```
case E_GESTURE :  this->handleGestures(pRoot->getGestureID());
...
```

Diese Routine erzeugt dann anhand der GestureID(); die dementsprechende Geometrie am Anfangspunkt der Geste.
Die Gesten-ID's sind:

0) Speichern der momentanen Szene
1) Kegel - cone
2) Würfel - cube
3) Zylinder - cylinder
4) dreiseitiges Prisma - prism
5) Pyramide - pyramide
6) Kugel - sphere
7) Spiralfeder - spring
8) Torus - torus
9) Röhre - tube

Die jeweilige Geometrie muss innerhalb des Arbeitsverzeichnisses unter library/name/name00.wrl abgelegt werden.

Beispiel: Die ID 2 zeigt auf Cube und lädt somit die Datei library/cube/cube00.wrl an die Stelle, an der die Geste begonnen wurde.

Die Dateien, die nachgeladen werden sollen, müssen valide VRML97-Dateien sein und mit der Geometrie auf { 0; 0; 0} zentriert sein. Erweiterungen müssen innerhalb der Klasse vramManipulation eingebunden werden.

Anwendung von Gesten

Bei der Standardinstallation wird eine bereits komplette Gestenbibliothek (die Datei vram_gesture_set.gst) mitgeliefert. Diese kann nach Belieben erneuert oder modifiziert werden. Beispielhaft seien hier die wichtigsten Gesten aufgezeigt:

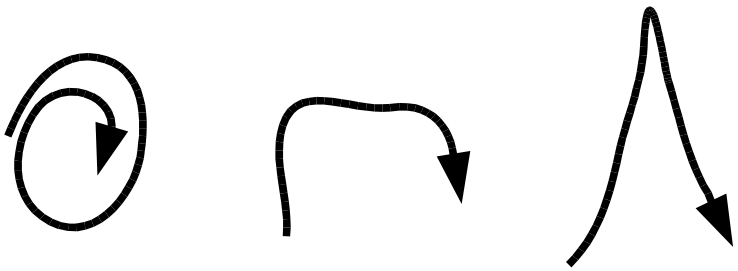


Abb 3) von links nach rechts: Kugel, Würfel, Kegel

Anlernen von Gesten

Wenn die schon gespeicherten Gesten unbrauchbar oder keine Gesten vorhanden (die Datei vram_gesture_set.gst fehlt) sind, so muss VRAM diese initialisieren.

1) VRAM starten
2) nachdem THREAD-READY in der Konsole erscheint [s] (klein) auf der Tastatur drücken
3) nach erfolgtem Speicher erscheint „IMGR: writing current gesture_set to disk.“ auf der Konsole

Achtung! Bereits gespeicherte Daten, bzw. die Datei vram_gesture_set.gst werden überschrieben, respektive gelöscht.
Wenn ein bereits gespeichertes Gestenset geladen werden soll:

1) VRAM starten
2) die Taste [p] auf der Tastatur drücken
3) das Gestenset wurde geladen, wenn „IMGR: reading gesture_set from disk.“ auf der Konsole erscheint

Beim Anlernen von Gesten geht man wie folgt vor:

1) VRAM starten
2) wenn keine Gestendatei vorhanden ist, Taste [s] auf der Tastatur drücken ansonsten weiter bei 4)
3) die Taste [p] auf der Tastatur drücken
4) die Taste [t] auf der Tastatur drücken, es sollte „IMGR: Gesture training ACTIVE.“ auf der Konsole erscheinen
5) nun beginnt man, mit dem Stylus Gesten einzugeben und diese an der Konsole zu

korrigieren
6) mit gedrücktem Stylus-Knopf eine Geste beschreiben
7) nach dem Loslassen des Stylus-Knopfs sollte auf der Konsole „IMGR (999): Recognized Gesture : 999“ und einige zusätzliche Angaben zur gemachten Geste erscheinen
8) es kann nun mittels Tastatur die erkannte Geste korrigiert werden:

0) Save
1) Kegel
2) Würfel
... siehe Implementierung

9) wenn Geste korrigiert werden musste, Geste wiederholen, ansonsten eine neue Geste erzeugen – weiter bei 6)
10)zum Abspeichern [s] auf der Tastatur drücken
11)zum Abschliessen des Lernmodus [t] auf der Tastatur drücken

Probleme

Im Zusammenspiel mit der Gestenerkennung sind einige Parameter zu beachten. Zunächst muss der Knopf Stylus am Polhemus Fastrak sehr zuverlässig arbeiten. Ansonsten kann es zu Fehlinterpretationen der Erkennung kommen. Wenn dies beim Anlernen passiert, erscheinen nicht die Objekte, deren Geste man glaubte gemacht zu haben. Sollte dieser Effekt nach einem erneuten Laden der Gestenbibliothek aus der Datei auftreten, so muss die gesamte Gestenerkennung neu angelernt werden.

Zusammenfassung

In zukünftigen Entwicklungen, wie Jolanda und TAP werden Teile der Programmlogik übernommen werden. Abschliessend muß jedoch festgestellt werden, dass die Gesten wie sie in VRAM/G verwendet werden, nicht zum schnellen Skizzieren geeignet sind, da ein hoher Lernaufwand und eine sehr hohe Präzission beim Gestikulieren erforderlich sind.

Quellen:

[1] Homepage von Oliver Bimber http://www.rostock.igd.fhg.de/~obimber/
[2] Gesture-Based Interaction in Virtual Environments, Oliver Bimber, Februar 1999, FHG-IGD, TU Darmstadt
[3] A Translucent Sketchpad for the Virtual Table Exploring Motion-based Gesture Recognition
L.M.Encarnação(CRCG, Providence/USA),O.Bimber (Fraunhofer-IGD Rostock) ,D.Schmalstieg(Viena Univ. of Technology, Viena/Austria), and

S.D.Chandler(CRCG, Providence/USA) Proceedings of EUROGRAPHICS'99, Milan, Italy, vol. 19, no. 3, 1999
[4] Gesture Controlled Object Interaction : A Virtual Table Case-Study
Oliver Bimber (Fraunhofer-IGD Rostock)
Proceedings of 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media'99, vol. 1, Plzen, Czech Republic, 1999
[5] Continuous 6D Gesture Recognition : A Fuzzy Logic Approach
Oliver Bimber (Fraunhofer-IGD Rostock) Proceedings of 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media'99, vol. 1, Plzen, Czech Republic, 1999
[6] Continuous 3D Gesture Recognition : A Fuzzy Logic Approach Oliver Bimber (Fraunhofer-IGD Rostock) Technical Report, no. 98i013-FEGD, Fraunhofer Institute for Computer Graphics, 1998

Bildverzeichnis: