Performance analysis of a virtual

Gra

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like

ABSTRACT

Performance is a critical concer

1. INTRODUCTION

In Virtual Reality(VR), more than in any other interactive 3-dimensional environment, performance is a critical concern. It can affect the utility of an application, the accuracy of a simulation, and even the comfort of users and audience.

-

the program. This prevents duplication of effort, reduces bugs and development time, and makes it poss h

intermachine) communication

since we t

drops from 60 fps to 30 fps. Moreover, while it is waiting fo

the hardware, and set an additional timestamp once we have used tha

Physical and psychological impact. The effects of poor perfor

var

VR developer [16] – th

o ccai na a ma lies, o simp y o no t h a nd e multitl

to that recorded by `gprof` – total and average times for each function call – but it does have a number of enhancem

discussed at the end of chapter 0, and example tables and screenshots occur throughou

("pthreads"), while the Irix version uses either pthreads or Irix processe

fou

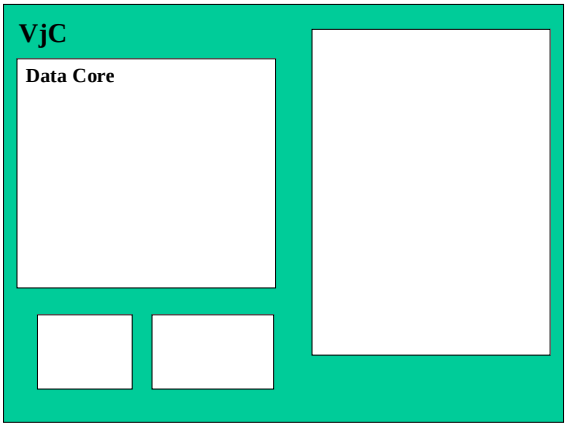
2

Performance Impact: The Mi

As part of its communication

3. VJCONTROL: CONFIGURATION, CONTROL, AND MONITORING PROGRAM

VjControl is, in a sense, VR Juggler's control panel. T


VjC
Data Core

function – fixed Tc 3378Tc 5.27998 0 Td(c)Ta(f)Tj -429124.79998 0 Td(i)TjmTj -0.0768.1 9.j -0.24p0 Tc 3.11997

A VR Juggler configuration file is simply a collection of ConfigChunks, stored as ASCII tex

Chunks into another, and so on, as seen in the figurd1

VjCon

gathe

In a typical V

If we want to do real-time analysis or viewing of performance data, then we need something more significant than just file writer – we need a protocol that an external program can use

5. IMP

compa

a

```
PerfData1 "vjGlPipe 1" 40
0 440528.500000
1 440539.437500
2 440547.781250
3 440834.312500
4 45671
```

refer to

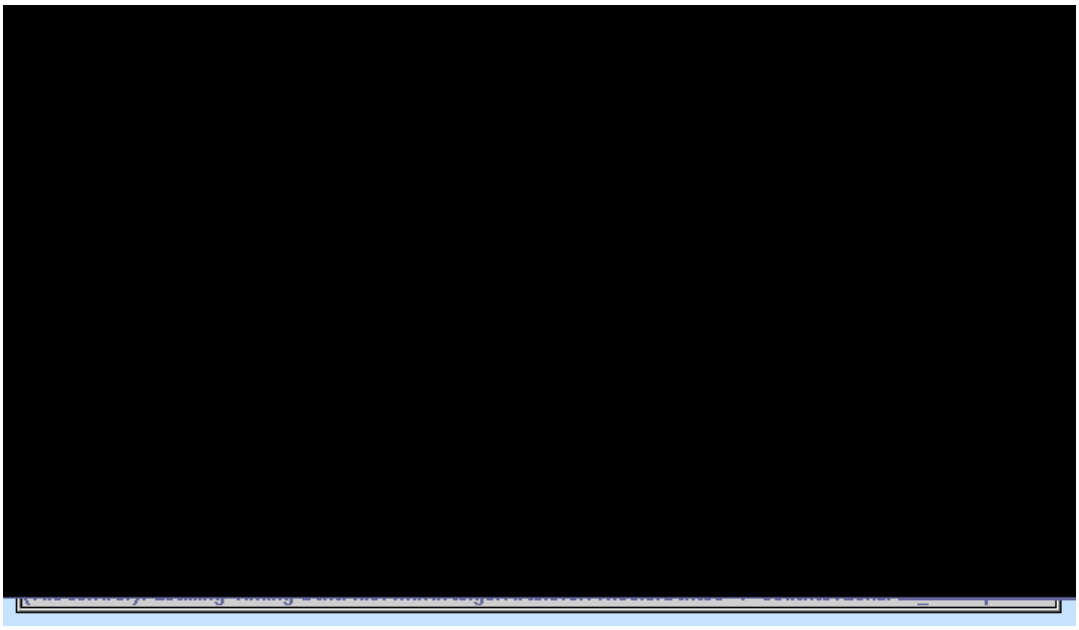


Figure 5.4. VjControl performance data summary for the main loop of a VR Juggler drawing thread.

```
                "updateFrameData"
                }
        end
```

Of course, the overview display in figure 5.4 is not al

once every 20 frames) or something in the system itself (for example, an interrupt being called or another process interfering with our drawing thread).

Figure 5.5. Graph of per-frame timings for the applet

6. TEST HARDWARE, S

are projected onto the walls and floor providing a s

swapping, the test program should run com

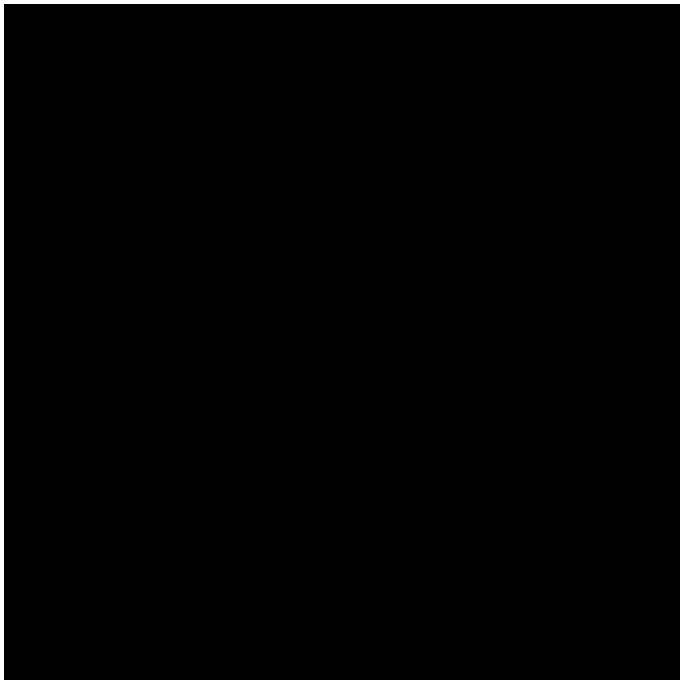


Figure 6.4. The Robot Playground test application.

demonstration during th

7. INITIAL VALIDATION TESTING AND IMPORTANT FINDINGS

The first phase of VR Juggler performance testing was meant to validate the VR Juggler implementation itself – essentially, to show that the overhead and scalability of the software matched

Th

was vi

The next step wa

7.2. Thread class improvements

Table 7

acquiring the `vjThread`'

variation in the inte

8. FINE-GRAINED TESTS

The VR Juggler validation tests resolved several performance issues in the library's

i

Cubes application, navigation based on user input is performed in the `preFrame` function, w

co

`sproc()`-based VR Juggler threads discu

These results ar

Several parts of the display

invoking th

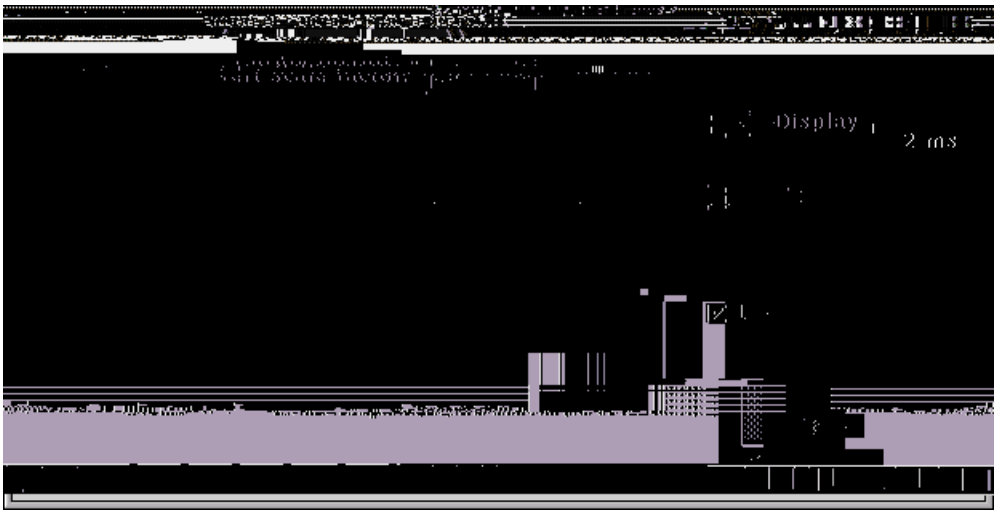


Figure 8.2. Latency variations for keyboard-based tracker simulator.

9. APPLICATION OPTIMIZATION TESTS

The performance measure

after seven interva

a

are analogous to those for instrumenting the drawing methods.

Just as in the case of instrumenting drawing, the decision about exactly where to place tim

We might also experiment with turning off features such as t

10. CO

adequately det

st

10.3

BIBLIOGRAPHY

[1] Allen BierbI um, PI trick HI rtling, Christopher Just, Kevin Meinert, i Irolia Cruz-