Floating Point and CPU vs GPU Rendering

Sam Moore, David Gow

July 24, 2014

Abstract

Modern GPUs appear to not be compliant with IEEE-754 when using floating point operations. There is also difference between the behaviour of different GPU models. We compare the rendering of filled circles on a x86-64 CPU and a GPU (AMD/ATI Whilstler LE [Radeon HD 6610M/7610M]).

1 Introduction

Although it is well known that the behaviour of GPU drivers is inconsistent, there is little research into the behaviour of floating point operations using such drivers.

In 2004 Hillesland and Lastra adapted Kahan's well known program for testing floating point arithmetic on CPUs during the 1980s "Paranoia" for GPUs and found that many GPUs did not appear to be compliant with IEEE-754[?].

Given the recent interest in use of the GPU for vector graphics[?] this seems worthy of further investigation. Using the same algorithm implemented in C/C++ and GLSL we have shown that a particular GPU using a particular driver exhibit less precision than IEEE-754 binary32 floats.

2 Algorithm

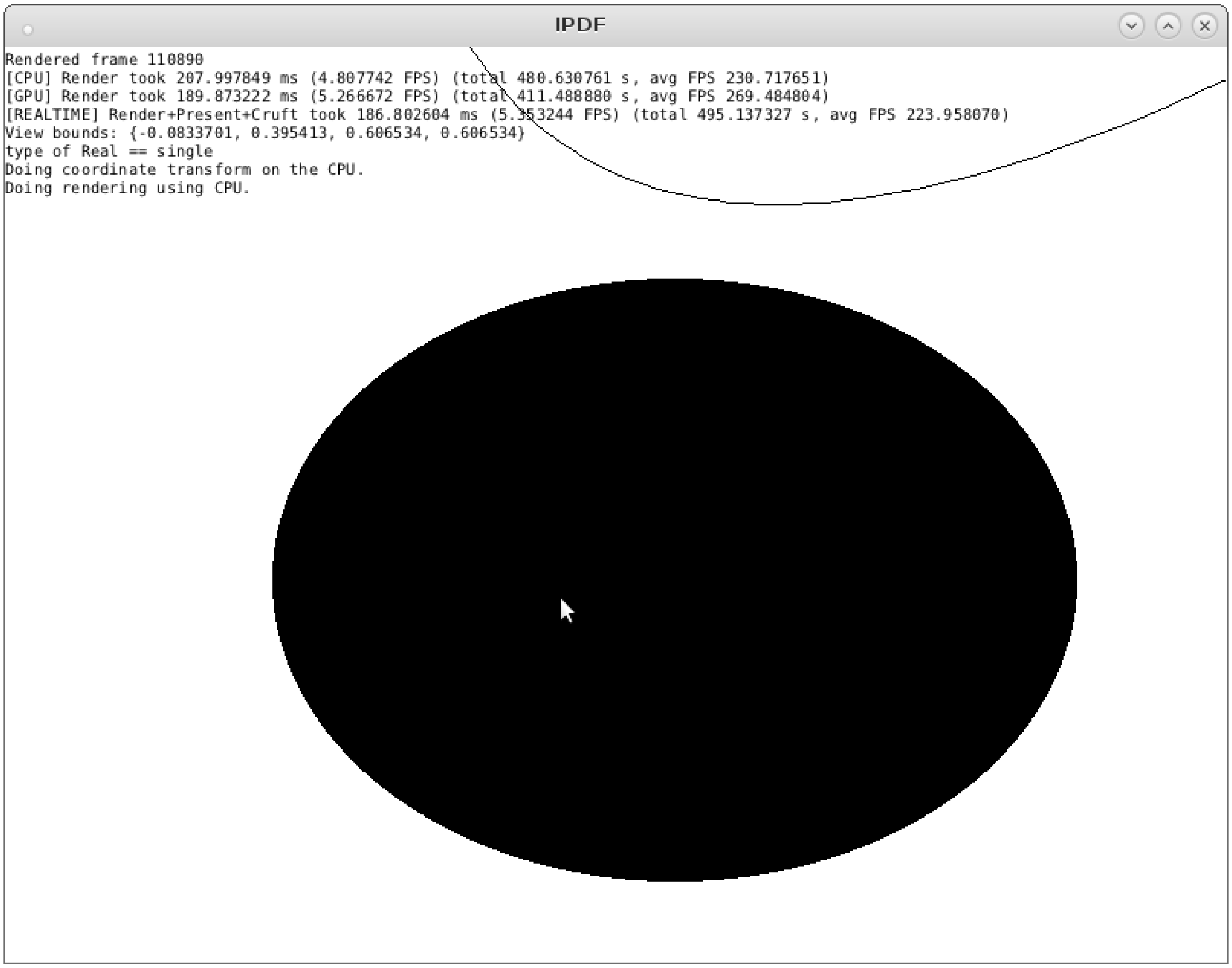
For each integer valued (x, y) in the bounding rectangle, if $x^2 + y^2 \leq r^2$ where r is the radius of the circle, then (x, y) should be filled.

There are two sources of error; the coordinate transforms of vertices *before* rendering, and the operations required to subsequently render the circle at that position. The difference between using the CPU and GPU for the former is apparent but is only easily demonstrated in a live demo. The images below demonstrate the precision issues with the latter[1].

3 Results

Each pair of figures compares rendering using an x86-64 CPU and OpenGL shaders running on the AMD/ATI Whilstler LE [Radeon HD 6610M/7610M] using the *fglrx* proprietry graphics drivers.

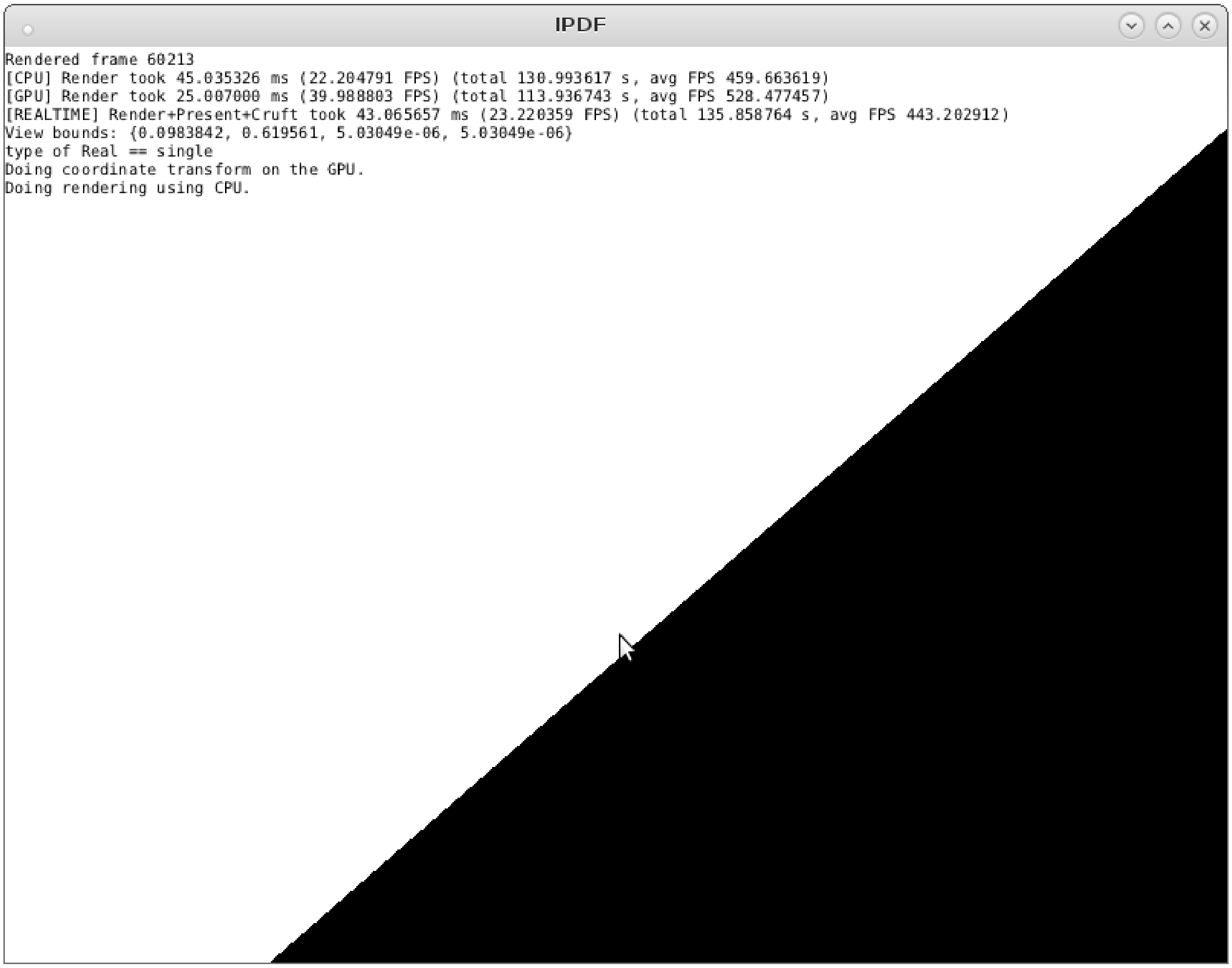
[1] Me write um good english



```
Rendered frame 110890
[CPU] Render took 207.997849 ms (4.807742 FPS) (total 480.630761 s, avg FPS 230.717651)
[GPU] Render took 189.873222 ms (5.266672 FPS) (total 411.488880 s, avg FPS 269.484804)
[REALTIME] Render+Present+Cruft took 186.802604 ms (5.353244 FPS) (total 495.137327 s, avg FPS 223.958070)
View bounds: {-0.0833701, 0.395413, 0.606534, 0.606534}
type of Real == single
Doing coordinate transform on the CPU.
Doing rendering using CPU.
```

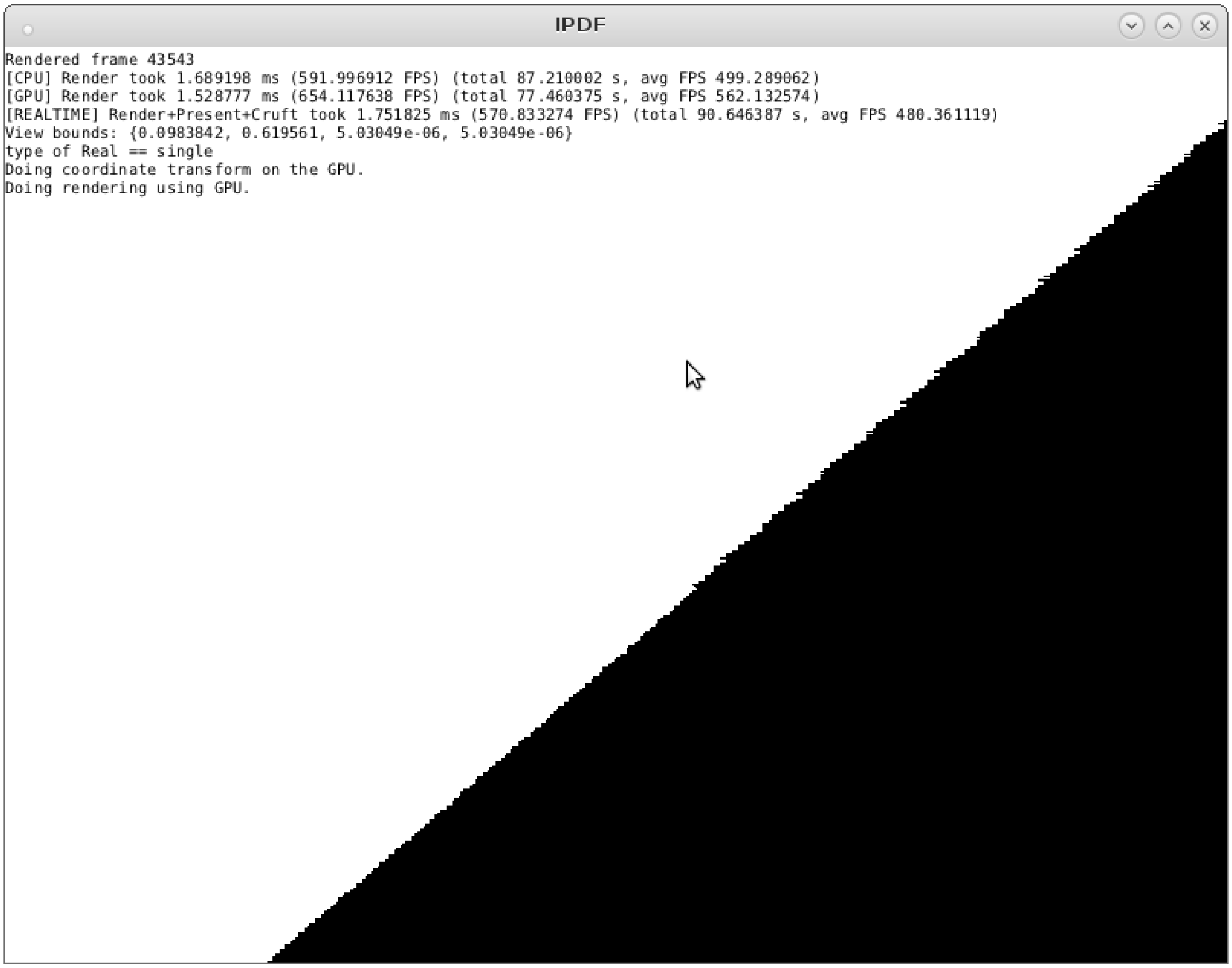
Figure 1: A circle. Sort of.

The images produced by GPU and CPU rendering are indistinguishable at the original scale. Note that the ”CPU rendering” is essentially producing a bitmap and then sending that to the GPU; so all floating point operations are still done on the CPU, wherase ”GPU rendering” involves passing floats representing vertex positions to a GLSL shader program.



```
Rendered frame 60213
[CPU] Render took 45.035326 ms (22.204791 FPS) (total 130.993617 s, avg FPS 459.663619)
[GPU] Render took 25.007000 ms (39.988803 FPS) (total 113.936743 s, avg FPS 528.477457)
[REALTIME] Render+Present+Cruft took 43.065657 ms (23.220359 FPS) (total 135.858764 s, avg FPS 443.202912)
View bounds: {0.0983842, 0.619561, 5.03049e-06, 5.03049e-06}
type of Real == single
Doing coordinate transform on the GPU.
Doing rendering using CPU.
```

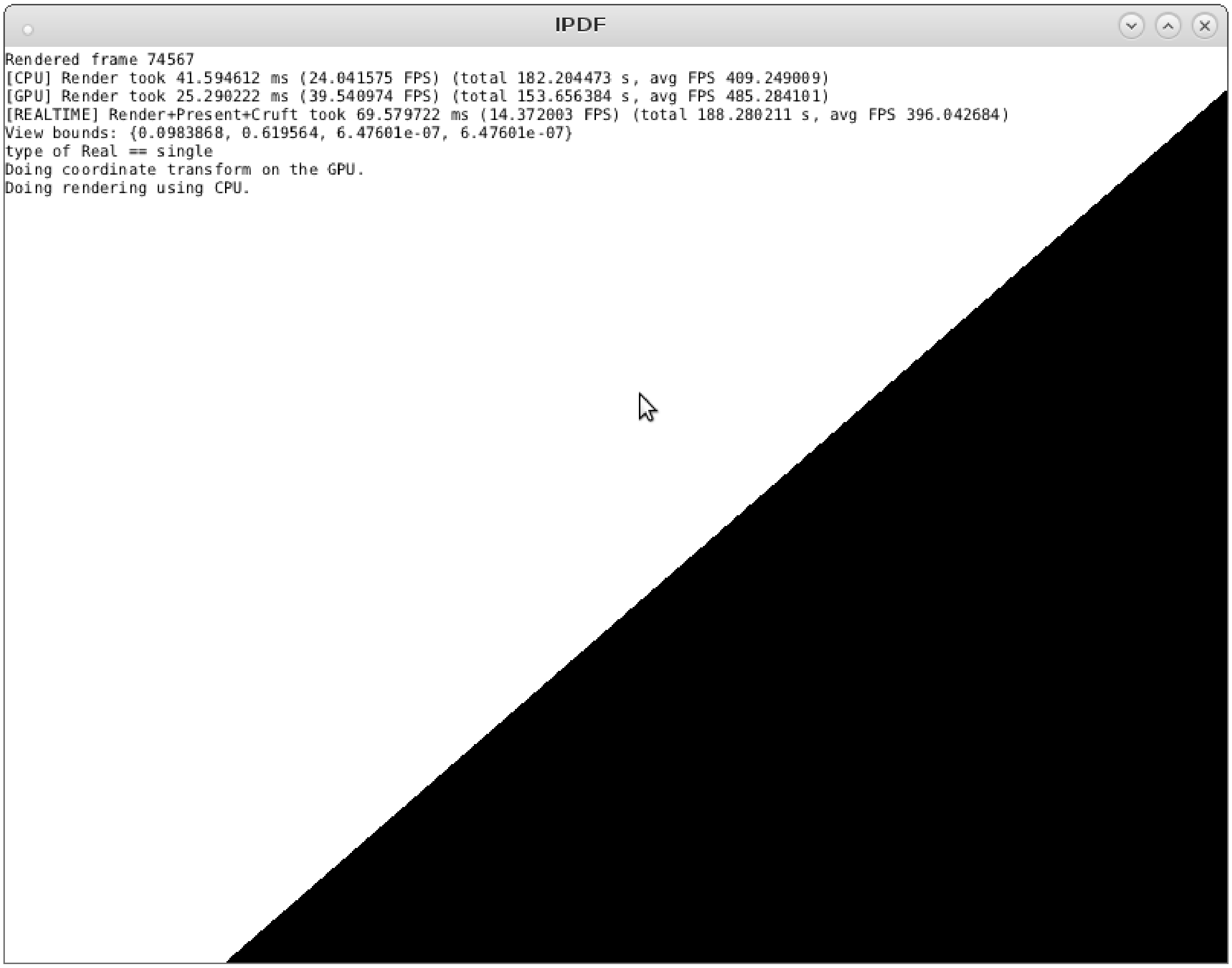
Figure 2: CPU, zoomed



```
Rendered frame 43543
[CPU] Render took 1.689198 ms (591.996912 FPS) (total 87.210002 s, avg FPS 499.289062)
[GPU] Render took 1.528777 ms (654.117638 FPS) (total 77.460375 s, avg FPS 562.132574)
[REALTIME] Render+Present+Cruft took 1.751825 ms (570.833274 FPS) (total 90.646387 s, avg FPS 480.361119)
View bounds: {0.0983842, 0.619561, 5.03049e-06, 5.03049e-06}
type of Real == single
Doing coordinate transform on the GPU.
Doing rendering using GPU.
```

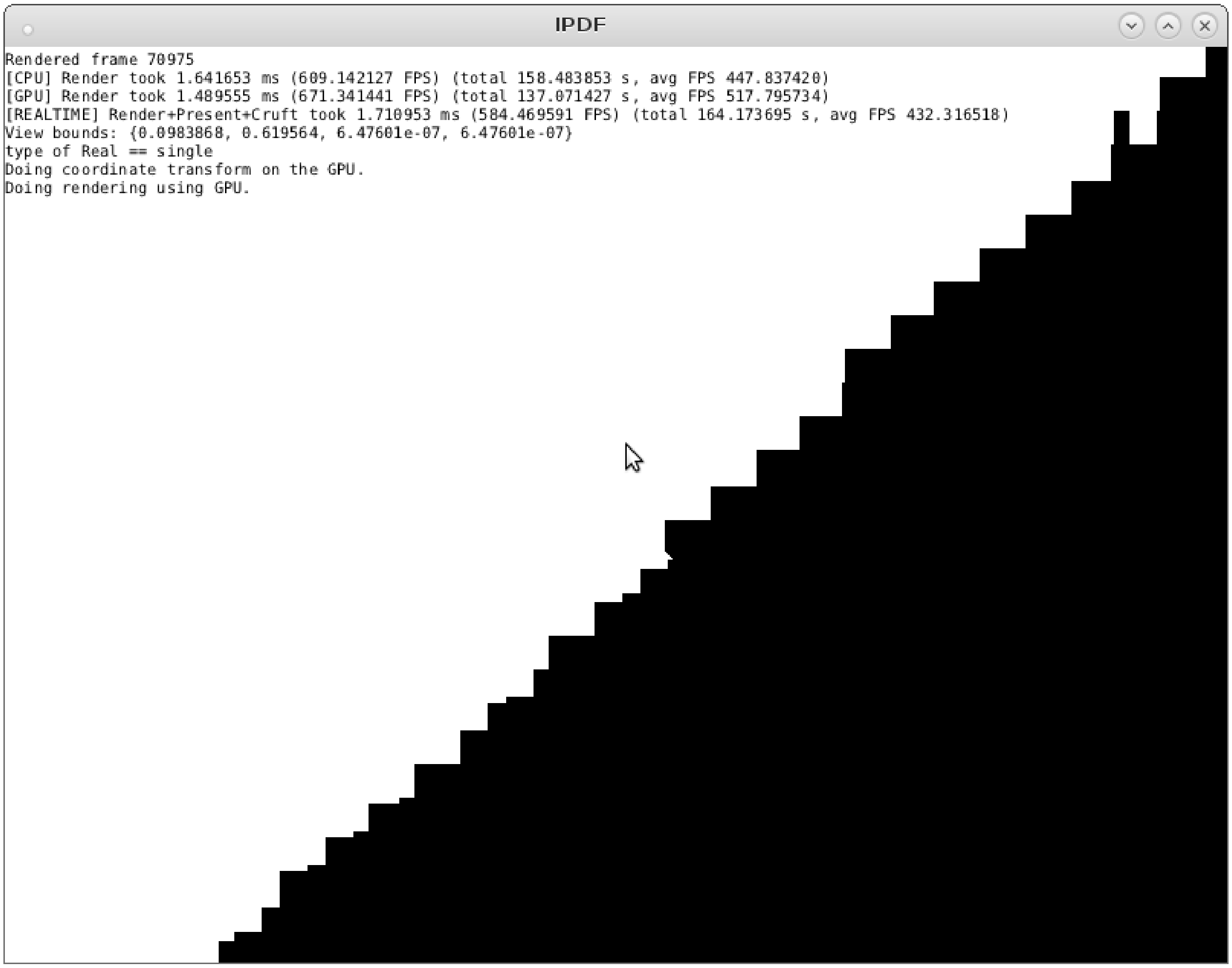
Figure 3: GPU, zoomed

The rounding errors begin to become apparent.



```
Rendered frame 74567
[CPU] Render took 41.594612 ms (24.041575 FPS) (total 182.204473 s, avg FPS 409.249009)
[GPU] Render took 25.290222 ms (39.540974 FPS) (total 153.656384 s, avg FPS 485.284101)
[REALTIME] Render+Present+Cruft took 69.579722 ms (14.372003 FPS) (total 188.280211 s, avg FPS 396.042684)
View bounds: {0.0983868, 0.619564, 6.47601e-07, 6.47601e-07}
type of Real == single
Doing coordinate transform on the GPU.
Doing rendering using CPU.
```

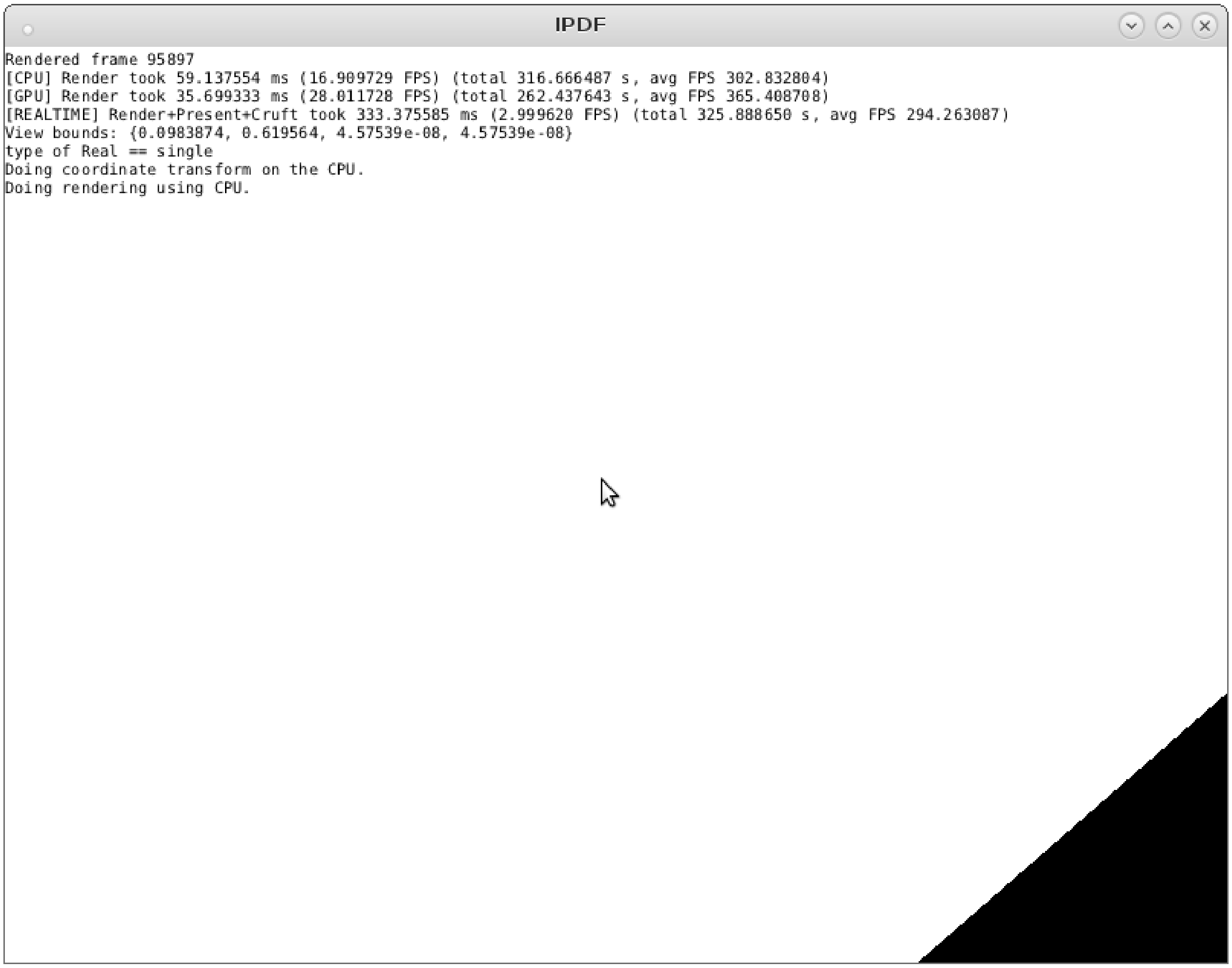
Figure 4: CPU, zoomed more



```
Rendered frame 70975
[CPU] Render took 1.641653 ms (609.142127 FPS) (total 158.483853 s, avg FPS 447.837420)
[GPU] Render took 1.489555 ms (671.341441 FPS) (total 137.071427 s, avg FPS 517.795734)
[REALTIME] Render+Present+Cruft took 1.710953 ms (584.469591 FPS) (total 164.173695 s, avg FPS 432.316518)
View bounds: {0.0983868, 0.619564, 6.47601e-07, 6.47601e-07}
type of Real == single
Doing coordinate transform on the GPU.
Doing rendering using GPU.
```

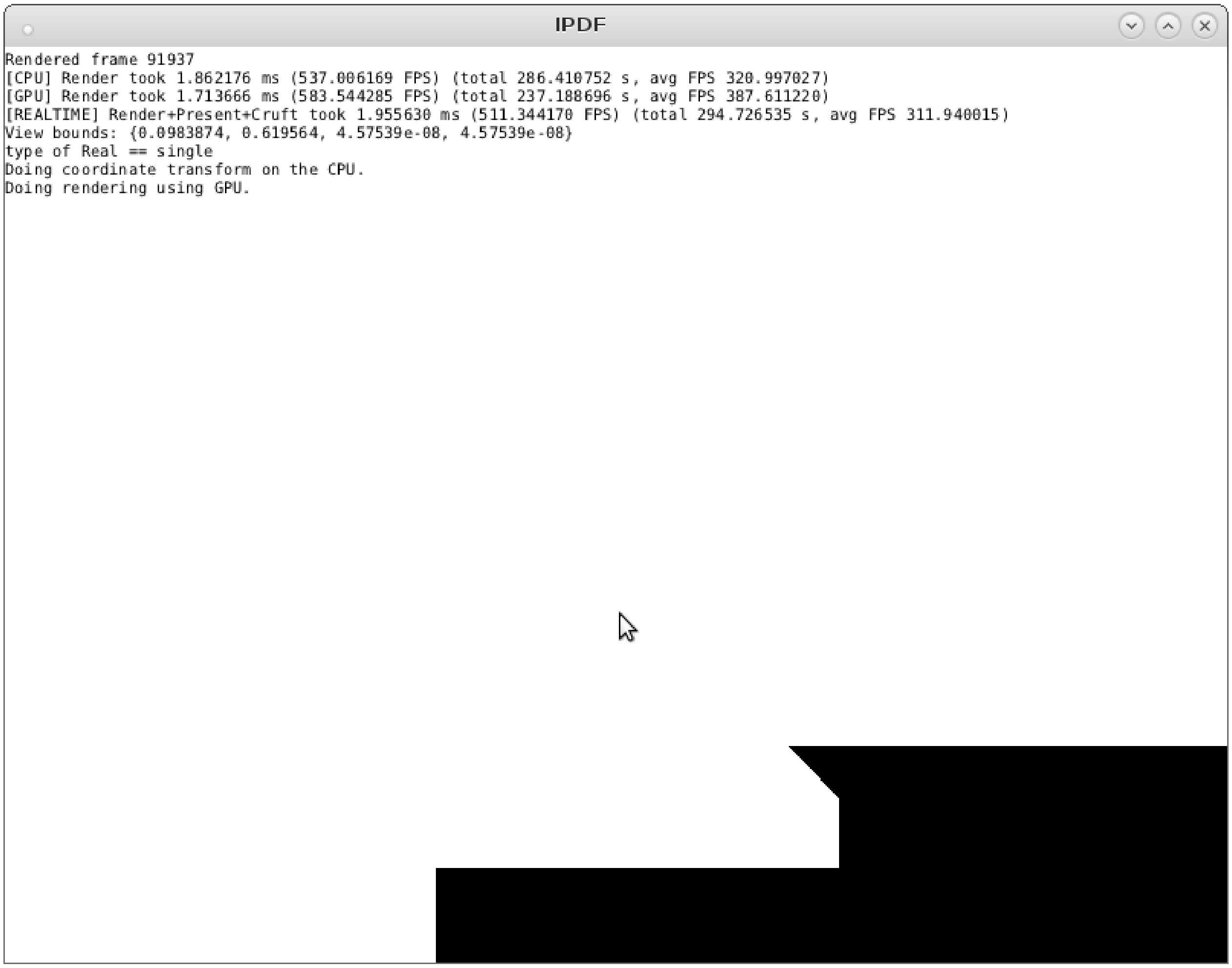
Figure 5: GPU, zoomed more

Even worse...



```
Rendered frame 95897
[CPU] Render took 59.137554 ms (16.909729 FPS) (total 316.666487 s, avg FPS 302.832804)
[GPU] Render took 35.699333 ms (28.011728 FPS) (total 262.437643 s, avg FPS 365.408708)
[REALTIME] Render+Present+Cruft took 333.375585 ms (2.999620 FPS) (total 325.888650 s, avg FPS 294.263087)
View bounds: {0.0983874, 0.619564, 4.57539e-08, 4.57539e-08}
type of Real == single
Doing coordinate transform on the CPU.
Doing rendering using CPU.
```

Figure 6: CPU, zoomed more



```
Rendered frame 91937
[CPU] Render took 1.862176 ms (537.006169 FPS) (total 286.410752 s, avg FPS 320.997027)
[GPU] Render took 1.713666 ms (583.544285 FPS) (total 237.188696 s, avg FPS 387.611220)
[REALTIME] Render+Present+Cruft took 1.955630 ms (511.344170 FPS) (total 294.726535 s, avg FPS 311.940015)
View bounds: {0.0983874, 0.619564, 4.57539e-08, 4.57539e-08}
type of Real == single
Doing coordinate transform on the CPU.
Doing rendering using GPU.
```

Figure 7: GPU, zoomed more

Image is not recognisable as once being a circle. Note that at this scale there are also issues with translation using the CPU (ie: It was not possible to position the circle so that it covered the same fraction of the screen as in the earlier images).

It is hard to tell how much of this is due to bugs in the fglrx driver and how much is actually due to physical limitations on the GPU hardware.

David found code for a graphics driver with a `USE_IEEE_FLOATS` define that was `false` by default...[2]

4 Conclusions

fglrx is pretty terrible. GPUs are probably not as good at floating point as CPUs.

[2] David say more here?