SET10108 Concurrent and Parallel Systems Coursework 1

***Index Terms*—I**

I. Introduction

II. Inital Analysis

To perform the initial analysis of the program the program was run serval times changing several variables. These test showed that the algorithm performed well with small sample sizes however similarly to most serial programs the system struggled with large sample sizes showing a bottleneck exist as the time taken to complete the complete the picture increased exponentially the larger the sample per pixel this was also the case for changing the size of the image where a larger image would take a greater time. However while testing adding more objects to the scene it was noted that this had little effect on the overall time taken to complete which made sense as this did not add more pixels to the image but changed the picture itself.

A. *Parallelizable Places*

```
for (size_t y = 0; y < dimension; ++y)
{
  cout << "Rendering " << dimension << " * " << dimension <<↩
      "pixels. Samples:" << samples * 4 << " spp (" << 100.0 * y ↩
     / (dimension − 1) << ")" << endl;
  for (size_t x = 0; x < dimension; ++x)
  {
    for (size_t sy = 0, i = (dimension − y − 1) * dimension + x; sy <↩
      2; ++sy)
    {
      for (size_t sx = 0; sx < 2; ++sx)
      {
        r = vec();
        for (size_t s = 0; s < samples; ++s)
        {
          double r1 = 2 * get_random_number(), dx = r1 < 1 ? sqrt↩
    (r1) − 1 : 1 − sqrt(2 − r1);
          double r2 = 2 * get_random_number(), dy = r2 < 1 ? sqrt↩
    (r2) − 1 : 1 − sqrt(2 − r2);
          vec direction = cx * static_cast<double>(((sx + 0.5 + dx)↩
     / 2 + x) / dimension − 0.5) + cy * static_cast<double>(((sy + ↩
    0.5 + dy) / 2 + y) / dimension − 0.5) + camera.direction;
          r = r + radiance(spheres, ray(camera.origin + direction * ↩
    140, direction.normal()), 0) * (1.0 / samples);
        }
        pixels[i] = pixels[i] + vec(clamp(r.x, 0.0, 1.0), clamp(r.y, ↩
    0.0, 1.0), clamp(r.z, 0.0, 1.0)) * 0.25;
      }
    }
  }
}
```

Listing 1. The code that would be parallised

III. Methodology

1) *OpenMP*

2) *Multi-Threading*

3) *Algorithimic Skeletons*

4) *CPU-Paralleism*

IV. Results

A. *Discussion*

V. Conclusion

References

[1] M. X, “Something great,” 2005.