DATA STRUCTURES AND ALGORITHMS

KEIRAN MILLAR

PURPOSE

The purpose of this program is to compute the Mandelbrot set and display it.

To increase the speed of the computation, I have parallelised it using the GPU.

STRUCTURE

- My program is structured with an infinite while loop that set a variable called button to whatever key the user presses, these button presses then manipulate the numbers that are given into the compute Mandelbrot function

```
if (button != NULL)
{
    compute_mandlebrot((num1 * zoom) + offsetX, (num2 * zoom) + offsetX, (num3 * zoom) + offsetY, (num4 * zoom) + offsetY);
}
```

THREADS

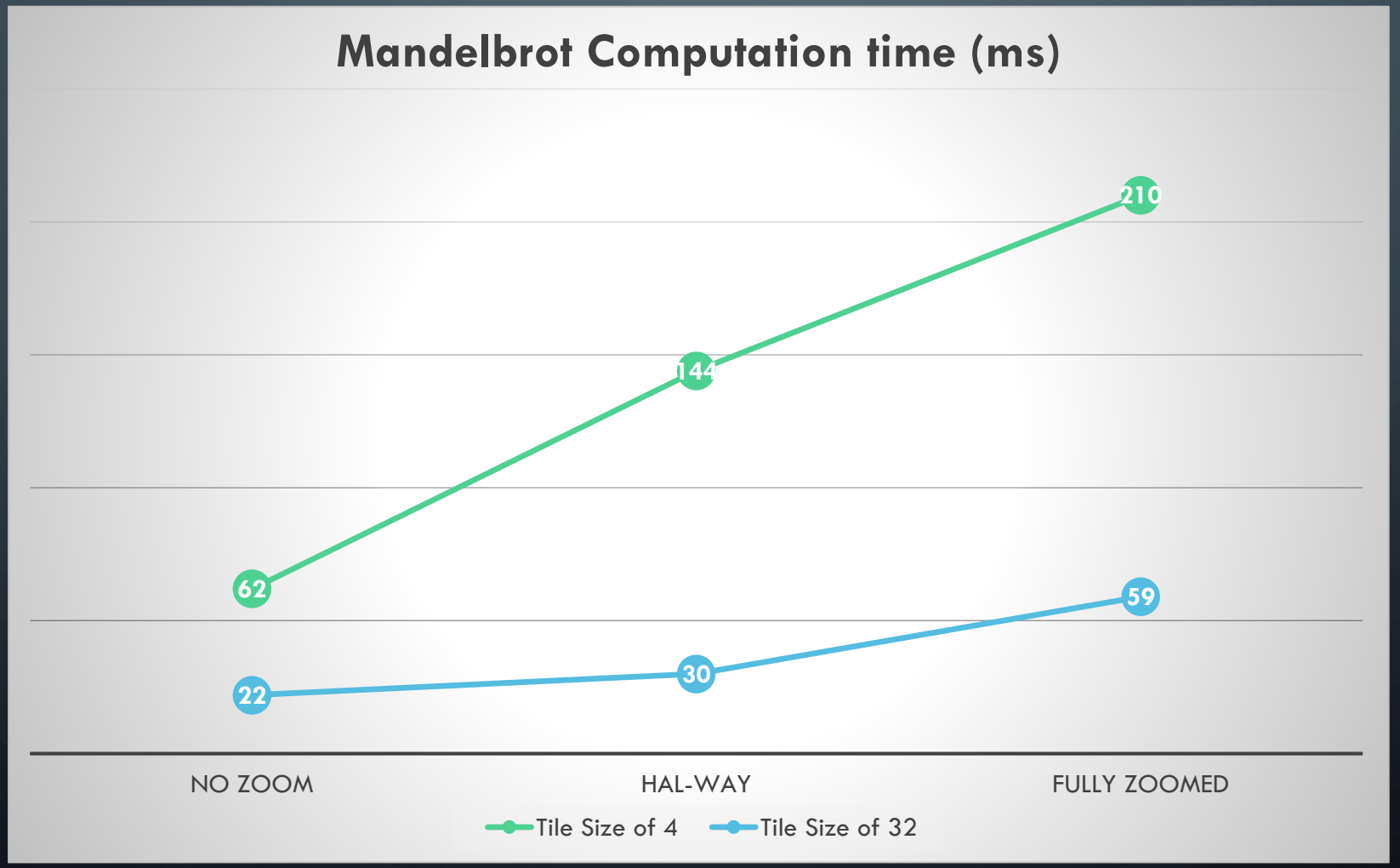
- My application uses a tile size of 32 to split up the work required to calculate the Mandelbrot set.
- There was no need for interaction between threads due to the fact that there are no shared resources between the threads.

PERFORMANCE EVALUATION

- GPU-
  - NVIDIA GeForce GT 440

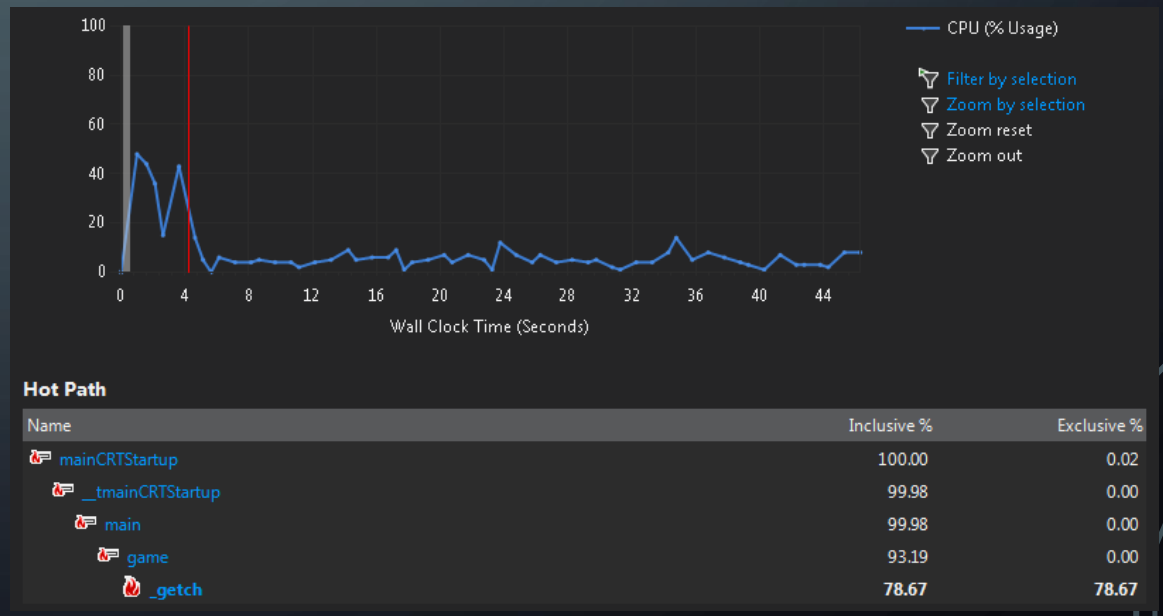
At the start I used a series of for loops and if statements to display my Mandelbrot on the screen but it was extremely slow due to having to set each pixel to a different colour, every time the frame was displayed. Instead I decided to make a bitmap image of my Mandelbrot set and display the bitmap, this drastically increased the speed at which my Mandelbrot set was displayed

PERFORMANCE EVALUATION



PERFORMANCE EVALUATION

- Using the profiler built into Visual Studio 2013 I was able to determine that the _getch function is where my program spent most of its time, knowing this if I was to further improve my program I would try and find a more efficient way to take in user input in real-time



CRITICAL EVALUATION

One problem I came across when using my program is that when you zoom in enough the Mandelbrot set starts to look very blocky, I assume that this is due to the fact the parallelisation code is using a uint32_t variable. If I was to improve my program I would find another variable type which can hold more information than a uint32_t. I tried using a uint64_t but it was not supported by AMP.