RUNTIME (DOUBLES)

| **FRAMES** | **RESOLUTION** | **RUNTIME** |
| --- | --- | --- |
| 30 | 640 x 360 | 0.003074 |
| 30 | 1280 x 720 | 0.011665 |
| 30 | 1920 x 1080 | 0.025645 |
| 30 | 2560 x 1440 | 0.044989 |
| 30 | 3840 x 2160 | 0.099644 |
| 60 | 640 x 360 | 0.006084 |
| 60 | 1280 x 720 | 0.023227 |
| 60 | 1920 x 1080 | 0.051256 |
| 60 | 2560 x 1440 | 0.089905 |
| 60 | 3840 x 2160 | 0.193351 |

RUNTIME (FLOATS)

| **FRAMES** | **RESOLUTION** | **RUNTIME** |
| --- | --- | --- |
| 30 | 640 x 360 | 0.001438 |
| 30 | 1280 x 720 | 0.005347 |
| 30 | 1920 x 1080 | 0.011652 |
| 30 | 2560 x 1440 | 0.020414 |
| 30 | 3840 x 2160 | 0.045368 |
| 60 | 640 x 360 | 0.002790 |
| 60 | 1280 x 720 | 0.010562 |
| 60 | 1920 x 1080 | 0.023205 |
| 60 | 2560 x 1440 | 0.040944 |
| 60 | 3840 x 2160 | 0.090651 |

Bonus:  
As seen from the tables, for the floats it seems to have roughly halved the runtime for each problem size then compared to doubles. This may be since floats are single precision and doubles are double precision. Thus there is a halved runtime for floats when compared to doubles. Also for GPUs, the majority of the ALU is devoted to floating point computation, meaning that the floating point code is easier to compute compared to the double code.