

# Speeds of Sphere Engines.

---

## Section 1: The code<sup>1</sup>

For: increment in Check:

```
for (var i = 0; i++ < 10000000;) { }
```

For: increment in Step:

```
for (var i = 0; i < 10000000; i++) { }
```

For: increment in Body:

```
for (var i = 0; i < 10000000;) { i++; }
```

For: addition by 1 in Step:

```
for (var i = 0; i < 10000000; i += 1) { }
```

For: addition by 1 in Body:

```
for (var i = 0; i < 10000000;) { i += 1; }
```

For: addition by 2 in step:

```
for (var i = 0; i < 10000000; i += 2) { }
```

For: addition by 1 in Step and Body:

```
for (var i = 0; i < 10000000; i++) { i++; }
```

While: increment in Check:

```
var i = 0;  
while (i++ < 10000000) { }
```

While: increment in Body:

```
var i = 0;  
while (i < 10000000) { i++; }
```

---

<sup>1</sup> Pure ECMA Script v5.

## Section 2: The Tests<sup>2</sup>

**Table 1: Sphere v1.5**

Name of Test:	Total time (milliseconds)
For: increment in Check	891.6
For: increment in Step	1062.8
For: increment in Body	1062.3
For: addition by 1 in Step	1319.7
For: addition by 1 in Body	1319.4
For: addition by 2 in Step	662.9
For: addition by 1 in Step and Body	715.0
While: increment in Step	897.7
While: increment in Body	1061.8

**Table 2: Sphere v1.6**

Name of Test:	Total time (milliseconds)
For: increment in Check	397.5
For: increment in Step	561.6
For: increment in Body	559.7
For: addition by 1 in Step	958.4
For: addition by 1 in Body	963.2
For: addition by 2 in Step	487.5
For: addition by 1 in Step and Body	462.8
While: increment in Check	396.2
While: increment in Body	564.0

**Table 3: Sphere SFML**

Name of Test:	Total time (milliseconds)
For: increment in Check	28.0
For: increment in Step	19.9
For: increment in Body	30.4
For: addition by 1 in Step	336.7
For: addition by 1 in Body	335.5
For: addition by 2 in Step	171.3
For: addition by 1 in Step and Body	28.9
While: increment in Check	27.6
While: increment in Body	30.6

---

<sup>2</sup> Tested on an Intel i5 3.30Ghz processor with 8GB of 1600 speed RAM running Sphere natively on Windows 7 sp1.

### Section 3: Conclusion

It seems out of the results in the tables, while loops tend to be the fastest. But there is an interesting innovation I made early on: a for loop with an increment in its check is rival to the speed of a while loop with an increment in its check.

In SSFML, which is my Jurassic and SFML based Sphere Engine, which compiles JS to CIL, runs markedly faster, up to 28x vs. Sphere 1.6 and 56x vs. Sphere 1.5. And Sphere 1.6 is about two times faster than Sphere 1.5. I notice that the addition operator is markedly slower in all engines vs. its increment counterpart. I was surprised to see in the SSFML, that the addition operator is about 17 times slower. In fact, both Sphere 1.6 and SSFML it's cheaper to increment twice than to use the addition operator. (Particularly in SSFML, it seems that repeating the increment operator on loops even up to += 12 would still hold a slight advantage).