

The humble FIFO

Single Consumer, Single Producer

```
template <typename T> class fifo {
public:
    bool push (T && arg) {
        auto pos = writepos.load();
        auto next = (pos + 1) % slots.size();

        if (next == readpos.load())
            return false;

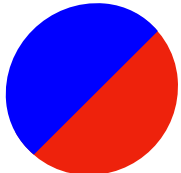
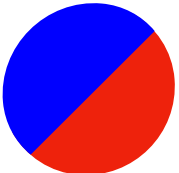
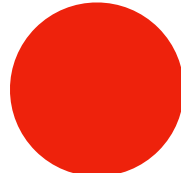
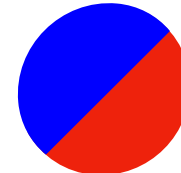
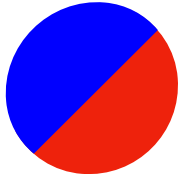
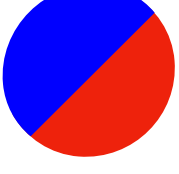
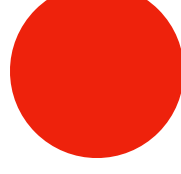
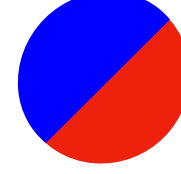
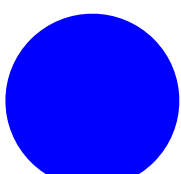
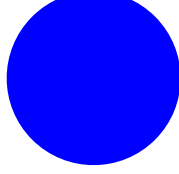
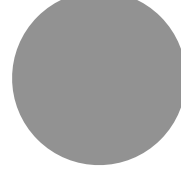
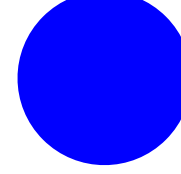
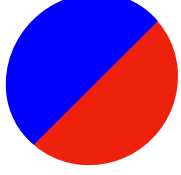
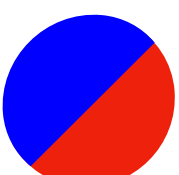
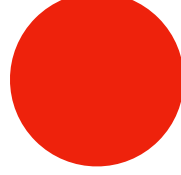
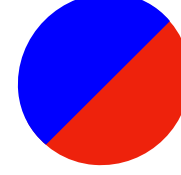
        slots[pos] = std::move (arg);
        writepos.store (next);
        return true;
    }

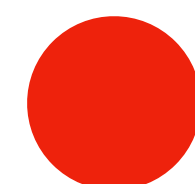
    bool pop(T& result) {
        auto pos = readpos.load();

        if (pos == writepos.load())
            return false;

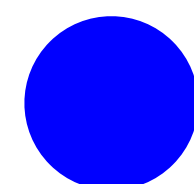
        result = std::move (slots[pos]);
        readpos.store ((pos + 1) % slots.size());
        return true;
    }
private:
    std::vector<T> slots = {};
    std::atomic<int> readpos = {0}, writepos = {0};
};
```

Costs of various FIFOs

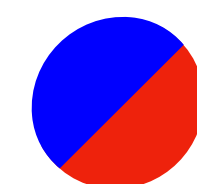
<div> <div>Producer</div> <div>Consumer</div> </div>		Single Producer		Multiple Producer	
		Report Full	Overwrite on Full	Report Full	Overwrite on Full
Single Consumer	Report Empty				
	“null” on Empty				
Multiple Consumer	Report Empty				
	“null” on Empty				



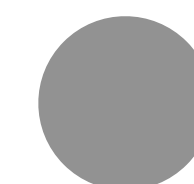
Wait free on read



Wait free on write



Wait free on
read and write



Not wait free on
write or read