



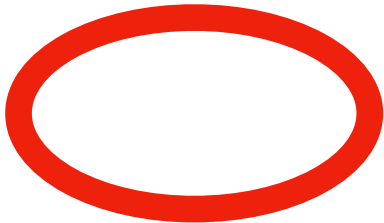

```
class WavetableSynthesizer
{
public:
    void audioCallback()
    {
        if (std::unique_lock<mutex> tryLock (mutex, std::try_to_lock); tryLock.owns_lock())
        {
            // Do something with wavetable
        }
        else
        {
            // Do something else as wavetable is not available
        }
    }

    void updateWavetable (/* args */)
    {
        // Create new Wavetable
        auto newWavetable = std::make_unique<Wavetable> (/* args */);

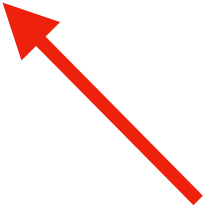
        {
            std::lock_guard<std::mutex> lock (mutex);
            std::swap (wavetable, newWavetable);
        }

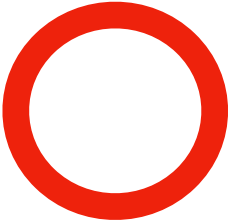
        // Delete old wavetable here to lock for least time possible
    }

private:
    mutex mutex;
    std::unique_ptr<Wavetable> wavetable;
};
```



What is nmutex?





What happens here?







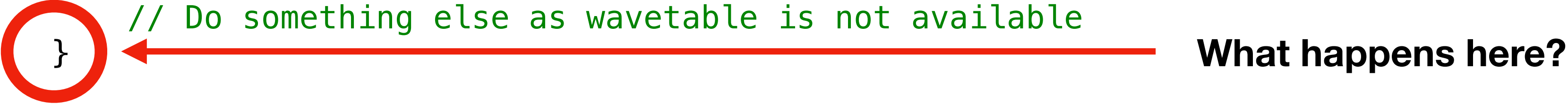
35

5

```

class WavetableSynthesizer
{
public:
    void audioCallback()
    {
        if (std::unique_lock<mutex> tryLock (mutex, std::try_to_lock); tryLock.owns_lock())
        {
            // Do something with wavetable
        }
        else
        {
            // Do something else as wavetable is not available
        }
    }
}

```



What happens here?

```

void updateWavetable (/* args */)
{
    // Create new Wavetable
    auto newWavetable = std::make_unique<Wavetable> (/* args */);

    {
        std::lock_guard<std::mutex> lock (mutex);
        std::swap (wavetable, newWavetable);
    }

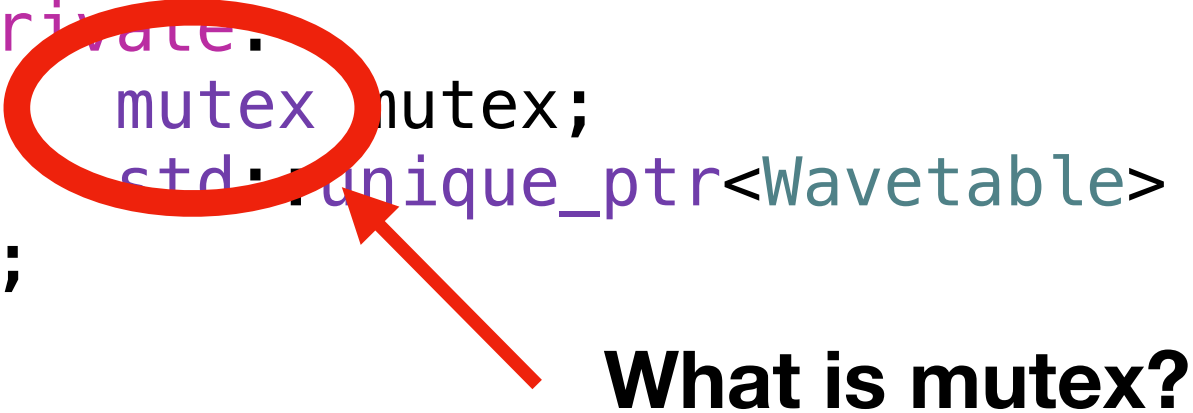
    // Delete old wavetable here to lock for least time possible
}

```

```

private:
    mutex mutex;
    std::unique_ptr<Wavetable> wavetable;
};

```



What is mutex?

`std::mutex<>`

- `std::mutex::try_lock()` is wait-free