CRTP как способ избежать дублирования кода

Вы можете задуматься, как избежать дублирования кода, переопределяя метод `Collide` для классов `Unit`, `Building`, `Tower` и `Fence`, ведь все эти реализации выглядят абсолютно одинаково — отличие состоит только в типе указателя `this`. Помочь в этом может слегка нетривиальная идиома CRTP — Curiously recurring template pattern:

```
// Создаём шаблон класса Collider, только в нём будет переопределяться метод Collide
template <typename T>
struct Collider : GameObject {
  bool Collide(const GameObject& that) const final {
    // Статически приводим тип *this к типу const T&, потому что мы знаем,
    // что T — наш наследник (см. ниже)
    return that.CollideWith(static_cast<const T&>(*this));
  }
};

// Наследуем класс Unit от класса Collider<Unit>, который в свою очередь
// наследуется от GameObject.
class Unit final : public Collider<Unit> {
public:
  Unit(geo2d::Point position);

  // Переопределения методов CollideWith — метод Collide переопределять уже не нужно
  bool CollideWith(const Unit& that) const override;
  bool CollideWith(const Building& that) const override;
  bool CollideWith(const Tower& that) const override;
  bool CollideWith(const Fence& that) const override;

private:
  // ...
};
```

Конечно, можно просто накопипастить перегрузки метода `Collide` для каждого класса или воспользоваться макросами, но применение CRTP в данном случае избавляет от дублирования кода и обладает большей типобезопасностью.