just use latex

A Plane-Transformation Simplification-Algorithm

Introduction

Algorithm

```
func Simplify(t Transformation) Transformation {
    if len(t) < 2  { return t }
    if len(t) == 2 { return Simplify2(t[0], t[1]) }
    if len(t) == 3 { return Simplify3(t[0], t[1], t[2]) }
    return Simplify(Compose(
        Simplify(t[:len(t)-4]),
        Simplify4(t[len(t)-4], t[len(t)-3], t[len(t)-2], t[len(t)-1]),
    ))
}

func Simplify2(a, b Line) Transformation {
    if AreSameLine(a, b) { return Transformation{} }
    return Transfromation{a, b}
}

func Simplify3(a, b, c Line) Transformation {
    if len(Simplify2(a, b)) == 0 { return Transformation{c} }
    if len(Simplify2(b, c)) == 0 { return Transformation{a} }
    if AreParallel(a, b) && AreParallel(b, c) {
        return Transformation{ShiftBToC(a, b, c)}
    }
    a, b, c = RotateToParallelAndPerpendicular(a, b, c)
    return Compose(simplify2(a, b), Transformation{c})
}

func Simplify4(a, b, c, d Line) Transformation {
    f3 := simplify3(a, b, c)
    if len(f3) < 3 { return Simplify(Compose(f3, Transformation{d})) }
    l3 := simplify3(a, b, c)
    if len(l3) < 3 { return Simplify(Compose(Transformation{a}, l3)) }
    a, b, c = f3[0], f3[1], f3[2]
    if AreParallel(b, d) {
        return Transformation{ShiftBToC(a, b, d), c}
    }
    a, d = RotateBCToSame(a, c, b, d)
    return Transformation{a, d}
}
```

Correctness

Complexity

Examples