# Learn You a Physics for Great Good!

# >>> WORK IN PROGRESS <<<

## Examples / Box on an incline

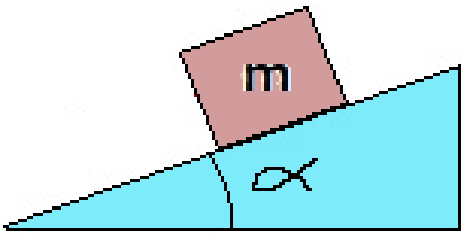
[src: Examples/Box_incline.lhs] Previous: Teeter Table of contents Next: Table of contents

Improvmenet: notation formulas tests

# Box on an incline

```
import Vector.Vector
```

All vectors are in newton.




Incline

Notation: fg = gravitational accelleration

m = mass of box

```
fg = V2 0 (-10)
m = 2
```

```
unit_normal :: Double -> Vector2 Double
unit_normal a = V2 (cos a) (sin a)
```

Force against the incline from the box:

```
f_l_ :: Vector2 Double -> Angle -> Vector2 Double
f_l_ fa a = scale ((magnitude fa) * (cos a)) (unit_normal (a-(pi/2)))
```

The normal against the incline:

```
fn :: Vector2 Double -> Angle -> Vector2 Double
fn fa a = negate (f_l_ fa a)
```

Friction free incline:

Resulting force:

```
fr :: Vector2 Double -> Angle -> Vector2 Double
fr fa a = (fn fa a) + fa
```

With friction:

$$F_{friction} = \mu * F_{normal} \iff \mu = \frac{F_{friction}}{F_{normal}}$$

```
us = 0.5
uk = 0.4
```

Add image how friction depends if there is movement.

 Friction Friction

```
type FricConst = Double
```

Friction:

friks = Fn * us, us = friction static

frikk = Fn * uk, uk = friction kinetic

We have the normal force and only needs the constants.

The current speed does not affect the friction.

```
motscalar :: FricConst -> Vector2 Double -> Scalar
motscalar u f = u * (magnitude f)
```

Från en rörelse eller vekt, fixa komplementet

```
enh_vekt :: Vector2 Double -> Vector2 Double
enh_vekt v  | magnitude v == 0 = (V2 0 0)
            | otherwise = scale (1 / (magnitude v)) v


motkrafts :: FricConst -> Scalar -> Vector2 Double -> Vector2 Double
motkrafts u s v = scale (u * s) (negate (enh_vekt v))

motkraftv :: FricConst -> Vector2 Double -> Vector2 Double -> Vector2 Double
motkraftv u n v = scale (u * (magnitude n)) (negate (enh_vekt v))
```

Now we just need to sum the force vectors:

```
fru :: Vector2 Double -> Angle -> FricConst -> Vector2 Double
fru fa a u = (fr fa a) + (motkraftv u (fn fa a) (fr fa a))

fru' :: Vector2 Double -> Angle -> FricConst -> Vector2 Double
fru' fa a u = (motkraftv u (fn fa a) (fr fa a))
```