Angles

This sheet explains how to make a sprite move (or point!) in an angle, and how to get the right angle to move (or point!) in a particular direction.

Moving at an angle

You can use *sin* and *cos* to move a sprite at an angle. These take an angle in degrees and give us the change in X and Y position we need to make.

```
player.posX += uw.sin(player.angle)
player.posY += uw.cos(player.angle)
```

If we want to move faster, or control the speed, we can multiply both the X and Y differences by a number:

```
player.posX += 3 * uw.sin(player.angle)
player.posY += 3 * uw.cos(player.angle)
```

👉 Check the sprite moves in the same direction as it's facing.

Pointing towards your finger

If you want to point towards something, to move in a particular direction, you can use *atan2*. This takes a difference in X/Y position and gives an angle in degrees. (This is a special version of inverse *tan()*.)

For example, to point the sprite toward your finger when you tap, we first need to work out the difference in X and Y position between your finger and the sprite. We'll call the difference in X `dx`, and the difference in Y will be `dy`.

```
world.onTap(e => {
    var dx = e.fingerX - player.posX
    var dy = e.fingerY - player.posY
    ...
})
```

We can then use the difference in position to get the right angle. This needs to go **inside the `onTap` block**.

```
    player.angle = uw.atan2(dx, dy)
```

👉 Check the sprite faces toward your finger when you tap.

Pointing towards a sprite

We can point towards any X/Y position in just the same way. For example, we could point toward another sprite.

```
    var dx = target.posX - player.posX
    var dy = target.posY - player.posY
```