Transforms

The following are the commonly used properties of transforms

Position

position - x,y,z values in world space

localPosition – x,y,z values relative to the parent

Rotation

rotation – a Quaternion. Great to interpolate with, hard for us to understand. This is what Unity stores internally

eulerAngles – the rotations around the x, y and z world axes, in degrees.

localEulerAngles – rotations around the axes relative to the parent object. This is what you see in the Inspector

right – the local x-axis. Setting this tends to preserve the up vector and change the forward vector.

up – the local y-axis

forward – the local z-axis

Scale

localScale – the scale of the object relative to its parent. This is what you see in the Inspector.

Some common operations

```
GameObject target;
```

I want to move towards another object at a fixed speed

```
        Vector3 targetPosition;
        float speed;
        transform.position = Vector3.MoveTowards(transform.position, targetPosition,
speed * Time.deltaTime);
```

MoveTowards is a static function of Vector3 that does smooth movement from one point ot another.

I want to point straight at another object

```
        transform.forward = target.position - transform.position;
```

We can set the forwards, right or up direction of a transform and it will normalize the vector we pass in, and rotate the object smoothly to face that direction.

I want to point at another object but stay horizontal.

```
            transform.forward = target.position - transform.position;
            // zero the rotation around x and z
            transform.eulerAngles = Vector3.up * transform.eulerAngles.y;
```

We can do this by eliminating any elevation (eulerAngles.x) or roll (eulerAngles.z) and just using the rotation around vertical (eulerAngles.y)

I want to track another object at a fixed angular speed

```
       Vector3 fwd = target.position - transform.position;
       Quaternion towards = Quaternion.LookRotation(fwd);
       transform.rotation = Quaternion.RotateTowards(transform.rotation, towards, speed *
Time.deltaTime);
```

Speed here is measured in angles per second

I want to track another object at a fixed angular speed while staying horizontal.

```
       Vector3 fwd = target.position - transform.position;
       angle = Mathf.Atan2(fwd.x, fwd.z) * Mathf.Rad2Deg;
       Vector3 angles = transform.eulerAngles;
       angles.y = Mathf.MoveTowardsAngle(angles.y, angle, speed * Time.deltaTime);
       transform.eulerAngles = angles;
```

Atan2 gives us the angle from the x-axis and the point (x,y) pair on a 2D plane in radians. We convert this to degrees to work with Euler angles, which are in degrees. We only manipulate the y part of the euler angle so that we stay horizontal.

Mathf.MoveTowardsAngle moves smoothly from one angle to another taking wraparound into account, so it never moves 350 degress the long way instead of -10 degrees the short way.