Applying a force at a position point

The most easy way to use the `orxObject_ApplyForce` function is the pass an `orxNULL` value for the last parameter which means to apply force to the centre of the object. Most of the time that is what you probably want.

I came across a scenario where I wanted to apply force always at the left hand edge, and always in the direction that the object was facing.

Here is the signature for the function:

`orxObject_ApplyForce (orxOBJECT *_pstObject, const orxVECTOR *_pvForce, const orxVECTOR *_pvPoint)`

The first two parameters are the object and force vector to apply. But the last parameter is the position to apply the force. This position is not a local coordinate, but rather, a world position vector. Nor is the vector relative to the object rotation.

Turned out to be mathematically trickier than I expected.

What I really wanted, was the ability to provide a point vector relative to the centre of the object, who's position and direction would remain constantly relative to the objects position and rotation.

That way I could provide this relative point to the `orxObject_ApplyForce` function.

Here's what I came up with:

```
void PostDebugSpot(orxVECTOR position) {
    orxObject_SetPosition(debugSpot, &position);
}

void Thrust(orxVECTOR localInfluencePoint, orxFLOAT power) {
    orxFLOAT FORCE = power;
    orxVECTOR thrust = { , -FORCE,  };

    orxVECTOR thrustDirection = { ,, };
    orxVECTOR thrustPosition = orxVECTOR_0;

    orxVECTOR objectPosition = orxVECTOR_0;
    orxObject_GetPosition(object, &objectPosition);

    orxFLOAT objectRadians = orxObject_GetRotation(object);
    orxFLOAT facingRotation = (orxMATH_KF_RAD_TO_DEG * objectRadians) + ;

    orxVector_2DRotate(&thrustDirection, &thrust, objectRadians);

    orxVector_2DRotate(&localInfluencePoint, &localInfluencePoint,
objectRadians);
```

```
    orxVector_Add(&thrustPosition, &objectPosition, &localInfluencePoint);

    orxObject_ApplyForce(object, &thrustDirection, &thrustPosition);
    PostDebugSpot(thrustPosition);
}

void DoInputs()
{
    if (orxInput_IsActive("Drive")) {
        orxVECTOR localInfluencePoint = { -25, ,  }; //relative position
from object centre
        orxFLOAT power = 0.5;
        Thrust(localInfluencePoint, power);
    }
}
```

I'll leave it to you, the reader to pick through the `Thrust()` function. But in short, you provide a vector, which is relative to the centre of the object. In this case, `{ -25, 0, 0 }` which means 25 pixels to the left of the object, and 0 vertically in the middle (my test object was 50 x 50 pixels).

The second parameter is the strength of the force to apply.

The routine checks the rotation of the object in order to work out which direction to apply the force. Then it uses the rotation, location, and the passed in vector to work out the relative offset position of the object and direction.

Finally, a small debug is placed on top of this position to ensure all is working as expected.

When all is done, applying force to the relative point, will give the following result: