Calculate Shadow

```
bool light::CalculateShadow(const intersection &inter, lightOutput &output){
	const LinkedList<shape *> &shapes = inter.w->shapes;
	intersection tmp;
	double tLight;

	if (CanShadow() && inter.s->CanBeShadowed())
	{
		ray r;
		r.v = output.L;
		r.p = inter.p + ALMOST_ZERO * r.v;

		tLight = Distance(inter.p, GetPosition()) / output.L.Length();

		for (shapes.MoveFirst(); !shapes.AtEnd(); shapes.MoveNext())
		{
			tmp.w = inter.w;
			shape *s = shapes.GetCurrent();
			if ( s->CanShadow() && s->CalculateIntersection(r,tmp) ){
				if( tmp.t < tLight ){
					output.d = output.d  * shadowColor;
					output.s = 0;
					return true;
				}
			}
		}
	}
	return false;
}

bool shape::CalculateIntersection(const ray &R, intersection &I){
	ray RTrans = TransformRay(R);

	if (Intersect(RTrans, I))
	{
		I.p = R.PointOnRay(I.t);
		I.n = TransformNormal(I.n);
		I.v = -R.v;
		I.n.Normalize();
		I.v.Normalize();
		return true;
	}
	else
		return false;
}

vector shape::TransformNormal(const vector &n){
	return inverseTranspose * n;
}

ray shape::TransformRay(const ray &r){
	ray result;
	result = r;
	result.p = inverseTransform * result.p;
	result.v = inverseTransform * result.v;
	return result;
}
```