Linear Interpolator

```
point LinearInterp::pointOnCurve(double s){

	// ***--- find the interval by the floor of s ---***
	int interval = s;

	// ***--- if s is less than zero return the first point ---***
	if(s < keys[0].t){
		keys[0].p;
	}
	//***--- if s is greater than the last point return the last point---***
	else if(s > keys[nKeys-1].t ){
		keys[nKeys-1].p;
	}

	// ***--- % along the curve is the decimal value of s ---***
	double percentage = s - interval;

	// ***--- Linear Interpolation calculation ---***
	return keys[interval].p + (percentage * (keys[interval+1].p - keys[interval].p));
}

point Interpolator::Evaluate(double t)  {
	int i = FindInterval(t);

	if(nKeys < 2){

		return pointOnCurve(0);

	}else if (!keys[i].constant){

		// Calculate s which is the % along and add to it i which is the interval
		double s = ((t-keys[i].t) / ( keys[i+1].t - keys[i].t )) + i;

		return pointOnCurve(s);
	}
	else
	{

		double s = 0;

		double speed = (keys[i+1].dist - keys[i].dist) /( keys[i+1].t - keys[i].t);
		double distance = keys[i].dist + (t - keys[i].t) * speed;

		int found = 0;

		while (distTable[found].d < distance){
			found++;
		}

		s = distTable[found].s + ( ( (distance - distTable[found].d) /
			(distTable[found+1].d - distTable[found].d) ) *
			(distTable[found+1].s - distTable[found].s) );


		return (pointOnCurve(s));
	}
}
```