UPPSALA
UNIVERSITET

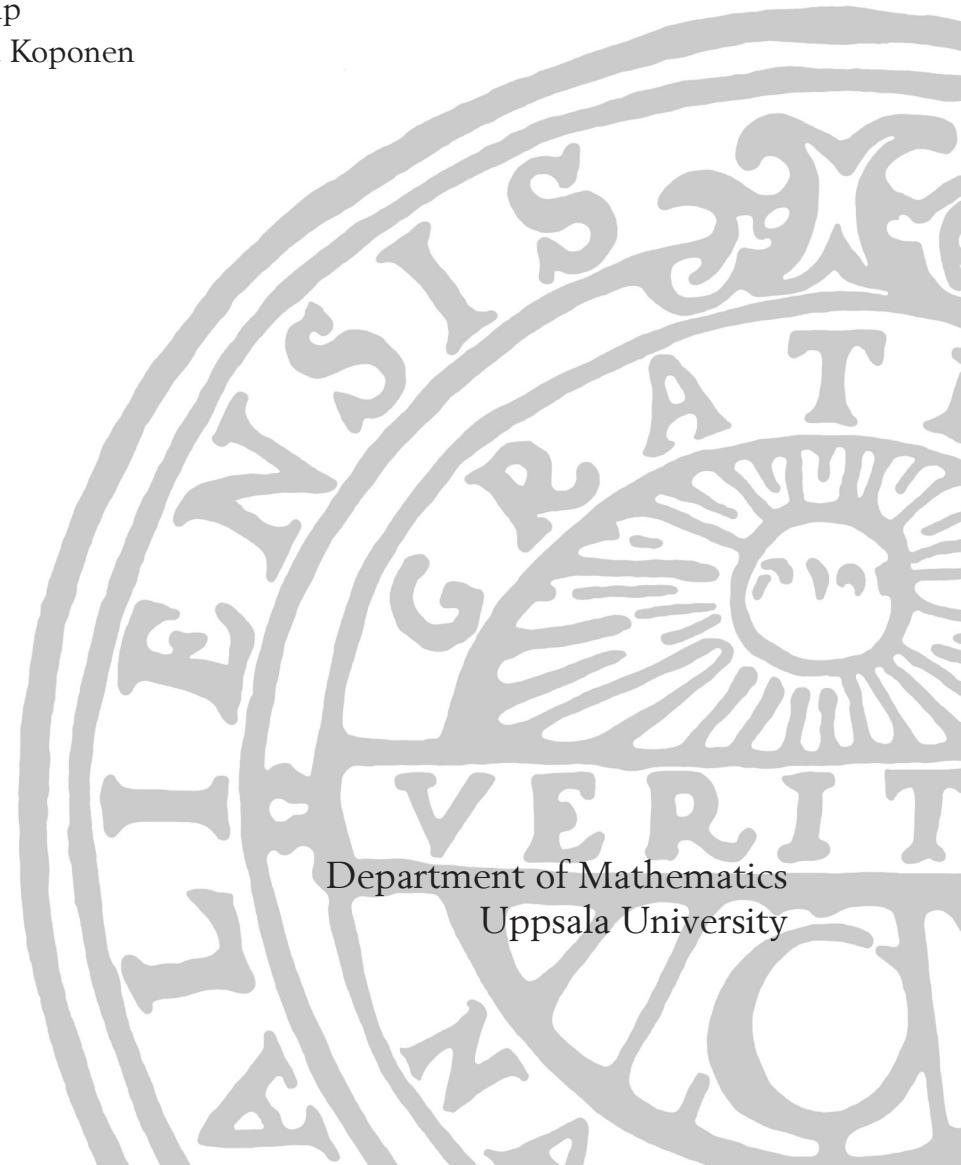
U.U.D.M. Project Report 2011:3

Barycentric and Wachspress coordinates in two dimensions: theory and implementation for shape transformations

Petter Lidberg

Examensarbete i matematik, 15 hp
Handledare och examinator: Vera Koponen

Februari 2011



Department of Mathematics
Uppsala University

Abstract

Barycentric coordinates and their generalisations, knowned by their family name Generalized Barycentric coordinates, are widely used in computer graphics. In particular for shape or volume transformations. Barycentric coordinates are constructed and limited to three reference points (i.e. the vertices on a triangle). Generalized Barycentric coordinates are constructed using an arbitary number of reference points.

This paper will explain and discuss Barycentric coordinates including their connection to classical geometry. This paper will also explain and discuss Wachspress coordinates – one of the first methods for generalizing Barycentric coordinates. This will be done in two dimensions.

This paper will also provide an implementation of Barycentric and Wachspress coordinates for shape transformation in two dimensions.

Key words

Barycentric coordinates, Medians, Cevians, Centroid, Ceva's theorem, Barycenter, Areal coordinates, Generalized Barycentric coordinates, Wachspress coordinates, Archimedes, Shape transformations, Actionscript 3, Adobe Flash.

Contents

Introduction

August Möbius discovered Barycentric coordinates in his groundbreaking work ”Der barycentrische Calcül, ein neues Hülfs- mittel zur analytischen Behandlung der Geometrie”, 1827 [1].

Barycentric coordinates are defined on arbitrary triangles, which in turn are defined by three vertices. These vertices act as reference points for all points contained inside and at the boundary of the triangle.

Because of Barycentric coordinates' great properties and applications, the problem of generalising them to work with more advanced polygons than triangles, has been the subject of intense research within the community. The problem is surprisingly difficult and have until quite recently been restricted to Wachspress coordinates.

These efforts are driven by the need to interpolate a set of points with an arbitrary number of reference points (i.e. vertices). If we want to control a shape encapsulated with something other than a triangle, but want the system to behave in a similar manner, then we need to extend Barycentric coordinates in some way. For example: It quickly gets too tediuos for a computer animator to move every point in a character for every movie frame. The character needs to be rigged in such a way that the animator can move just a couple of reference points for every frame. Barycentric coordinates solves this rigging problem in a great way. However, just three reference points are normally to few to articulate the character. This is a key example why there is a need to generalize Barycentric coordinates to function with more reference points (i.e. vertices).

This paper will explain and implement Barycentric and Wachspress coordinates in two dimensions. The implementations of these two coordinates are for shape transformations.

Preliminaries

Before we jump in and start explaining Barycentric and Wachspress coordinates there is a number of preliminaries that we need to cover.

1.1 Archimedes of Syracuse (287-212 B.C.)
Archimedes of Syracuse is generally considered to be the greatest mathematician of antiquity and perhaps of all time. He can be compaired to Newton and Gauss when it comes to genial innovation.

A lof of myths obviously surround such a legendary person as Archimedes. The most famous fact concerning his life is of course his classic exlamation ”Eureka, Eureka!” – the frase he shouted while running naked across town, shortly after he discovered that the amount of water which flowed over the tub was equal to the amount by which his body was immersed [2].

Archimedes of Syracuse was one of the greatest polymaths - he was a mathematician, engineer, inventor, physicist and astronomer. Greatly ahead of his time, I think Archimedes has more in common with the renaissance man Leonardo da Vinci than his ancient contemparies. It is no surprise that Leonardo writes about Archimedes in his notebooks [3].

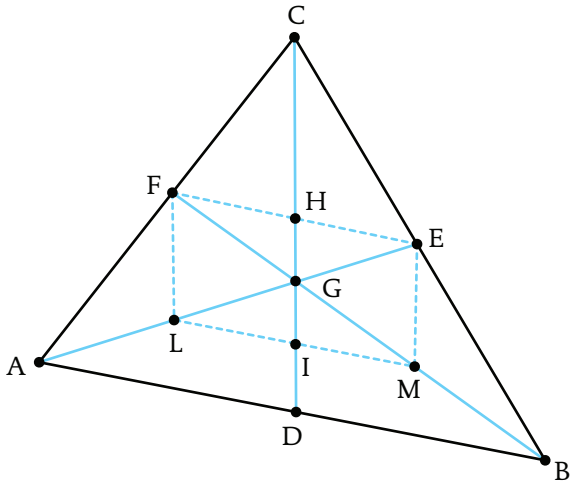
The unique library of Alexandria was founded at the time of the birth of Archimedes. The library attracted the greatest minds of the time, and quickly became the most famous centre of learning. Euclid probably died before the time of the arrival of Archimedes in Alexandria, so it is a fact that Archimedes had access to Euclid’s masterpiece of geometry: The “Elements” [4]. His studies in Alexandria became the foundation on which he built his career as a scientist and mathematician.

Following the Elements’ line of thought, Archimedes impressed the most as a geometrician. This paper is partly related to one of Archimedes finest works “On the Equilibrium of Planes” [5].

1.2 Cevians, Medians and concurrent lines
The line segment joining a vertex of a triangle to any given point on the opposite side is called a Cevian. Medians are Cevians that connect vertices with the midpoints of the opposite sides. Lines or segments are concurrent, when they all pass through one common point.

1.3 The Centroid of a triangle

The three Medians AE, BF and CD, where E, F and D are points on the opposite sides of facing vertices A, B and C in triangle ABC, are concurrent at point G (see figure below). This unique common point G for the three Medians is called the Centroid of the triangle.



Proof

Consider two of the three Medians, AE and BF, and let them meet in the point G. Let L and M be the midpoints of AG and BG. By Euclid VI.2 and VI.4 (see appendix), it follows that EF and LM are parallel to AB and that LM is half the length of AB. The angles FEG and GLM are equal by Euclid I.29 (see appendix). So the triangle EFL is congruent with the triangle ELM by Euclid I.4 (see appendix). Thus FL and EM have the same lenght. It follows by Euclid I.27 (see appendix) that FL and EM are parallel. So EFLM is a parallelogram (see figure above). Since the diagonals of a parallelogram bisect each other, we have EG = GL = AL, GF = GM = BM. Triangles CEF and CBA are similar (EF is parallel with AB). Recall that D is the midpoint of AB, so the line CD must pass the line EF in a midpoint H (see figure). Triangle EFG is congruent with triangle GLM. Triangle EFG is similar to triangle ABG. Similar triangles are triangles that have the same shape. Triangle EFM is congruent with triangle FLM, so the line FE equals LM, which in turn is half the length of AB. Triangle EGH is similar to triangle ADG, so the angle ADG equals the angle EHG. Triangle EFH is similar to triangle EFL so the angle EFL equals EHG so the line CD is parallel to the lines FL and EM. Thus the three Medians AE, BF and CD are concurrent at point G.

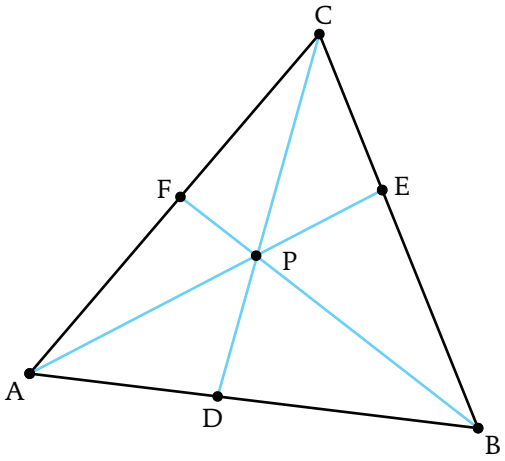
This is what Archimedes geometrically deduced as the center of gravity of a triangular plate of uniform density in his work "On the Equilibrium of Plane Figures" [6].

So if we want to balance a triangle on a fulcrum (a support or prop) with equal masses at the vertices of the triangle, then the Centroid or balance point is simply dependent on the distances from the vertices.

1.4 Ceva's theorem

The three Cevians AE, BF and CD, where E, F and D are points on the opposite sides of facing vertices A, B and C in triangle ABC, are concurrent at point P, if and only if [7]:

$$\frac{AD}{DB} \cdot \frac{BE}{EC} \cdot \frac{CF}{FA} = 1$$



Proof

(i) Suppose AE, BF, CD are concurrent at point P.

Construct a line that is parallel to AB and goes through C. Extend the lines BE and BF beyond the triangle ABC until they meet the newly drawn parallel line at points G and H respectively (see figure 1.4a below). By using Euclid's proposition I.29 (see appendix), we identify four pairs of similar triangles (see figure 1.4a). Similar triangles are triangles that have the same shape. Their corresponding angles are the same and their corresponding sides are all in the same proportion. From all this, we get the following ratios:

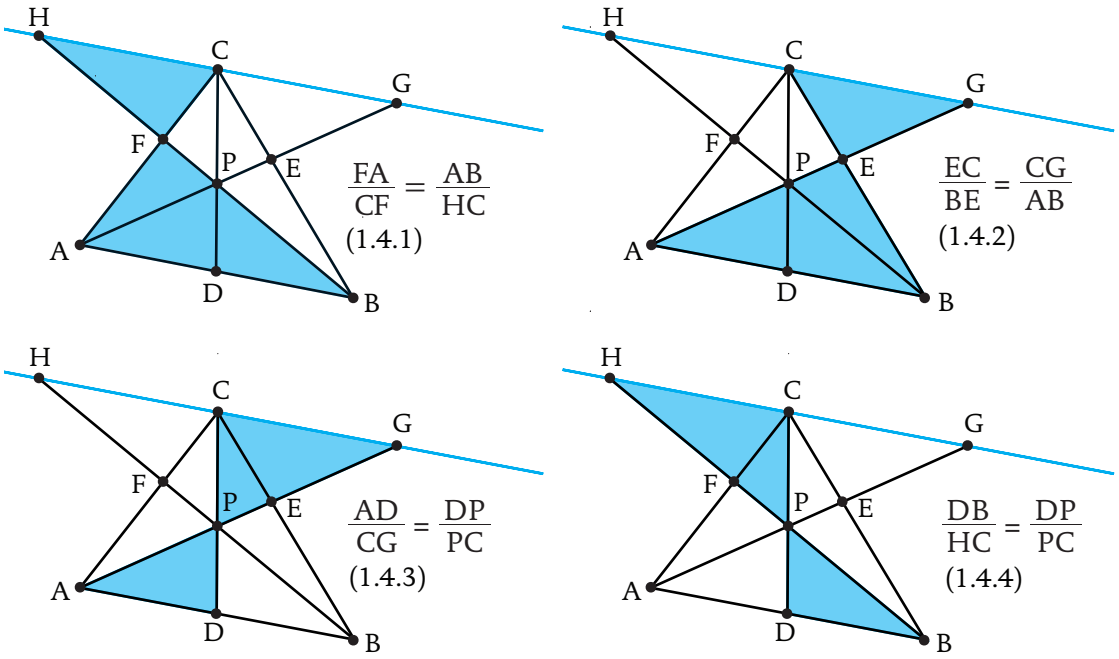


figure 1.4a

By combinding (1.4.4) and (1.4.3), we get:

$$\frac{DB}{HC} = \frac{AD}{CG} \leftrightarrow \frac{DB}{AD} = \frac{HC}{CG} \quad (1.4.5)$$

The product of (1.4.2), (1.4.1) and (1.4.5) is:

$$\frac{EC}{BE} \cdot \frac{FA}{CF} \cdot \frac{DB}{AD} = \frac{CG}{AB} \cdot \frac{AB}{HC} \cdot \frac{HC}{CG} = 1 \quad (1.4.6)$$

By rearranging the terms of (1.4.6), we get the ratios in Ceva's theorem:

$$\frac{AD}{DB} \cdot \frac{BE}{EC} \cdot \frac{CF}{FA} = 1$$

(ii) Suppose:

$$\frac{AD}{DB} \cdot \frac{BE}{EC} \cdot \frac{CF}{FA} = 1 \longleftrightarrow AD \cdot BE \cdot CF = DB \cdot EC \cdot FA \quad (1.4.7)$$

Let P be the concurrent point of the Cevians AE and BF. Define X as the concurrent point on AB and on the extended line CP (see figure 1.4b). By (1.4.7), we get:

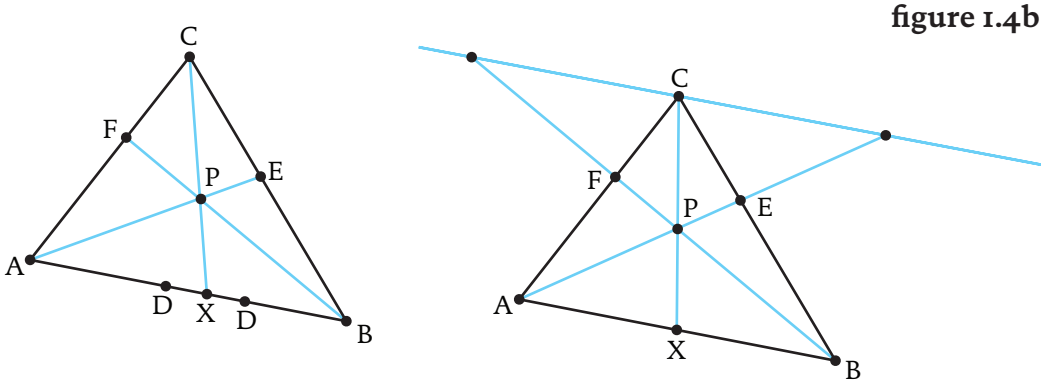
$$\frac{AX}{XB} \cdot \frac{BE}{EC} \cdot \frac{CF}{FA} = 1 \longleftrightarrow AX \cdot BE \cdot CF = XB \cdot EC \cdot FA \quad (1.4.8)$$

Dividing (1.4.7) with (1.4.8) gives:

$$\frac{AD}{AX} = \frac{DB}{XB} \longleftrightarrow AD \cdot XB = AX \cdot DB \longleftrightarrow \frac{AD}{DB} = \frac{AX}{XB} \quad (1.4.9)$$

If X does not coincide with D, then X lies on segment AD or DB (see figure 1.4b). If X lies on AD then AX < AD and XB > DB, which contradicts (1.4.9). The same contradiction occurs if X lies on segment AD (because of AD < AX and DB > XB). Thus X coincides with D.

figure 1.4b



The theorem is named after the Italian mathematician Giovanni Ceva, who in 1678 published this very useful theorem.

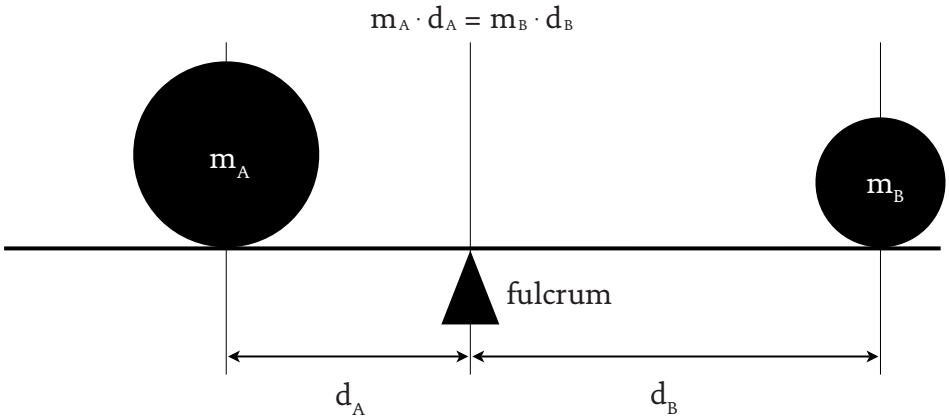
1.5 Barycenter

The term Barycenter is most often used as a another word for the center of mass of an object. One can imagine the Barycenter as the center where all of the mass of the object is positioned. This approach is often used in physics and mechanics, to simplify calculations concerning gravitational or other forces acting on an object. The word, Barycenter, originates from the Greek word "barus" which means "weight" [8].

1.6 Archimedes' Law of the lever

Archimedes decrees in his Law of the lever that "two magnitudes balance at distances reciprocally proportional to their magnitudes" [9].

This notion is perhaps best explained in the following way: Place masses, m_A and m_B, at opposite sides of a massless rodd. Place the rodd with masses m_A and m_B on a fulcrum. A fulcrum is litterally a support, a prop (see figure below). Archimedes' Law of the lever states that a system is at an equilibrium when its moments on respective sides of the fulcrum are equal. Moment is equal to mass times displacement:



However, it is important to state that Archimedes never directly, in a physical way, defined the concept of the center of gravity. Nor did he multiply two such different quantities as a weight and a distance. As in all of his works, Archimedes stuck to the format establised by Euclid. He expresses balance by the equality of two proportions:

$m_A : d_B = m_B : d_A$ (where m_A is to d_B as m_B is to d_A)

And he states that the weights balance at distances inversely proportional to their magnitudes.

1.7 Convex and concave polygons

In Euclidean space, a polygon is convex if every line segment between every two vertices remains inside or at the boundary of the polygon.

Every internal angle in a convex polygon is less than 180 degrees (= π radians). Conceptually, this means that all the vertices of the polygon will point outwards, away from the interior of the shape. Think of it as a 'bulging' polygon (see figure 1.7a).

Triangles are always convex.

A non-convex polygon is called a concave polygon (see figure 1.7b).

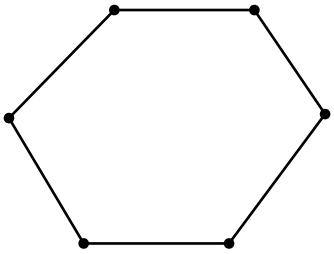


figure 1.7a

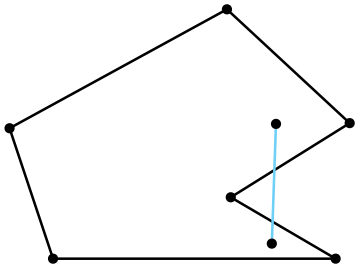


figure 1.7b

1.8 Cage

A cage is a piecewise linear closed polygon in 2-dimesions (or polyhedron in 3-dimensions). The cage encapsules the relevant set of points with a minimum of three reference points.

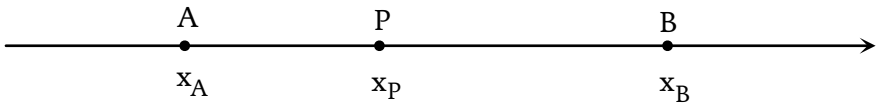
By construction, both Barycentric and Wachspress coordinates use a cage. As will be explained later, the shape of the cage is of paramount importance.

Barycentric coordinates

August Möbius discovered Barycentric coordinates in his genius work ”Der barycentrische Calcül, ein neues Hülfs- mittel zur analytischen Behandlung der Geometrie”, 1827 [10].

2.1 Barycentric coordinates construction

In defining Barycentric coordinates, we start by placing two masses m_A and m_B at two fixed points A and B on the real line (see figure below). If $m_A + m_B \neq 0$ and the two masses are both positive (or both negative) then we can calculate the Barycenter P using Archimedes' Law of the lever:



$A = x_A$, $B = x_B$, $P = x_P$, $x_A \neq x_B$, $m_A + m_B \neq 0$

$m_A(x_P - x_A) = m_B(x_B - x_P)$

$m_A x_P - m_A x_A = m_B x_B - m_B x_P$

$m_A x_P + m_B x_P = m_A x_A + m_B x_B$

$x_P(m_A + m_B) = m_A x_A + m_B x_B$

$$x_p = \frac{m_A}{m_A + m_B} x_A + \frac{m_B}{m_A + m_B} x_B$$

$x_p = w_A x_A + w_B x_B$

$P = w_A A + w_B B$

where $w_A = \frac{m_A}{m_A + m_B}$ and $w_B = \frac{m_B}{m_A + m_B}$

The multipliers or weights w_A and w_B are called the Barycentric coordinates of P relative to the points A and B.

We continue and set up Barycentric coordinates in the plane of triangle ABC, where the three masses m_A, m_B and m_C are placed at the vertices of the triangle ABC. If the sum of m_A, m_B and m_C is greater than zero, then we can determine the Barycenter P according the following formula:

$P = w_A A + w_B B + w_C C$

where $w_A = \frac{m_A}{m_A + m_B + m_C}$ $w_B = \frac{m_B}{m_A + m_B + m_C}$ $w_C = \frac{m_C}{m_A + m_B + m_C}$

and $m_A + m_B + m_C > 0$

The three multipliers or weights w_A, w_B and w_C of the points A, B and C are the Barycentric coordinates of P relative to vertices A, B and C.

This can be thought of in the following manner; in order to balance a massless solid triangle with different masses at its vertices on a fulcrum, we systematically balance the triangle between A and BC, B and CA, C and AB, in order to find the Barycenter P.

2.2 Normalized Barycentric coordinates

The Barycentric coordinates that we have defined above are automatically normalized, since $w_A + w_B + w_C = 1$. In this case, the Barycentric coordinates of a point within the triangle ABC are unique (see proof in chapter 2.5).

However, if we identify $w_A = m_A$, $w_B = m_B$ and $w_C = m_C$ (and don't divide with the sum of $m_A + m_B + m_C$) then our Barycentric coordinates become homogenous barycentric coordinates. The coordinates are homogenous because masses km_A, km_B and km_C, where the scalar $k \neq 0$, determine the same point as m_A, m_B and m_C.

2.3 Useful formula for calculating the area of a triangle

There are several formulas for calculating the area of a triangle. However, a very useful formula is as follows [11]:

$$\text{Area of triangle ABC} = \text{the absolute value of } \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} =$$

$$\text{the absolute value of } (x_A y_B + x_C y_A + x_B y_C - x_C y_B - x_A y_C - x_B y_A)$$

where $A = (x_A, y_A)$, $B = (x_B, y_B)$ and $C = (x_C, y_C)$

Note that we take the absolute value of the determinant expression. This is to ensure that we get a positive area.

2.4 Areal coordinates

Normalized Barycentric coordinates in triangles defined on the plane are also known as Areal coordinates. This is because normalized Barycentric coordinates are the actual areas of the sub triangles PAC, PAB and PBC normalized by the total area of triangle ABC. The Barycentric coordinate for the respective vertex correspond to the area of the sub triangle formed on the opposite side of the vertex (divided by the total area of triangle).

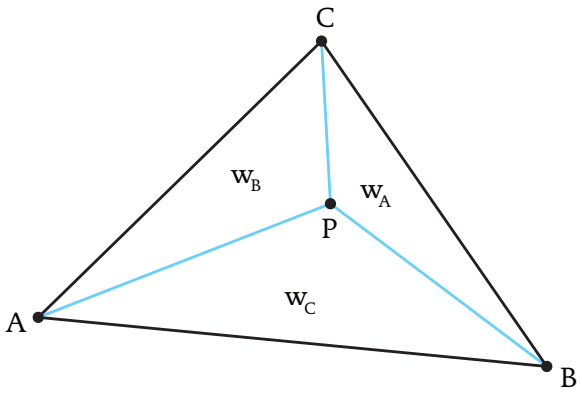
Proof

Suppose $P = w_A A + w_B B + w_C C$, $w_A + w_B + w_C = 1$ and $0 \le w_A, w_B, w_C \le 1$

$A = (x_A, y_A)$, $B = (x_B, y_B)$, $C = (x_C, y_C)$, $P = (x_P, y_P)$

$x_P = w_A x_A + w_B x_B + w_C x_C$

$y_P = w_A y_A + w_B y_B + w_C y_C$



We want to show that:

$$w_A = \frac{\text{area of BCP}}{\text{area of ABC}}$$

$$w_B = \frac{\text{area of CAP}}{\text{area of ABC}}$$

$$w_C = \frac{\text{area of ABP}}{\text{area of ABC}}$$

Recall from (2.3):

$$\text{Area of ABC} = \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} = x_A y_B + x_C y_A + x_B y_C - x_C y_B - x_A y_C - x_B y_A$$

So we get:

$$\text{Area of BCP} = \frac{1}{2}\begin{vmatrix} x_B & y_B & 1 \\ x_C & y_C & 1 \\ x_P & y_P & 1 \end{vmatrix} = \frac{1}{2}[x_B y_C + x_P y_B + x_C y_P - x_P y_C - x_B y_P - x_C y_B] =$$

$$\frac{1}{2}[x_B y_C + (w_A x_A + w_B x_B + w_C x_C) y_B + x_C (w_A y_A + w_B y_B + w_C y_C)$$
$$- (w_A x_A + w_B x_B + w_C x_C) y_C - x_B (w_A y_A + w_B y_B + w_C y_C) - x_C y_B] =$$
$$\frac{1}{2}[x_B y_C + w_A x_A y_B + w_B x_B y_B + w_C x_C y_B + w_A x_C y_A + w_B x_C y_B + w_C x_C y_C$$
$$- w_A x_A y_C - w_B x_B y_C - w_C x_C y_C - w_A x_B y_A - w_B x_B y_B - w_C x_B y_C - x_C y_B] =$$
$$\frac{1}{2}[x_B y_C - x_C y_B + w_A (x_A y_B + x_C y_A - x_B y_A - x_A y_C) + w_B (x_C y_B - x_B y_C)$$
$$+ w_C (x_C y_B - x_B y_C)] =$$
$$\frac{1}{2}[x_B y_C - x_C y_B + w_A (x_A y_B + x_C y_A - x_B y_A - x_A y_C) + (w_B + w_C)(x_C y_B - x_B y_C)] =$$
$$\frac{1}{2}[x_B y_C - x_C y_B + w_A (x_A y_B + x_C y_A - x_B y_A - x_A y_C) + (1 - w_A)(x_C y_B - x_B y_C)] =$$
$$\frac{1}{2}[x_B y_C - x_C y_B + w_A (x_A y_B + x_C y_A - x_B y_A - x_A y_C) + x_C y_B - x_B y_C - w_A x_C y_B + w_A x_B y_C] =$$
$$\frac{1}{2} w_A [x_A y_B + x_C y_A - x_B y_A - x_A y_C - x_C y_B + x_B y_C] =$$
$$\frac{1}{2} w_A [x_A y_B + x_C y_A + x_B y_C - x_C y_B - x_A y_C - x_B y_A] = \frac{1}{2} w_A \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} =$$

$$w_A \cdot (\text{area of ABC}) \longleftrightarrow w_A = \frac{\text{area of BCP}}{\text{area of ABC}}$$

Similarly,

$$\text{Area of CAP} = \frac{1}{2}\begin{vmatrix} x_C & y_C & 1 \\ x_A & y_A & 1 \\ x_P & y_P & 1 \end{vmatrix} = \frac{1}{2} w_B \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} =$$

$$w_B \cdot (\text{area of CAP}) \quad \longleftrightarrow \quad w_B = \frac{\text{area of CAP}}{\text{area of ABC}}$$

and

$$\text{Area of ABP} = \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_P & y_P & 1 \end{vmatrix} = \frac{1}{2} w_C \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} =$$

$$w_C \cdot (\text{area of ABC}) \quad \longleftrightarrow \quad w_C = \frac{\text{area of ABP}}{\text{area of ABC}}$$

See appendix for further details.

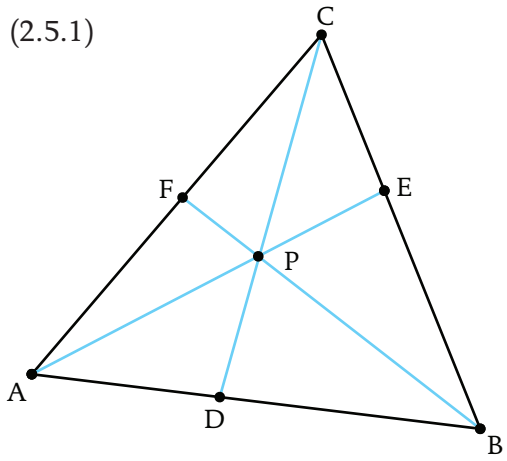
2.5 Unique Barycenter of Areal coordinates

Areal coordinates define a unique Barycenter P inside or at the boundary of an arbitrary triangle ABC.

Proof

This proof uses the fact that areas of triangles with equal altitudes are proportional to the bases of the triangle. Using this fact and by referring to figure below, we get:

$$\frac{AD}{DB} = \frac{CAD}{CDB} = \frac{PAD}{PDB} \Rightarrow \frac{CAD - PAD}{CDB - PDB} = \frac{CAP}{BCP} \quad (2.5.1)$$



This is motivated by:

$$\frac{a}{b} = \frac{c}{d} = k \Rightarrow c = dk,\ a = bk$$

$$\frac{a - c}{b - d} = \frac{bk - dk}{b - d} = \frac{k(b - d)}{b - d} = k = \frac{a}{b}$$

Similarly:

$$\frac{BE}{EC} = \frac{AEB}{AEC} = \frac{BPE}{CPE} \Rightarrow \frac{AEB - BPE}{AEC - CPE} = \frac{ABP}{CAP} \quad (2.5.2)$$

$$\frac{CF}{FA} = \frac{FCB}{ABF} = \frac{FPC}{AFP} \Rightarrow \frac{FCB - FPC}{ABF - AFP} = \frac{BCP}{ABP} \quad (2.5.3)$$

By multipling (2.5.1), (2.5.2) and (2.5.3) together we get:

$$\frac{AD}{DB} \cdot \frac{BE}{EC} \cdot \frac{CF}{FA} = \frac{CAP}{BCP} \cdot \frac{ABP}{CAP} \cdot \frac{BCP}{ABP} = 1 \quad (2.5.4)$$

Note the following about (2.5.4):
(i) The internal triangles CAP, BCP, ABP are the Areal coordinates for ABC.
(ii) The first part of the equation are the same ratios used in Ceva's theorem, so they define a unique point P.

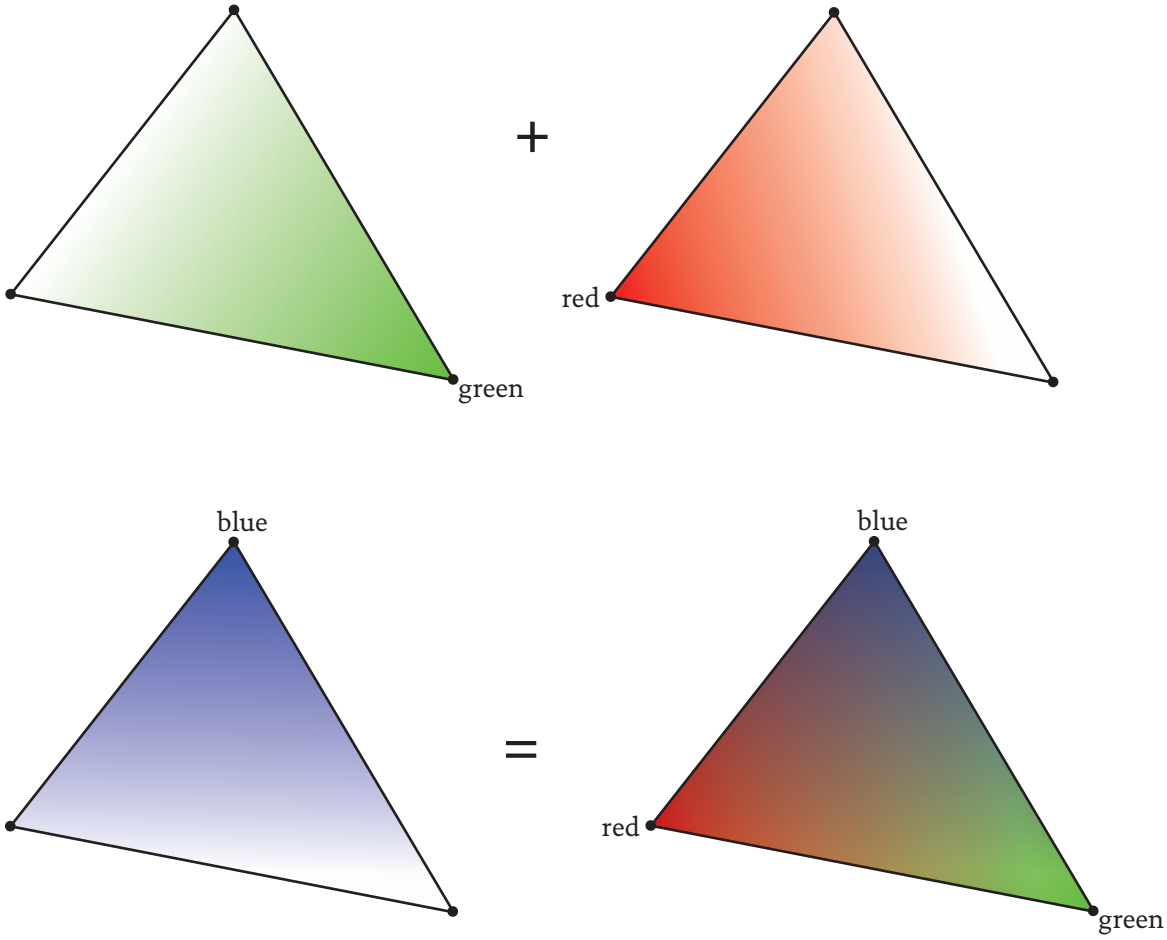
By combining our observations about (2.5.4), we have proved that Areal coordinates define a unique point P. The point is inside or at the boundary of triangle ABC because the Areal coordinates' sum (= the sum of the internal triangles' CAP, BCP and ABP areas) never can exceed the total area of ABC.

2.6 Application of Barycentric coordinates - Gouraud shading

A good example of using Barycentric coordinates for calculating something else than positions is Gouraud shading. It was invented by Henri Gouraud in 1971 [12]. His shading technique is an effective method of adding a smooth feel to a triangle by linearly interpolating a color or shade.

Gouraud shading achieves its smooth shading, by interpolating the normals at each interior point from the vertex normals at the triangle's three vertices. The calculated normal for the interior point is then used for calculating the right color or shade. The calculation of the normals for the interior point can be beautifully handled by barycentric coordinates. This can be illustrated by representing each of the triangle's vertex by a pure color, for example red, blue and green. The correct color (= combination of red, blue and green) for each of the triangle's interior points are the same as its barycentric coordinates (see figure 2.6 below).

figure 2.6



Generalized Barycentric coordinates

Considerable efforts have been put into extending Barycentric coordinates to function with a richer geometric structure while keeping the properties of Barycentric coordinates. The coordinates are often denoted with the family name; Generalized Barycentric coordinates.

These efforts are driven by the need to interpolate a set of points with a arbitrary number of reference points (i.e. vertices). If we want to control a shape encapsulated with something other than a triangle, but want the system to behave in a similar manner, then we need to extend Barycentric coordinates in some way. For example: It quickly gets to tediuos for a computer animator to move every point in a character for every movie frame. The character needs to be rigged in such a way that the animator can move just a couple of reference points for every frame. Barycentric coordinates solves this rigging problem in a great way. However, just three reference points are normally to few to articulate the character. This is a key example why there is a need to generalize Barycentric coordinates to function with more reference points (i.e. vertices).

Despite strives, it is not obvious how to make this generalization from triangles to n-sided polygons. The key issue that rises is the problem when the domain forms a non-convex polygon or polyhedron.

One of the first metod to generalize Barycentric coordinates was made in 1975 by Eugene Wachspress. His coordinates, called Wachspress coordinates, are presented in the next chapter.

Wachspress coordinates

Almost all work on generalizations of Barycentric coordinates was, until quite recently, restricted to Eugene Wachspress paper from 1975 [13]. The coordinates he proposed logically bears his name; Wachspress coordinates. Wachspress certainly got his inspiration from areal coordinates, as his coordinates also deals with ratios of areas. However, Wachspress deals with signed areas. Signed areas, in his construction, are simply areas that can be negative. Wachspress coordinates are only defined on convex polygons, which is it's major drawback.

4.1 Wachspress coordinates construction

Wachspress coordinates are only defined on the interior of convex polygons. Wachspress coordinates are constructed on convex polygons in the following way:

(i) Start by arranging the ordering of the Euclidean reference points $v_1,\ldots,v_n$ counter-clockwise around the interior point p and form the n-sided polygon with $v_1,\ldots,v_n$ as vertices (see figure 4.1 below).

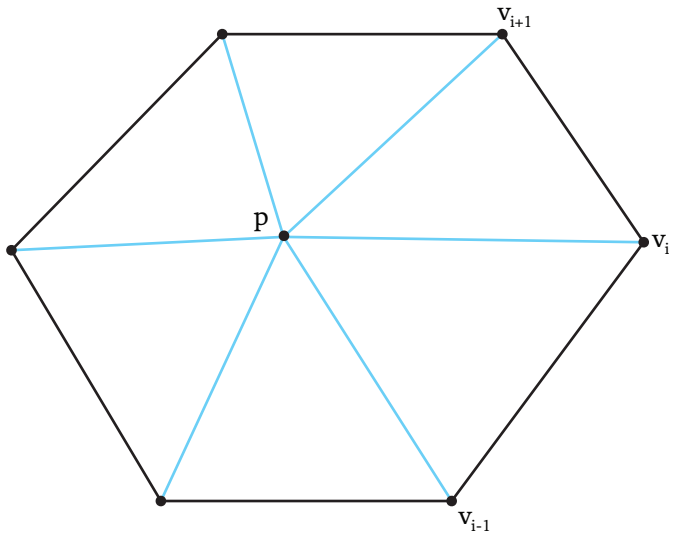


figure 4.1

(ii) Calculate the weights using Wachspress' formula:

$$\lambda_i(p) = \frac{A(v_{i-1}, v_i, v_{i+1})}{A(v_{i-1}, v_i, p)\,A(v_i, v_{i+1}, p)}$$

where A(a,b,c) is the signed triangle area formed by the points (a,b,c). Recall that a signed area is an area that can be negative. However, if the vertices are ordered correctly and the polygon is convex then then areas will positive.

In this way, the signed areas are incrementally defined by vertices v_{i-1}, v_i, v_{i+1} together with the interior point p.

(iii) Normalize each $\lambda_i(p)$

$$w_i(p) = \frac{\lambda_i(p)}{\sum_{j=1}^{n} \lambda_j(p)}$$ where n is the number of vertices

In this way, Wachspress coordinates are expressed in terms of rational polynomials. As long as the reference points (its cage) form a convex polygon, Wachspress's coordinates works beautifully. Note that Wachspress coordinates, on the boundary of the polygon, are not defined because of a division by zero – the areas $A(v_{i-1}. v_i, p)$ and $A(v_i, v_{i+1}, p)$ will be zero.

4.2 Numerical example

Consider a polygon with the following vertex points: A = (-4,-5), B = (9,2), C = (5,8), D = (-10,5). The polygon encapsulates a point P=(-2,4). See figure 3.2 below. If we move vertex point B to B'=(5,4), where will point P be placed? This is a good use of Wachspress coordinates.

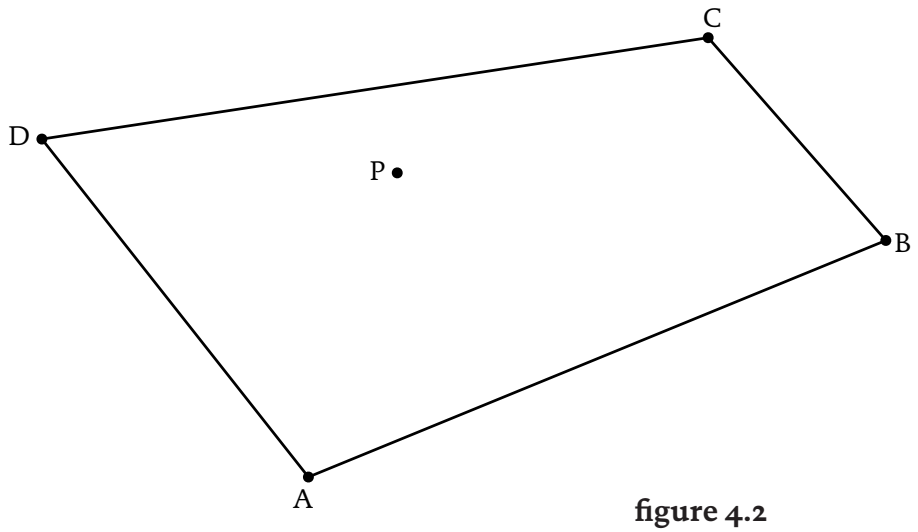


figure 4.2

$$\lambda_A = \frac{\text{area}(D,A,B)}{\text{area}(D,A,P)\cdot\text{area}(A,B,P)} = \frac{86}{37\cdot 51.5} = 0.04513251115192863$$

$$\lambda_B = \frac{\text{area}(A,B,C)}{\text{area}(A,B,P)\cdot\text{area}(B,C,P)} = \frac{53}{51.5\cdot 29} = 0.03548711081352528$$

$$\lambda_C = \frac{\text{area}(B,C,D)}{\text{area}(B,C,P)\cdot\text{area}(C,D,P)} = \frac{51}{29\cdot 19.5} = 0.09018567639257294$$

$$\lambda_D = \frac{\text{area}(C,D,A)}{\text{area}(C,D,P)\cdot\text{area}(D,A,P)} = \frac{84}{19.5\cdot 37} = 0.11642411642411643$$

Normalize:

$$\sum_{j=A,B,C,D}^{4} \lambda_j(P) = \lambda_A + \lambda_B + \lambda_C + \lambda_D = 0.28722941478214326$$

$$w_A = \frac{0.04513251115192863}{0.28722941478214326} = 0.04513251115192863$$

$$w_B = \frac{0.03548711081352528}{0.28722941478214326} = 0.12354970969961908$$

$$w_C = \frac{0.09018567639257294}{0.28722941478214326} = 0.31398482102181857$$

$$w_D = \frac{0.11642411642411643}{0.28722941478214326} = 0.40533493588190256$$

We check that we get point P with our newly calculated coordinates:
$P = w_A A + w_B B + w_C C + w_D D = (-2,4)$, which is correct.

Calculate the new position for point P:
$P' = w_A A + w_B B' + w_C C + w_D D \approx (4.08,3.59)$

4.3 Alternative formulation for Wachspress coordinates

Meyer, Lee, Barr, and Desbrun [14] constructed an alternative formulation for Wachspress coordinates that uses angles α and β (see figure 4.3 below).

$$\lambda_i(p) = \frac{\cot\alpha_{i-1} + \cot\beta_i}{\|v_i - p\|^2}$$

$$w_i(p) = \frac{w_i(p)}{\sum_{j=1}^{n} w_j(p)} \text{ where n is the number of vertices}$$

Proof

We want to prove that:

$$\lambda_i(p) = \frac{A(v_{i-1}, v_i, v_{i+1})}{A(v_{i-1}, v_i, p)\,A(v_i, v_{i+1}, p)} = \frac{\cot\alpha_{i-1} + \cot\beta_i}{\|v_i - p\|^2}$$

This can be proved by using Heron's area formula [15]:

$$\text{Area}(a,b,c) = \frac{a \cdot b \cdot \sin(C)}{2} \text{ where C is the angle formed by sides a and b.}$$

$$\lambda_i(p) = \frac{A(v_{i-1}, v_i, v_{i+1})}{A(v_{i-1}, v_i, p)\,A(v_i, v_{i+1}, p)} = \text{[Heron's formula]} =$$

$$\frac{\sin(\alpha_{i-1} + \beta_i) \cdot \|v_i - v_{i-1}\| \cdot \|v_i - v_{i+1}\|}{\sin(\alpha_{i-1}) \cdot \|v_{i-1} - v_i\| \cdot \|v_i - p\|^2 \cdot \sin(\beta_i) \cdot \|v_i - v_{i+1}\|} = \frac{\sin(\alpha_{i-1} + \beta_i)}{\sin(\alpha_{i-1}) \cdot \sin(\beta_i) \cdot \|v_i - p\|^2} =$$

$$[\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)] = \frac{\sin(\alpha_{i-1})\cos(\beta_i) + \cos(\alpha_{i-1})\sin(\beta_i)}{\sin(\alpha_{i-1}) \cdot \sin(\beta_i) \cdot \|v_i - p\|^2} =$$

$$\frac{\sin(\alpha_{i-1})\cos(\beta_i) + \cos(\alpha_{i-1})\sin(\beta_i)}{\sin(\alpha_{i-1}) \cdot \sin(\beta_i) \cdot \|v_i - p\|^2} = [\cot\theta = \frac{\sin\theta}{\cos\theta}] = \frac{\cot(\beta_i)}{\|v_i - p\|^2} + \frac{\cot(\alpha_{i-1})}{\|v_i - p\|^2} =$$

$$\frac{\cot\alpha_{i-1} + \cot\beta_i}{\|v_i - p\|^2}$$

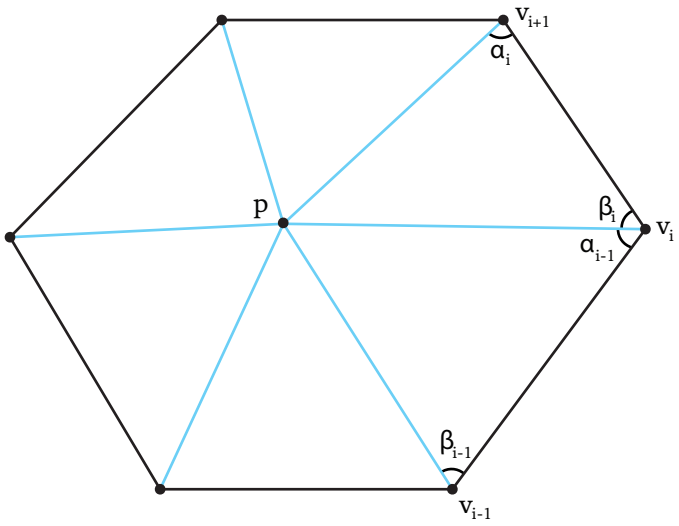


figure 4.3

4.4 Wachspress coordinates on concave polygons

The alternative formula in chapter 4.3 helps us understand why Wachspress coordinates does not work on concave (non-convex) polygons.

The trigometric cotangent function is not definied for angles equal to any positive or negative integer multipel of 180 degrees (= π radians). As this can only occur if α_{i-1} or β_i are 180 degrees, the angle sum ($\alpha_{i-1} + \beta_i$) must be equal or greater than 180 degrees. This, in turn, can only occur if the polygon is concave (see figure 4.4 below).

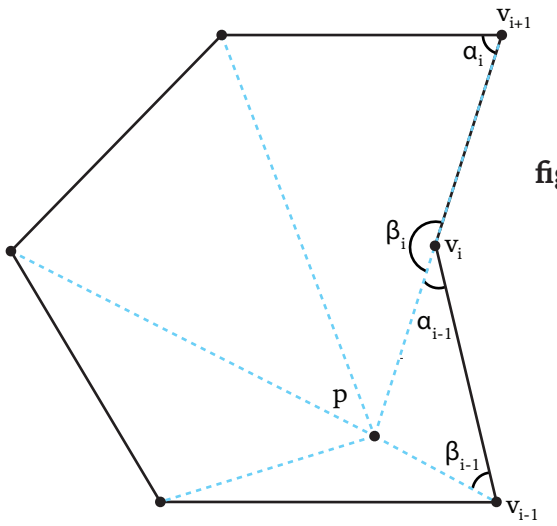


figure 4.4

Implementations of Barycentric and Wachspress coordinates for shape transformations

The implementations for Barycentric and Wachspress coordinates provided in this paper are for shape transformations. The implementations are written in Actionscript 3, a script language used in Adobe Flash. The source code is provided in the end of this paper.

5.1 Shape transformation using Barycentric coordinates

This shape transformation encapsulates a star shaped octagon with a triangle (see figure 5.1a). The octagon's eight vertices are expressed as Barycentric coordinates defined by the triangle's three Euclidean coordinates. In this way, we can control the octogon's shape through the three reference/vertex points. If we move the system's vertex points (its cage), the octogon's shape will follow accordingly (see figure 5.1b).

For an interactive demo see:
http://www.lidberg.se/math/shapetransforms/barycentric.html

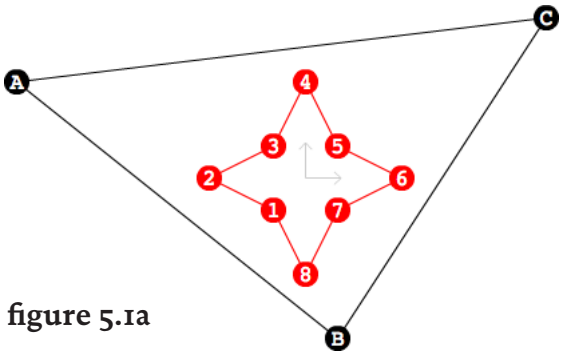


figure 5.1a

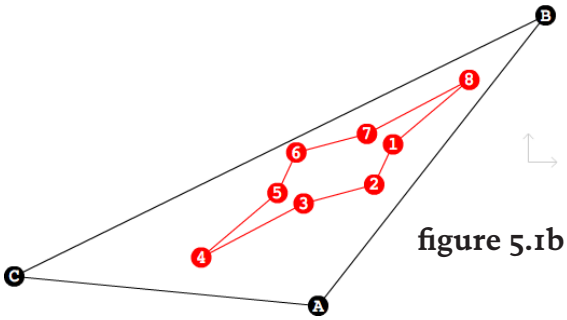


figure 5.1b

5.2 Shape transformation using Wachspress coordinates

This shape transformation encapsulates a star shaped octagon with a pentagon (a polygon with five vertices, see figure 5.2a). The octagon's eight vertices are expressed as Wachspress coordinates defined by the pentagon's five Euclidean coordinates. In this way, we can control the octagon's shape through five reference/vertex points. If we move the system's vertex points (its cage), the octagon's shape will follow accordingly (see figure 5.2b).

Shape transformation using Wachspress coordinates gives us the advantage of using a higher number of reference points than using Barycentric coordinates. But it also comes with the drawback that the cage formed by the reference points must be convex. If the cage forms a concave (non-convex) polygon, the system's values will start to behave in a nonsmooth manner, and may even "pop", meaning that they will fall outside of the cage (see figure 5.2c).

For an interactive demo see:
http://www.lidberg.se/math/shapetransforms/wachspress.html

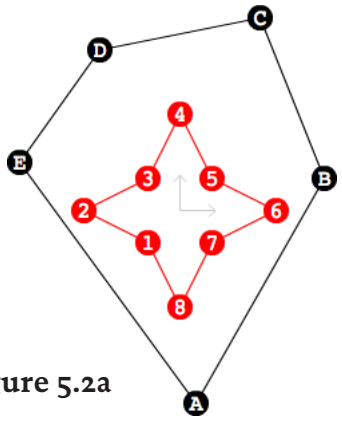


figure 5.2a

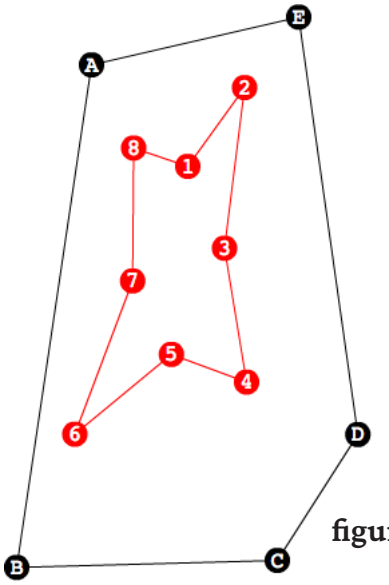


figure 5.2b

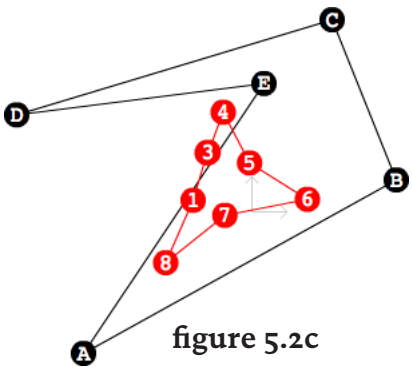


figure 5.2c

Final thoughts

A key advantage of using Barycentric or Wachspress coordinates is the binding process. The process of changing coordinate systems, from Cartesian coordinates to Wachspress or Barycentric coordinates, is called binding. Given the initial cage, each of it's interior points need to be computed, that is translated to the new coordinate system. The binding computation is normally computationally intense. However, after the binding process, much less computation power is needed for calculating the new locations/values of the interior points when the reference points are moved. This is because all interior points are expressed as linear combinations after the binding process. Linear combinations are, of course, calculated blazingly fast on a computer.

Another key advantage of Barycentric and Wachspress coordinates are their intrinsic property. The coordinates are defined relative to the vertex points and are independent of the surrounding Euclidean space. Coordinates with this property are also known as local coordinates. Local coordinates locate points relative to existing points, rather than to an origin. This property isolates different sets of points/shapes from each other so that they can be managed in a much simpler way.

These two key advantages, that applies for all generalized and classical Barycentric coordinates, together with the cage are of great practical use for different applications in computer graphics. In particular, generalizations of Barycentric coordinates are used for solving the shape and volume transformations that arise in computer animation.

In 2003, Michael S. Floater [16] introduced a generalization of Barycentric coordinates which he calls Mean value coordinates. Mean value coordinates takes their inspiration from Wachspress coordinates. Floater's coordinates marks the starting point of an explosion of new generalizations of Barycentric coordinates.

The famous animation studio Pixar (behind the acclaimed movies Toy Story and Finding Nemo) introduced another generalization at the annual graphic convention Siggraph in 2007 [17]. Their generalized Barycentric coordinates springs from the solutions of Laplace's equation. Since solutions to Laplace's equation are generally referred to as Harmonic functions, Pixar therefore call their coordinates: Harmonic coordinates. Pixar developed and used Harmonic coordinates to articulate their character Remy in the critically acclaimed movie Ratatouille. Using their Harmonic coordinates construction, a computer animator can completely articulate a character through deforming a cage. This is a big improvement compared to previous transformations methods.

I hope this paper have explained and implemented Barycentric and Wachspress coordinates in a good way.

Appendix

Euclid I.4
If two triangles have two sides equal to two sides respectively, and have the angles contained by the equal straight lines equal, then they also have the base equal to the base, the triangle equals the triangle, and the remaining angles equal the remaining angles respectively, namely those opposite the equal sides.

Euclid I.27
If a straight line falling on two straight lines makes the alternate angles equal to one another, then the straight lines are parallel to one another.

Euclid I.29
A straight line falling on parallel straight lines makes the alternate angles equal to one another, the exterior angle equal to the interior and opposite angle, and the sum of the interior angles on the same side equal to two right angles.

Euclid VI.2
If a straight line is drawn parallel to one of the sides of a triangle, then it cuts the sides of the triangle proportionally; and, if the sides of the triangle are cut proportionally, then the line joining the points of section is parallel to the remaining side of the triangle.

Euclid VI.4
In equiangular triangles the sides about the equal angles are proportional where the corresponding sides are opposite the equal angles.

Proof 2.4 details

$$\text{Area of CAP} = \frac{1}{2}\begin{vmatrix} x_C & y_C & 1 \\ x_A & y_A & 1 \\ x_P & y_P & 1 \end{vmatrix} = \frac{1}{2}[x_C y_A + x_P y_C + x_A y_P - x_P y_A - x_C y_P - x_A y_C] =$$

$$\frac{1}{2}[x_C y_A + (w_A x_A + w_B x_B + w_C x_C) y_C + x_A (w_A y_A + w_B y_B + w_C y_C)$$
$$- (w_A x_A + w_B x_B + w_C x_C) y_A - x_C (w_A y_A + w_B y_B + w_C y_C) - x_A y_C] =$$

$$\frac{1}{2}[x_C y_A + w_A x_A y_C + w_B x_B y_C + w_C x_C y_C + w_A x_A y_A + w_B x_A y_B + w_C x_A y_C$$
$$- w_A x_A y_A - w_B x_B y_A - w_C x_C y_A - w_A x_C y_A - w_B x_C y_B - w_C x_C y_C - x_A y_C] =$$

$$\frac{1}{2}[x_C y_A + w_A x_A y_C + w_B x_B y_C + w_C x_C y_C + w_A x_A y_A + w_B x_A y_B + w_C x_A y_C$$
$$- w_A x_A y_A - w_B x_B y_A - w_C x_C y_A - w_A x_C y_A - w_B x_C y_B - w_C x_C y_C - x_A y_C] =$$

$$\frac{1}{2}[x_C y_A - x_A y_C + w_A (x_A y_C - x_C y_A) + w_C (x_A y_C - x_C y_A)$$
$$+ w_B (x_B y_C + x_A y_B - x_B y_A - x_C y_B)] =$$

$$\frac{1}{2}[x_C y_A - x_A y_C + (w_A + w_C)(x_A y_C - x_C y_A) + w_B (x_B y_C + x_A y_B - x_B y_A - x_C y_B)] =$$

$$\frac{1}{2}[x_C y_A - x_A y_C + (1 - w_B)(x_A y_C - x_C y_A) + w_B (x_B y_C + x_A y_B - x_B y_A - x_C y_B)] =$$

$$\frac{1}{2}[x_C y_A - x_A y_C + x_A y_C - x_C y_A - w_B x_A y_C + w_B x_C y_A + w_B (x_B y_C + x_A y_B - x_B y_A - x_C y_B)] =$$

$$\frac{1}{2} w_B [x_A y_B + x_C y_A + x_B y_C - x_C y_B - x_A y_C - x_B y_A] = \frac{1}{2} w_B \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} =$$

$$w_B \cdot (\text{area of CAP}) \quad \longleftrightarrow \quad w_B = \frac{\text{area of CAP}}{\text{area of ABC}}$$

$$\text{Area of ABP} = \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_P & y_P & 1 \end{vmatrix} = \frac{1}{2}[x_A y_B + x_P y_A + x_B y_P - x_P y_B - x_A y_P - x_B y_A] =$$

$$\frac{1}{2}[x_A y_B + (w_A x_A + w_B x_B + w_C x_C) y_A + x_B (w_A y_A + w_B y_B + w_C y_C)$$
$$- (w_A x_A + w_B x_B + w_C x_C) y_B - x_A (w_A y_A + w_B y_B + w_C y_C) - x_B y_A] =$$

$$\frac{1}{2}[x_A y_B + w_A x_A y_A + w_B x_B y_A + w_C x_C y_A + w_A x_B y_A + w_B x_B y_B + w_C x_B y_C$$
$$- w_A x_A y_B - w_B x_B y_B - w_C x_C y_B - w_A x_A y_A - w_B x_A y_B + w_C x_A y_C - x_B y_A] =$$

$$\frac{1}{2}[x_A y_B - x_B y_A + w_A (x_B y_A - x_A y_B) + w_B (x_B y_A - x_A y_B) + w_C (x_C y_A + x_B y_C - x_A y_C - x_C y_B)] =$$

$$\frac{1}{2}[x_A y_B - x_B y_A + (w_A + w_B)(x_B y_A - x_A y_B) + w_C (x_C y_A + x_B y_C - x_A y_C - x_C y_B)] =$$

$$\frac{1}{2}[x_A y_B - x_B y_A + (1 - w_C)(x_B y_A - x_A y_B) + w_C (x_C y_A + x_B y_C - x_A y_C - x_C y_B)] =$$

$$\frac{1}{2}[x_A y_B - x_B y_A + x_B y_A - x_A y_B - w_C x_B y_A + w_C x_A y_B) + w_C (x_C y_A + x_B y_C - x_A y_C - x_C y_B)] =$$

$$\frac{1}{2} w_C [x_A y_B + x_C y_A + x_B y_C - x_C y_B - x_A y_C - x_B y_A] = \frac{1}{2} w_C \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} =$$

$$w_C \cdot (\text{area of ABC}) \quad \longleftrightarrow \quad w_C = \frac{\text{area of ABP}}{\text{area of ABC}}$$

References

[1] Möbius, August. Der barycentrische Calcül, ein neues Hülfs- mittel zur analytischen Behandlung der Geometrie, Verlag von Johann Ambrosia Barth, Leipzig, 1827

[2] Stein, Sherman. Archimedes - What did he do besides cry eureka, The mathematical association of America, page 3, Washington, 1999

[3] Strathern, Paul. Archimedes & the Fulcrum – the big idea, Arrow Books, page 23, 2010

[4] http://aleph0.clarku.edu/~djoyce/java/elements/toc.html

[5] Archimedes. On the Equilibrium of Planes, from " The works of Archimedes", Cambridge University Press, page 189-220, T.L. Heath, 2002

[6] Archimedes. On the Equilibrium of Planes, from " The works of Archimedes", Cambridge University Press, page 189-220, T.L. Heath, 2002

[7] Brannan, David A, Esplen Matthew F & Gray, Jeremy J. Geometry, Cambridge University Press, page 75, Cambridge, 2004

[8] Stein, Sherman. Archimedes - What did he do besides cry eureka, The mathematical association of America, page 15, Washington, 1999

[9] Strathern, Paul. Archimedes & the Fulcrum – the big idea, Arrow Books, page 23, 2010

[10] Möbius, August. Der barycentrische Calcül, ein neues Hülfs- mittel zur analytischen Behandlung der Geometrie, Verlag von Johann Ambrosia Barth, Leipzig, 1827

[11] Råde, Lennart & Westergren, Bertil. Mathematics Handbook for science and engineering, Studentlitteratur, page 79, Lund, Fourth edition, 1998

[12] Gouraud, Henri. Continuous shading of curved surfaces, IEEE transactions on computers, vol. c-20, no. 6, juni 1971

[13] Wachspress, Eugene. A Rational Finite Element Basis, Academic Press, New York, 1975

[14] Meyer, M, Lee, H, Barr A. H & Desbrun, M. Generalized barycentric coordinates for irregular polygons. Journal of Graphics Tools, 7(1):13–22, 2002

[15] Råde, Lennart & Westergren, Bertil. Mathematics Handbook for science and engineering, Studentlitteratur, page 67, Lund, Fourth edition, 1998

[16] Floater, M. S. Mean value coordinates. Computer Aided Geometric Design, 20(1):19– 27, 2003

[17] http://graphics.pixar.com/library/HarmonicCoordinatesB/

Special thanks

Gustaf Halldin
Magnus Marin

Source code for shape transformations

The implementations are written in Actionscript 3, a script language used in Adobe Flash. The source code is also available on my web server: http://www.lidberg.se/math/shapetransforms/shape_transforms.zip

```
// Barycentric coordinates for shape transformations
package se.lidberg{
   public class Barycentric extends ShapeTransform {

      public function Barycentric() {}
      override protected function createReferencePoints():void {
         _referencePoints.push(new ReferencePoint("A"));
         _referencePoints[0].xPos = -18;
         _referencePoints[0].yPos = +6;
         _canvas.addChild(_referencePoints[0]);

         _referencePoints.push(new ReferencePoint("B"));
         _referencePoints[1].xPos = 2;
         _referencePoints[1].yPos = -10;
         _canvas.addChild(_referencePoints[1]);

         _referencePoints.push(new ReferencePoint("C"));
         _referencePoints[2].xPos = 15;
         _referencePoints[2].yPos = 10;
         _canvas.addChild(_referencePoints[2]);
      }
      override protected function calculateWeightListForPoint(thisPoint:Point):Array {
         var weightsList:Array = [];

         weightsList[0] = Math.abs(signedArea(new Point(_referencePoints[1].xPos,
         _referencePoints[1].yPos) ,new Point(_referencePoints[2].xPos,
         _referencePoints[2].yPos),thisPoint));

         weightsList[1] = Math.abs(signedArea(new Point(_referencePoints[0].xPos,
         _referencePoints[0].yPos) ,new Point(_referencePoints[2].xPos,
         _referencePoints[2].yPos),thisPoint));

         weightsList[2] = Math.abs(signedArea(new Point(_referencePoints[0].xPos,
         _referencePoints[0].yPos) ,new Point(_referencePoints[1].xPos,
         _referencePoints[1].yPos),thisPoint));

         // Normalize
         var weightsSum:Number = weightsList[0] + weightsList[1] + weightsList[2];
         for (var j:uint=0; j<weightsList.length; j++) {
            weightsList[j] = weightsList[j] / weightsSum;
         }

         return (weightsList);
      }
   }
}

// Wachspress coordinates for shape transformations
package se.lidberg{
   public class Wachspress extends ShapeTransform {

      public function Wachspress() {}
      override protected function createReferencePoints():void {
         _referencePoints.push(new ReferencePoint("A"));
         _referencePoints[0].xPos = 1;
         _referencePoints[0].yPos = -12;
         _canvas.addChild(_referencePoints[0]);

         _referencePoints.push(new ReferencePoint("B"));
         _referencePoints[1].xPos = 9;
         _referencePoints[1].yPos = 2;
         _canvas.addChild(_referencePoints[1]);

         _referencePoints.push(new ReferencePoint("C"));
         _referencePoints[2].xPos = 5;
         _referencePoints[2].yPos = 12;
         _canvas.addChild(_referencePoints[2]);

         _referencePoints.push(new ReferencePoint("D"));
         _referencePoints[3].xPos = -5;
         _referencePoints[3].yPos = 10;
         _canvas.addChild(_referencePoints[3]);

         _referencePoints.push(new ReferencePoint("E"));
         _referencePoints[4].xPos = -10;
         _referencePoints[4].yPos = 3;
         _canvas.addChild(_referencePoints[4]);
      }
```

```
      override protected function calculateWeightListForPoint(thisPoint:Point):Array {
         var weightsList:Array = [];
         var weightSum:Number = 0;
         for (var i:uint=0; i<_referencePoints.length; i++) {
            var prevIndex:int = i - 1;
            if (prevIndex < 0) {
               prevIndex = _referencePoints.length - 1;
            }
            var nextIndex:int = i + 1;
            if (nextIndex > _referencePoints.length - 1) {
               nextIndex = 0;
            }

            var prevRefPoint:Point = new Point(_referencePoints[prevIndex].xPos,
            _referencePoints[prevIndex].yPos);

            var currentRefPoint:Point = new Point(_referencePoints[i].xPos,
            _referencePoints[i].yPos);

            var nextRefPoint:Point = new Point(_referencePoints[nextIndex].xPos,
            _referencePoints[nextIndex].yPos);

            var thisWeight:Number = signedArea(prevRefPoint,currentRefPoint,nextRefPoint)/
            (signedArea(prevRefPoint,currentRefPoint,thisPoint)*signedArea(currentRefPoint,
            nextRefPoint,thisPoint));

            weightSum = weightSum + thisWeight;
            weightsList.push(thisWeight);
         }

         // Normalize
         for (var j:uint=0; j<weightsList.length; j++) {
            weightsList[j] = weightsList[j] / weightSum;
         }

         return (weightsList);
      }

   }
}

package se.lidberg{
   import flash.display.*;
   import flash.events.*;
   import flash.text.*;
   import flash.geom.*;

   public class ShapeTransform extends Sprite {
      protected var _referencePoints:Array = [];
      protected var _innerPoints:Array = [];
      protected var _weights:Array = [];
      protected var _canvas:Canvas = new Canvas  ;
      protected var _currentValue:TextField;

      public function ShapeTransform() {
         _canvas.x = stage.stageWidth / 2;
         _canvas.y = stage.stageHeight / 2;
         addChild(_canvas);

         _currentValue = Utils.createATextField();
         _currentValue.x = 10;
         _currentValue.y = 5;
         addChild(_currentValue);

         createInnerPoints();
         createReferencePoints();
         setUpMouseEvents();
         binding();
         renderCanvas();
      }
      protected function createInnerPoints():void {
         _innerPoints.push(new InnerPoint("1"));
         _innerPoints[0].xPos = -2;
         _innerPoints[0].yPos = -2;
         _canvas.addChild(_innerPoints[0]);

         _innerPoints.push(new InnerPoint("2"));
         _innerPoints[1].xPos = -6;
         _innerPoints[1].yPos = 0;
         _canvas.addChild(_innerPoints[1]);

         _innerPoints.push(new InnerPoint("3"));
         _innerPoints[2].xPos = -2;
         _innerPoints[2].yPos = 2;
         _canvas.addChild(_innerPoints[2]);

         _innerPoints.push(new InnerPoint("4"));
         _innerPoints[3].xPos = 0;
         _innerPoints[3].yPos = 6;
         _canvas.addChild(_innerPoints[3]);
```

```
            _innerPoints.push(new InnerPoint("5"));
            _innerPoints[4].xPos = 2;
            _innerPoints[4].yPos = 2;
            _canvas.addChild(_innerPoints[4]);

            _innerPoints.push(new InnerPoint("6"));
            _innerPoints[5].xPos = 6;
            _innerPoints[5].yPos = 0;
            _canvas.addChild(_innerPoints[5]);

            _innerPoints.push(new InnerPoint("7"));
            _innerPoints[6].xPos = 2;
            _innerPoints[6].yPos = -2;
            _canvas.addChild(_innerPoints[6]);

            _innerPoints.push(new InnerPoint("8"));
            _innerPoints[7].xPos = 0;
            _innerPoints[7].yPos = -6;
            _canvas.addChild(_innerPoints[7]);
        }
        // Gets overwritten:
        protected function createReferencePoints():void {
        }
        // Gets overwritten:
        protected function calculateWeightListForPoint(thisPoint:Point):Array {
            var weightsList:Array = [];
            return (weightsList);
        }
        protected function setUpMouseEvents():void {
            for (var i:uint=0; i<_referencePoints.length; i++) {
                _referencePoints[i].addEventListener("onChangedPosition", reCalculatePositions);
                _referencePoints[i].addEventListener("onMouseOver", showPosition);
            }
            for (var j:uint=0; j<_innerPoints.length; j++) {
                _innerPoints[j].addEventListener("onMouseOver", showPosition);
            }
        }
        protected function binding():void {
            for (var i:uint=0; i<_innerPoints.length; i++) {
                _weights[i] = calculateWeightListForPoint(new Point(_innerPoints[i].xPos,
                _innerPoints[i].yPos));
            }
        }
        protected function showPosition(e:Event):void {
            _currentValue.text = "x="+ e.target.xPos + ", y="+ e.target.yPos+"\n\n";
            for (var i:uint=0; i<_weights.length; i++) {
                if (e.target == _innerPoints[i]) {
                    for (var j:uint=0; j<_weights[i].length; j++) {
                        _currentValue.appendText("W(" + _referencePoints[j].nameOfPoint + ")="
                        + _weights[i][j] + "\n");
                    }
                }
            }
        }
        protected function reCalculatePositions(e:Event):void {
            for (var i:uint=0; i<_innerPoints.length; i++) {
                _innerPoints[i].xPos = 0;
                _innerPoints[i].yPos = 0;
                var thisCoordinates:Array = _weights[i];
                for (var j:uint=0; j<_referencePoints.length; j++) {
                    _innerPoints[i].xPos +=  thisCoordinates[j] * _referencePoints[j].xPos;
                    _innerPoints[i].yPos +=  thisCoordinates[j] * _referencePoints[j].yPos;
                }
            }
            showPosition(e);
            renderCanvas();
        }
        protected function renderCanvas():void {
            _canvas.render(_referencePoints, _innerPoints);
        }
        protected function signedArea(v1:Point,v2:Point,v3:Point):Number {
            var x1:Number = v1.x;
            var y1:Number = v1.y;
            var x2:Number = v2.x;
            var y2:Number = v2.y;
            var x3:Number = v3.x;
            var y3:Number = v3.y;

            var thisArea:Number = x1 * y2 + y1 * x3 + x2 * y3 - y2 * x3 - x1 * y3 - y1 * x2;
            thisArea = (thisArea /2);

            // Avoid returning zero area
            if (thisArea == 0) {
                thisArea = 0.0000000001;
            }

            return thisArea;
        }
    }
}
```

```
package se.lidberg{
   import flash.display.*;

   public class Canvas extends Sprite {
      private const AXIS_LENGTH:Number = 22;
      private const ARROW_LENGTH:Number = 4;
      private var _linesCanvas:Sprite = new Sprite  ;
      private var _lines:Shape = new Shape  ;

      public function Canvas() {
         scaleX = scaleY = 1.2;

         _linesCanvas.addChild(_lines);
         addChild(_linesCanvas);

         // Draw axis
         graphics.lineStyle(1,0xcccccc);

         graphics.moveTo(0,0);
         graphics.lineTo(AXIS_LENGTH,0);
         graphics.lineTo(AXIS_LENGTH - ARROW_LENGTH, -  ARROW_LENGTH);
         graphics.moveTo(AXIS_LENGTH,0);
         graphics.lineTo(AXIS_LENGTH - ARROW_LENGTH,ARROW_LENGTH);

         graphics.moveTo(0,0);
         graphics.lineTo(0, -  AXIS_LENGTH);
         graphics.lineTo(ARROW_LENGTH, -1*(  AXIS_LENGTH - ARROW_LENGTH));
         graphics.moveTo(0, -  AXIS_LENGTH);
         graphics.lineTo( -  ARROW_LENGTH, -1*(AXIS_LENGTH - ARROW_LENGTH));
      }
      public function render(referencePoints:Array, innerPoints:Array):void {
         _linesCanvas.removeChild(_lines);
         _lines = new Shape();
         _lines.graphics.lineStyle(1,0x000000);
         _lines.graphics.moveTo(referencePoints[0].x,referencePoints[0].y);
         for (var i:uint = 1; i < referencePoints.length; i++) {
            _lines.graphics.lineTo(referencePoints[i].x,referencePoints[i].y);
         }
         _lines.graphics.lineTo(referencePoints[0].x,referencePoints[0].y);

         _lines.graphics.lineStyle(1,0xff0000);
         _lines.graphics.moveTo(innerPoints[0].x,innerPoints[0].y);
         for (var j:uint = 1; j < innerPoints.length; j++) {
            _lines.graphics.lineTo(innerPoints[j].x,innerPoints[j].y);
         }
         _lines.graphics.lineTo(innerPoints[0].x,innerPoints[0].y);
         _linesCanvas.addChild(_lines);
      }
   }
}

package se.lidberg{
   import flash.display.*;
   import flash.events.*;

   public class InnerPoint extends Sprite {
      private const SCALE_FACTOR:Number = 10;

      public function InnerPoint(thisLetter = "X") {
         var circleShape:Shape = new Shape();
         circleShape.graphics.beginFill(0xff0000);
         circleShape.graphics.drawCircle(0,0,8);
         circleShape.graphics.endFill();
         addChild(circleShape);

         var textSprite:Sprite = Utils.createTextSprite(thisLetter);
         textSprite.x = -textSprite.width/2;
         textSprite.y = textSprite.height/2-3;
         addChild(textSprite);

         addEventListener(MouseEvent.MOUSE_OVER,mouseOverHandler);
         addEventListener(MouseEvent.MOUSE_OUT,mouseOutHandler);
      }
      public function set xPos(thisX:Number):void {
         x = thisX*SCALE_FACTOR;
      }
      public function set yPos(thisY:Number):void {
         y = -thisY*SCALE_FACTOR;
      }
      public function get xPos():Number {
         return(x/SCALE_FACTOR);
      }
      public function get yPos():Number {
         return(-y/SCALE_FACTOR);
      }
      private function mouseOverHandler(e:MouseEvent):void {
         scaleX = 1.2;
         scaleY = 1.2;
         dispatchEvent(new Event("onMouseOver"));
      }
```

```
      private function mouseOutHandler(e:MouseEvent):void {
         scaleX = 1;
         scaleY = 1;
         dispatchEvent(new Event("onMouseOut"));
      }
   }
}

package se.lidberg{
   import flash.display.*;
   import flash.events.*;

   public class ReferencePoint extends Sprite {
      private const SCALE_FACTOR:Number = 10;
      private var _nameOfPoint:String;

      public function ReferencePoint(thisLetter:String = "A") {
         var circleShape:Shape = new Shape();
         circleShape.graphics.beginFill(0x000000);
         circleShape.graphics.drawCircle(0,0,8);
         circleShape.graphics.endFill();
         addChild(circleShape);

         var textSprite:Sprite = Utils.createTextSprite(thisLetter);
         textSprite.x = -textSprite.width/2;
         textSprite.y = textSprite.height/2-3;
         addChild(textSprite);

         addEventListener(MouseEvent.MOUSE_DOWN,mouseDownHandler);
         addEventListener(MouseEvent.MOUSE_UP,mouseUpHandler);
         addEventListener(MouseEvent.MOUSE_OVER,mouseOverHandler);
         addEventListener(MouseEvent.MOUSE_OUT,mouseOutHandler);
         buttonMode = true;
         mouseChildren = false;

         nameOfPoint = thisLetter;
      }
      public function set nameOfPoint(thisName:String):void {
         _nameOfPoint = thisName;
      }
      public function get nameOfPoint():String {
         return(_nameOfPoint);
      }
      public function set xPos(thisX:Number):void {
         x = thisX*SCALE_FACTOR;
      }
      public function set yPos(thisY:Number):void {
         y = -thisY*SCALE_FACTOR;
      }
      public function get xPos():Number {
         return(x/SCALE_FACTOR);
      }
      public function get yPos():Number {
         return(-y/SCALE_FACTOR);
      }
      private function mouseOverHandler(e:MouseEvent):void {
         scaleX = 1.2;
         scaleY = 1.2;
         dispatchEvent(new Event("onMouseOver"));
      }
      private function mouseOutHandler(e:MouseEvent):void {
         scaleX = 1;
         scaleY = 1;
         dispatchEvent(new Event("onMouseOut"));
      }
      private function mouseDownHandler(e:MouseEvent):void {
         startDrag(false);
         addEventListener(MouseEvent.MOUSE_MOVE, mouseMoveHandler);
      }
      private function mouseMoveHandler(e:MouseEvent):void {
         dispatchEvent(new Event("onChangedPosition"));
      }
      private function mouseUpHandler(e:MouseEvent):void {
         removeEventListener(MouseEvent.MOUSE_MOVE, mouseMoveHandler);
         stopDrag();
         dispatchEvent(new Event("onChangedPosition"));
      }
   }
}
```

```
package se.lidberg{
   import flash.display.*;
   import flash.text.engine.*;
   import flash.text.*;

   public class Utils extends Sprite {

      public function Utils() {

      }
      public static function createTextSprite(thisText:String = “X”, thisWidth:Number = 50):Sprite {

         // Font & FontLookup
         var thisFontDescription:FontDescription=new FontDescription();
         thisFontDescription.fontLookup=FontLookup.DEVICE;
         thisFontDescription.fontName=”_typewriter”;

         // FontSize & FontColor
         var thisElementFormat:ElementFormat=new ElementFormat(thisFontDescription);
         thisElementFormat.fontSize=14;
         thisElementFormat.color=0xffffff;

         // TextElement, TextBlock & Content
         var thisTextElement:TextElement=new TextElement(thisText,thisElementFormat);
         var thisTextBlock:TextBlock = new TextBlock();
         thisTextBlock.content=thisTextElement;

         // Sprite & TextLine
         var thisTextLine:TextLine=thisTextBlock.createTextLine(null,thisWidth);
         var thisSprite:Sprite = new Sprite();
         thisSprite.addChild(thisTextLine);

         return (thisSprite);
      }
      public static function createATextField():TextField {
         var thisTextField = new TextField();
         thisTextField.condenseWhite = true;
         thisTextField.selectable = true;
         thisTextField.multiline = true;
         thisTextField.width = 250;

         var thisFormat:TextFormat = new TextFormat();
         thisFormat.color = 0x000000;
            thisFormat.size = 11;
         thisFormat.font = “_typewriter”;
             thisTextField.defaultTextFormat = thisFormat;

         return(thisTextField);
      }
   }
}
```