# The MorphoxX Project

March 2009.

Jean-Baptiste Fiot
Ecole Centrale Paris
Ecole Normale Supérieure de Cachan
jean-baptiste.fiot@student.ecp.fr

## Abstract

*The MorphoxX Project studies various kinds of morphing methods based on cages. Cages are a simple way to deform and animate objects. Basically, objects are surrounded by cages, the cages are deformed by artists, and the deformed objects are computed.*

*To deform objects via cages, one should introduce a coordinate system. Several choices are available, and this project aims to implement and compare Mean Value Coordinates, Harmonic Coordinates, and Green Coordinates.*

*Finally, my source code is available (for free!) at [8].*

## 1. Introduction

In the industries of animation movies and video games, 3D characters are modeled by artists using modeling tools such as 3D Studio Max, Blender, … When it comes to animation, various techniques are possible, such as the well known motion capture. Realistic, accurate, and rapid, this technique however requires costly and specific hardware equipments, and cannot be used to simulate animals, monsters, etc. In this case, the meshes of the animated objects have to be deformed in another way to create animations.

Obviously, nobody wants to pay artists to deform millions of vertices per picture! (and artists probably do not want to go though this boring task...). Here comes the idea of cages. As explained in the abstract, the basic idea is to surround an object, deform the cage – clearly quicker than deforming the object – , and let the computer do the rest! Sounds cool?

Various coordinate systems can be used for cage manipulation. This paper studies Mean Value Coordinates, Harmonic Coordinates, and Green Coordinates, respectively in sections 2, 3 and 4. Section 5 concludes this paper with a comparative statement of these methods.

## 2. Mean Value Coordinates

### 2.1. Definition

As explained in [1] and [2], Mean Value Coordinates are a generalization of barycentric coordinates to n-sided polygons or even sets of polygons.

Let's see how to compute these coordinates at a given point v.

Let $\{v_i\}$ be the vertices of the cage.

One would like to have smooth *homogeneous barycentric coordinates*

$$\{w_i : \mathbb{R}^2 \to \mathbb{R}, i \in [\![1,n]\!]\} \text{ so that}$$
$$\sum_{i=1}^{n} w_i(v)(v_i - v) = 0 \Leftrightarrow v = \frac{\sum_{i=1}^{n} w_i(v)\, v_i}{\sum_{i=1}^{n} w_i(v)}$$

The associated *normalized barycentric coordinates* are:

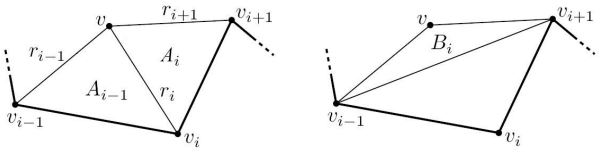
$$\forall i \in [\![1,n]\!], \lambda_i(v) = \frac{w_i(v)}{\sum_{i=1}^{n} w_i(v)}$$

Any point v could therefore be written as an affine combination of $v_1, \ldots, v_n$ with weights $\lambda_1(v), \ldots, \lambda_n(v)$.

These coordinates should also satisfy the *Lagrange property*:

$$\forall i \in [\![1,n]\!], \lambda_i(v_j) = \delta_{i,j} = \begin{cases} 1 & if\ i = j \\ 0 & otherwise \end{cases}$$

Let's get back to our cage, and define notations:




Let's define

$$R_i(v) = \|v - v_i\|$$

Let Ai be the signed area of the triangle [v, $v_i$, $v_{i+1}$] at the point v. Adding a 3rd coordinate equal to zero, this area is found using the 3rd coordinate of the vector cross product divided by 2.

$$A_i(v) = \frac{\left\|\overrightarrow{vv_i} \wedge \overrightarrow{vv_{i+1}}\right\|}{2} = \frac{\left|\overrightarrow{vv_i} \wedge \overrightarrow{vv_{i+1}}\right|_3}{2}$$

Let Bi be the signed area of the triangle [v, $v_{i-1}$, $v_{i+1}$] at the point v

$$B_i(v) = \frac{\left\|\overrightarrow{vv_{i-1}} \wedge \overrightarrow{vv_{i+1}}\right\|}{2} = \frac{\left|\overrightarrow{vv_{i-1}} \wedge \overrightarrow{vv_{i+1}}\right|_3}{2}$$

One can demonstrate that $A_i(v)$, $-B_i(v)$ and $A_{i-1}(v)$ are homogeneous barycentric of v with respect to the triangle [$v_{i-1}$, $v_i$, $v_{i+1}$]

Combining these equations (please refer to [2] for more details), we can define:

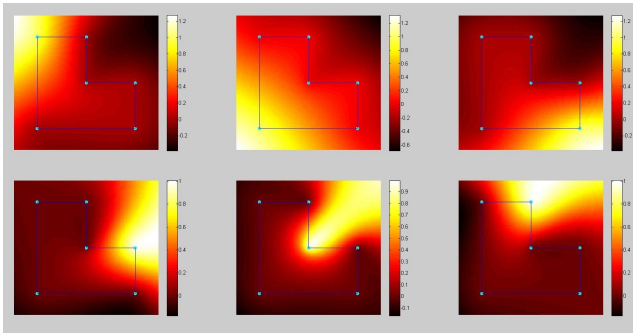
$$w_i = \frac{r_{i-1} A_i - r_i B_i + r_{i+1} A_{i-1}}{A_{i-1} A_i}$$

## 2.2. Properties

As explained in [1], these coordinates have interesting properties such as:

- interpolation
- smoothness
- linear precision

## 2.3. Examples



## 2.4. Object deformation

### *2.4.1 Principle*

As explained before, deforming objects via cages is simple, we draw a cage, we deform it, and we let the computer deform the object... But I bet you wonder what mysterious operations the computer does!

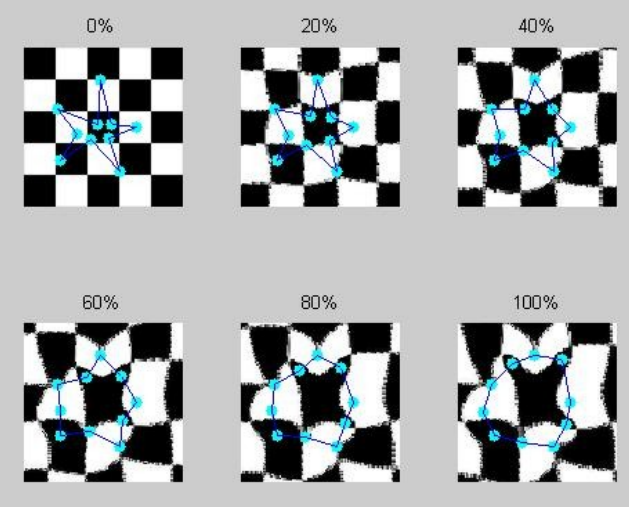
To create a deformed object, here is what we do:

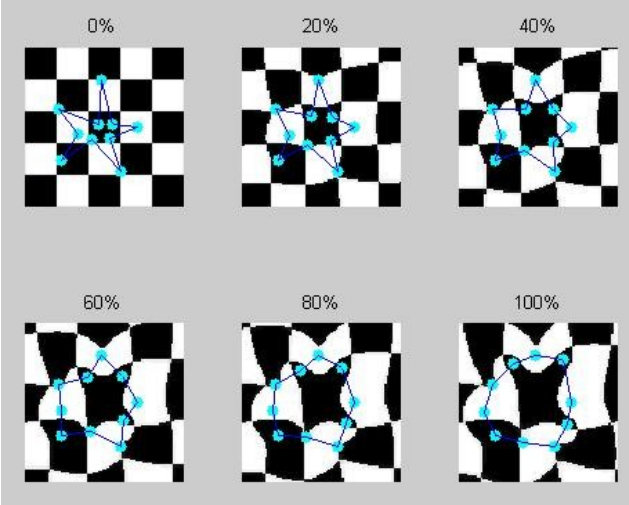
- For each pixel of the **final** picture, we compute the coordinates relatively to the **final** cage.
- We find where a point with such coordinates would be in the original image, with the original cage.
- We interpolate the colors of the neighbor pixels to get the color of this point.
- We color the pixel of the final image with this interpolated color.

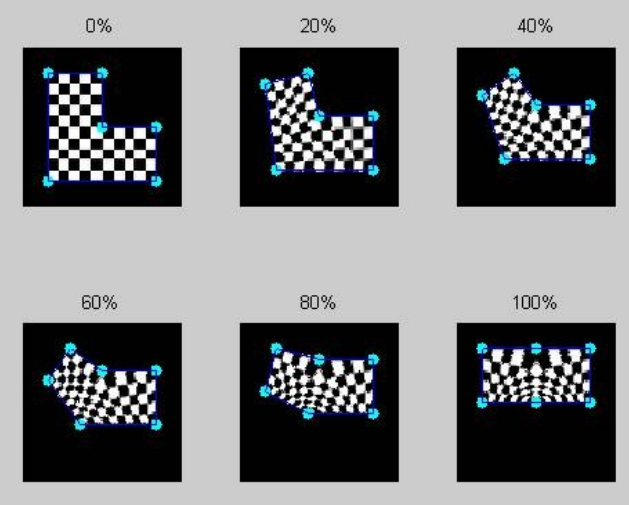
Well, that's really simple, the only thing to understand is that the loop is done on the final image, and not on the original. This is a well-know trick in Computer Vision.

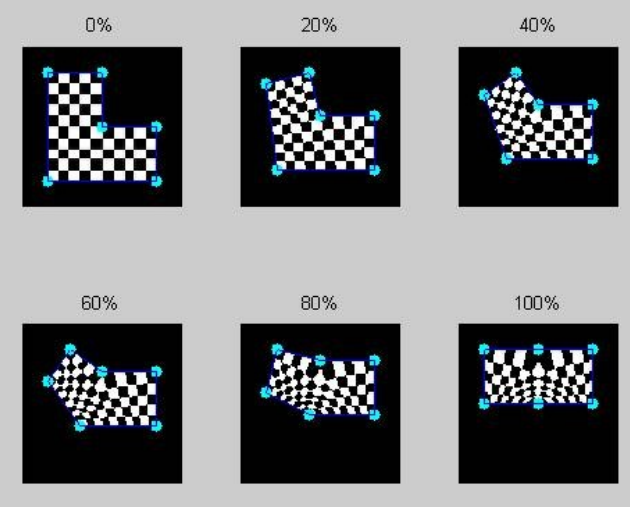
### *2.4.2 Results*

Here are some examples. In the first one, a star is progressively deformed to a circle. On the second one, an L shape is deformed to a rectangle. Finally, on the Sonic set, his right part is stretched even further to the right.




On these B&W pictures, we actually get better results by using the **color of the nearest pixel instead of interpolating** – see next column –. Indeed, instead of getting gray pixels dribbling on the edges, it gives sharp edges. For full-colored pictures like in the Sonic set, the difference between interpolated colors and nearest-pixel colors are insignificant (visually speaking).

And here is the result for the Sonic set:

## 3. Harmonic Coordinates

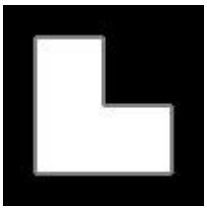
### 3.1. Definition

As Mean Value Coordinates, Harmonic Coordinates are **generalization of barycentric coordinates**, and give a way to write any point as a weighted combination of the vertices. Well, not any point to be honest: HC are defined only on the interior of the cage.

Harmonic Coordinates are all solution to the **Laplace's equation**:

$$\forall i \in [\![1,n]\!],\ \forall v \in \mathring{C},\ \Delta w_i(v) = 0$$

To obtain them, we first create a **mask**, with "INTERIOR", "EXTERIOR" and "BOUNDARY" labels.



To get this mask, I first used the "inpoly" matlab function, and then implemented an exhaustive search of the boundary points following the rule:

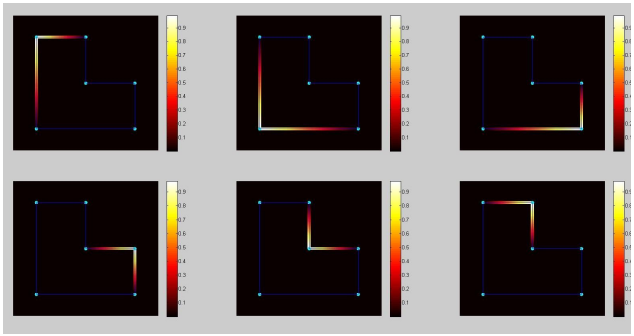
$$\left.\begin{array}{c} \|v - v_i\| < \|v_i - v_{i+1}\| \\ \|v - v_{i+1}\| < \|v_i - v_{i+1}\| \\ d(v, L) = \dfrac{\overrightarrow{v_i v_{i+1}} \wedge \overrightarrow{v_i v}}{\|v_i - v_{i+1}\|} \leq 0.5 \end{array}\right\} \Rightarrow v \in Boundary$$

In the 3rd condition, L is the line ($v_i$ $v_{i+1}$). This condition imposes the distance from the point v to L to be less or equal than 0.5 pixel. It can be found in a mundane way using 2 different expressions of the area of the triangle [v $v_i$ $v_{i+1}$].

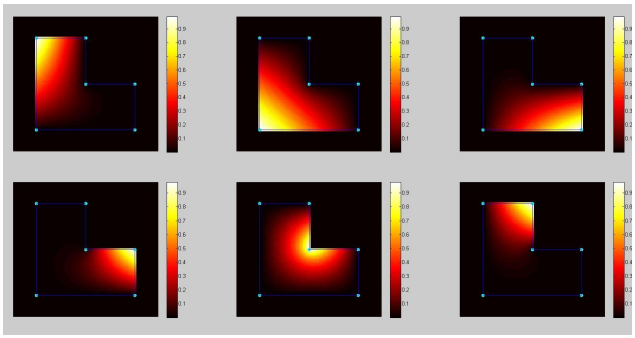
This mask creation is not optimal, but as long as this step is not the most computationally expensive, I have not implemented a better solution.

To implement a mask creation in another language, one should google "**polygon scan converting**". Nice illustrations are available in [7] on this topic.

Once the mask is created, **we initialize the values for the boundary points, so that the interpolation property is satisfied** (coordinate #i equal to 1 on vertex #i, and linear decrease to 0 up to the vertices i-1 and i+1).



Finally, **values are propagated on the interior using 4- connexity averaging**. (At each iteration, values of the interior points are replaced by the averages of their 4 neighbors).



### 3.2. Properties

Harmonic Coordinates have various interesting properties:

- Interpolation
- Smoothness
- Non-negativity
- Interior locality
- Linear Reproduction
- Affine Invariance
- Struc generalization of barycentric coordinates

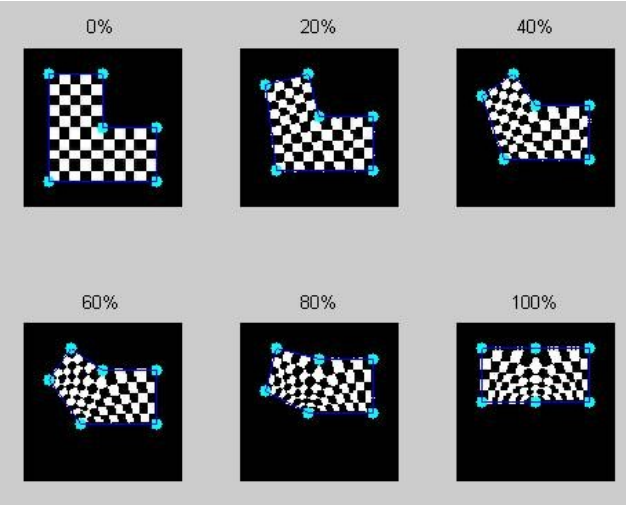
Please refer to [4] and [5] for details.

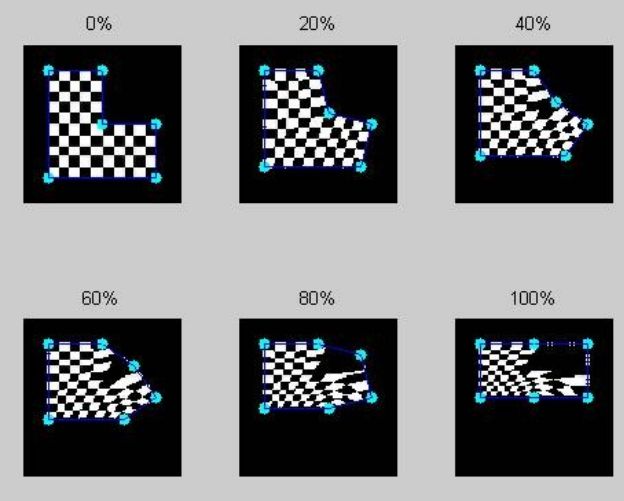
### 3.3. Object deformation

*3.3.1 Principle*

Objects are deformed using the same method as for MV coordinates (see section 2.4).

*3.3.2 Results on the "Lx Checkerboard" sets*

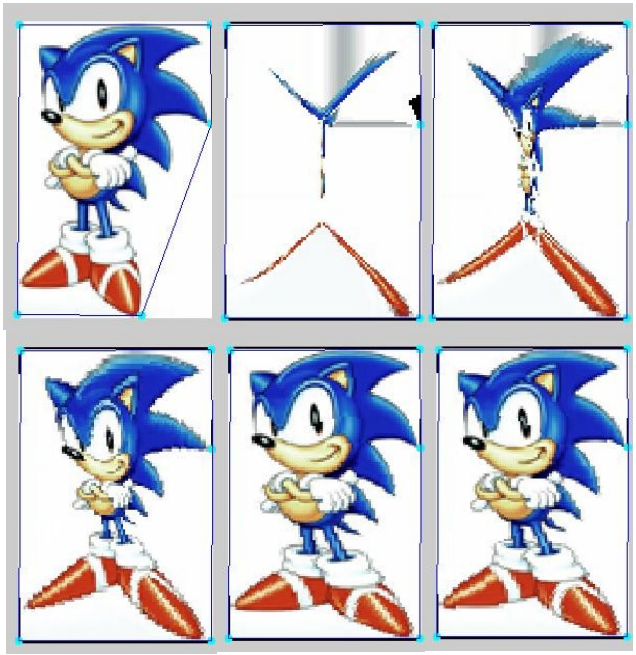



*3.3.3 Influence of the number of iterations.*

As the Harmonic Coordinates are not exact results, but obtained though an iterative process, we obviously wonder how many iterations should be used to get satisfying results.

Here is a series of tests on the Light Sonic test (100*100 picture, and 5 vertices). The first picture is the original shape, and the following are the computed deformed pictures with 100, 500, 2000, 5000, and 20000 iterations in HC computation.



Obviously, the larger is the cage (in terms of pixels), the higher the number of iterations should be (to let time for the propagation to go deep enough the cage). If the number of iterations is not high enough, the vertices do not have enough time to spread their influences, and the character is compressed towards the middle of the cage.

*3.3.4 Large scale computation*

I tried one large scale computation on the Sonic set (300*457 picture, 29 vertices, 25000 iterations for the computation of the coordinates).

After 21 long hours on my Dell XPS Gen II laptop (with a Pentium M 2.13GHz), a lil Sonic was born!

*Lil Sonic is finally born. Isn't he cute?*

## 4. Green Coordinates

### 4.1. Definition

Green Coordinates differ to Mean Value Coordinates and Harmonic Coordinates by having **new terms corresponding to face elements**.

Their goal is to also respect the cage's faces' orientations.

A point v could be written as:

$$v = \sum_{vertices} \phi_i(v) v_i + \sum_{edges} \psi_i(v) n(t_i)$$

and the corresponding moved point as:

$$v' = \sum_{vertices} \phi_i(v) v_i' + \sum_{edges} \psi_i(v) s_i n(t_i')$$

$$with \begin{cases} s_i = \frac{\|t_i'\|}{\|t_i\|} \left( = \frac{\|edge_i'\|}{\|edge_i\|} \text{ in 2D} \right) \\ n(t_i) \text{ outward normal of the edge \# } i \end{cases}$$

Pseudo code is given in [6] for the computation of the $\Phi_i$ and $\Psi_i$ coordinates.

Here is how to compute the outward normal of an edge e=(P1,P2):

- first we get **one** normal using:

$$\vec{n} = \begin{pmatrix} P1_2 - P2_2 \\ P2_1 - P1_2 \end{pmatrix}$$

- then we check if the normal is in the right direction, by considering the point at a distance of 2 pixels from the medium of the edge, and in direction of the previous normal. We **inverse the normal if need be:**

$$if \left( \frac{P1+P2}{2} + 2\vec{n} \right) \in C \text{ then } \vec{n} \leftarrow -\vec{n}$$

The normalization is however tricky! Indeed the previous formulas mix up affine and vectorial spaces. In order to have results independent to the origin (ie translation invariant), we must normalize using:

$$\sum_{vertices} \phi_i(v) = 1$$

$$\left( \text{and not } \sum_{vertices} \phi_i(v) + \sum_{edges} \psi_i(v) = 1 \, !! \right)$$

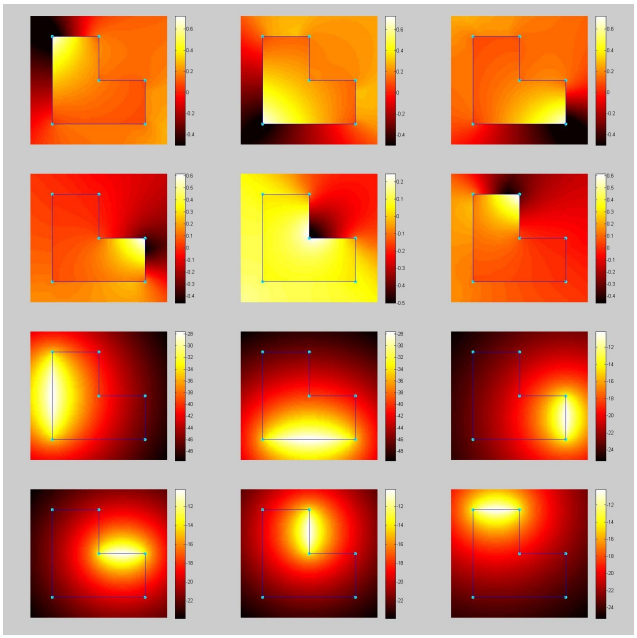
### 4.2. Properties

The Green coordinates have these additional properties:

- linear reproduction
- translation invariance
- rotation and scale invariance
- shape preservation
- smoothness

### 4.3. Examples

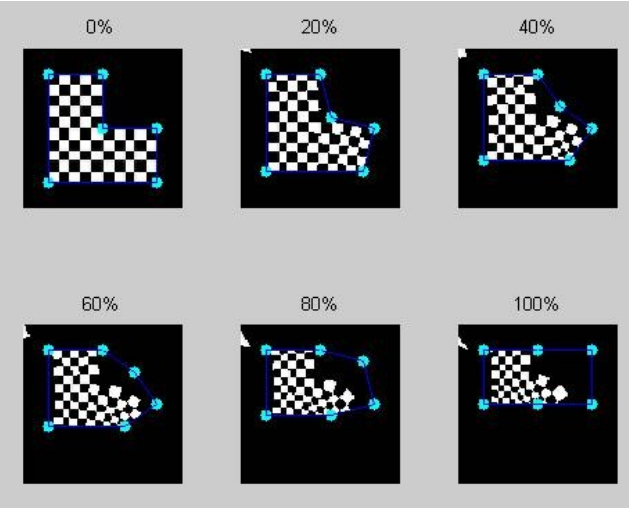
The 6 first sub-pictures correspond to the $\Phi_i$ coordinates, and the last 6 correspond to the $\Psi_i$ coordinates

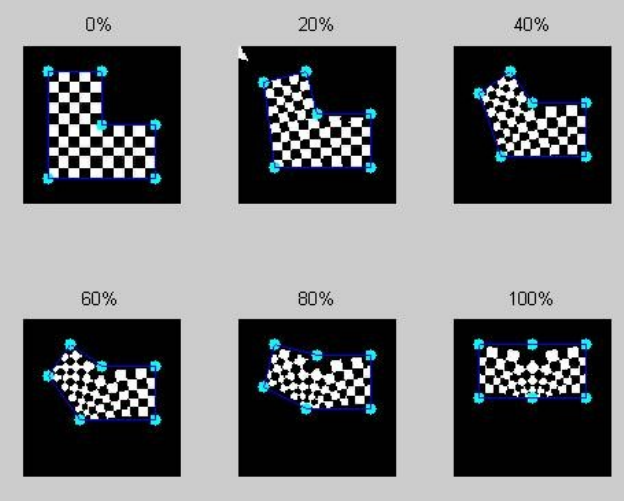


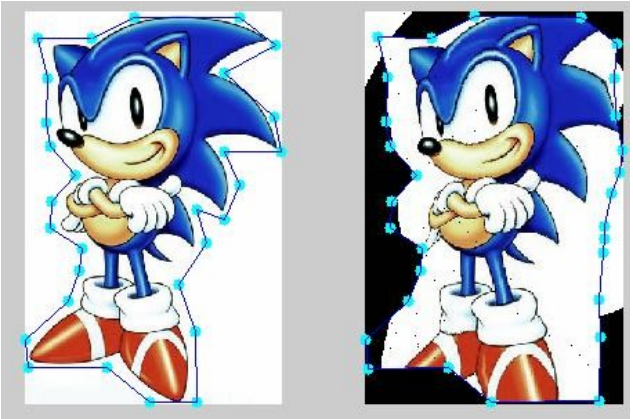
### 4.4. Extension to the cage's exterior

An extension in presented in [6] so that the $\Phi_i$ are smooth outside the cage. This is done by adding extra terms in the computation. A typical application is the deformation with partial cage. I have not implemented this extension.

### 4.5. Object deformation



The deformation process is pretty quick, it took about 25 minutes to deform the Sonic set.

Note that the Green Coordinates allow **flexibility**. The model can move away from the edges or step out of the cage. It generally gives better visual results, but can however be unwanted. (For instance in a video game, we do not want a character next to a wall to penetrate into it!).

As explained in section 4.4, the part of the Sonic model outside the cage is not displayed because I have not implemented the extension for the cage's exterior.

## 5. Conclusion

Here is a summary of the qualities and drawbacks of the various coordinates:

| | MVC | HC | GC |
|---|---|---|---|
| Computing time | ★★★ | ★ | ★★ |
| Simplicity | ★★★ | ★★★ | ★ |
| Flexibility | No | No | Yes |
| Defined everywhere | Yes | Yes | No |
| Partial deformations | ★ | No | ★★★ |

## References


[1] Tao Ju, S. Schaefer, J. Warren. Mean Value Coordinates for Closed Triangular Meshes.

[2] K. Hormann, M. S. Floater. Mean Value Coordinates for Arbitrary Planar Polygons.

[3] M. S. Floater, G. Kós, M. Reimers. Mean Value Coordinates in 3D.

[4] T. DeRose, M. Meyer. Harmonic Coordinates. Pixar Technical Memo #06-02

[5] P. Joshi, M. Meyer, T. DeRose, B. Green, T. Sanocki. Harmonic Coordinates for Character Articulations. Pixar Technical Memo #06-02b

[6] Y. Lipman, D. Levin, D. Cohen-Or. Green Coordinates.

[7] http://freespace.virgin.net/hugo.elias/graphics/x_polysc.htm

[8] The MorphoxX Project Home Page http://code.google.com/p/morphoxx