Landscape Specification Resizing

Leandro Cruz, Luiz Velho and Djalma Lucio
VISGRAF Lab - IMPA
Rio de Janeiro - Brazil
http://www.impa.br/~{lcruz, lvelho, dlucio}

Eric Galin, Adrien Peytavie and Eric Guerin
LIRIS Geomod - Universite Claude Bernard Lyon 1
Lyon - France
http://liris.cnrs.fr/~{egalin, apeytavi, eguerin}

Abstract—In this work, we introduce a method for resizing a landscape specification, i.e., a vector model containing a set of objects present in a virtual environment. Our goal is to change the landscape dimensions while keeping its overall appearance. Our method is based on the insertion and removal of objects in the specification, followed by some adjustments of the scene adapting the initial model to these changes. Besides the proposed approach for the positioning the objects, this method can be easily extended to use techniques of the state of the art for spreading objects in the landscape.

The adjustment of the scene components consists in performing vertical or horizontal translations onto the position of the objects. It is based on a removal or insertion of paths in the grid created over the scene space using dynamic programming. In spite of dealing with a vector model, all proposed operations are performed in the pixel space.

This technique is an adaptation of the Seam Carving for the context of vector landscapes specification. This model is simpler than common images, and thus, we can achieve results as good as those obtained in the image context, using simpler metrics.

I. INTRODUCTION

In general, large landscape models, such as those used in games, animations or some kind of simulations, are composed by a set of elements: the terrain topography, trees, rivers, houses, buildings, roads, etc. The 3D model is created from a specification model, i.e., a vector model specifying the position of each element, its shape, and some other attributes.

In this work, we present a technique to resize a given landscape specification. Our technique is based on the vector model of the scene. The main goal of this approach is to change the dimensions of the model preserving the overall appearance. Figure 1 illustrates this context. Figures 1a and 1b show the original model and the 3D landscape, respectively. Figures 1c and 1d show the enlarged version of this model.

In a game or in a simulation, it may be necessary to change the size of the scene according to the number of the characters. For instance, we can have a virtual environment being explored initially by N people, and in another situation, only half of those will explore that location. In this second case, it is not good keeping the same space, because the characters will be too far from each other. Thus, it may be interesting to reduce the area of the landscape returning to the initial proportion. Nevertheless, we do not want to change the overall appearance. In contrast, we want to obtain a smaller scenario by maintaining the same experience.

In general, a simple scaling of the model is not good enough, because it can produce some overlapping of objects. As well, a cropping of the model can remove some global feature of the landscape. Figure 2b and 2c illustrates this situation. On the other hand, a content-aware shrinking obtains an better resizing (Figure 2d).

We have adapted the traditional seam carving approach to the landscape specification context. However, it is a piecewise constant model. Thus, our seam carving approach is simpler than the methods used in image retargeting, still producing good results.

The core of our method works like a traditional seam carving. Its main functionality is to shrink or to enlarge the model by removal or by insertion of paths. Nevertheless, beyond replacing of objects, we will present how to insert and how to remove some objects in the landscape according to some criteria. Furthermore, this method can be easily extended, i.e. these criteria may be replaced by other technique of the state of the art for spreading objects in the landscape.

This paper has three main parts. The first one introduces the landscape resizing method. In Section III we present an overview of the technique and, in Section IV, we explain the resizing approaches. The second part, in Section V, regards to the discussion about some implementation details. Finally, in Section VI, we present some results and additional applications of our resizing approach.

Fig. 1. Landscape Resizing: specification model (left) and the respective 3D model (right)
II. RELATED WORK

Yeh et. al. [1] introduced an approach for the retargeting of scenes based on Markov chain Monte Carlo. Their model is based on a factor graph (a graph that encodes constraints as factors). It is a global approach applied on a specification with strong relations between the objects presented in the scene.

On one hand, the layout of an internal scene (spreading of furniture) and the buildings of a city are strongly constrained. On the other hand, there are many contexts where the constraints related to the position of the objects are weaker, for instance; the position of cities in a map, mountains in a landscape, houses and other buildings in a rural area (or in a medieval scene), trees in a field, etc. The work presented by Yeh et. al. [1] works well in the first situation, while the criteria we will present work better in the second one.

Our resizing approach is performed by applying shifts in the objects according to some inserted or removed paths in the scene. It is similar to the image resizing method introduced by Avidan et. al. [2].

A. Image Retargeting

Image retargeting has been widely studied in the last years. Valquero et. al. [3] presented a survey about this topic. The main motivation of this problem is to change the dimensions of an image to fit in a new region. It is becoming increasingly important due to the large variety of displays where the same image may be visualized. These displays differ in size and aspect ratio. Thus, these techniques go towards defining a good fitting: as a simple scaling or cropping is too limited, they try to perform a content-based resizing.

Image resizing can also be useful in photography. Liu et. al. [4] introduced a method to change the composition of the objects in some image to increase the aesthetic value.

Dekkers and Kobbelt [5] extended this retargeting approach to mesh deformation. They proposed a method that combines elastic and plastic deformation using the seam carving concept in the mesh editing context.

The content-based resizing problem is the same problem that we approach in this paper, as it is illustrated in Figure 2. Nevertheless, because the difference of the contexts (images and vector landscapes) there are differences between our method and those previously introduced for image resizing.

The Seam Carving approach [2] used for image retargeting is the base of our method to shrink and to enlarge the scene. The difference between our approach and the above methods is our model is simpler. Because of this, our method can also be simpler, but also producing results as good as those which use seam carving in the image or mesh context.

B. World Generation

There are a lot of methods to create the elements of a landscape. Smelik et. al. [6] discuss terrain synthesis, the creation and location of vegetation, water bodies, roads and buildings. They also discuss how to define a city layout, as well as, the spreading of furniture in building interiors.

An important element of the landscape is the terrain. However, the creation of the landscape topography is not the main goal of this research. For this purpose, we created a simple Digital Elevation Model based on the shape of some objects, and after, we added some noise in this model. Nevertheless, there are many approaches for terrain creation [7], [8].

There are many approaches for spreading objects in some specific models. However, all of these methods are based in specific rules according to the element type. Because of this, in this work we do not introduce complex methods (methods with hard constraints) for this operation, but we will show how to extend our method to use these particular rules.

Deussen et. al. [9] was the first to present a procedure to create an ecosystem to populate a height map. This method defines the position of plants through competition for resources. Alsweis and Deussen [10] extended this idea to use agents.


The procedural generation of cities is often hierarchical. For instance, it is possible first to create the city frontier and the neighborhood boundaries, after placing some landmarks, such avenues, streets, special buildings, squares, city blocks, and finally to create buildings and houses. Parish and Muller [13] introduced a seminal procedural method to create cities using L-systems. Recently, many other methods were presented using pattern-based approach, agent-simulations, tensor fields, etc. Watson et. al. [14] presented an overview of this topic.

In the same direction, the procedural definition of furniture layouts uses shape grammars [15], subdivision methods [16], tree-based or graph-based methods [17], among others.

It is important to highlight that the vector model has structural information which can be very useful in many of these methods. Thus, we can have a base model used as a seed of a procedural technique, and we can resize this model and use it again to generate another with different dimensions.

![Fig. 2. Comparisson between a simple cropping (b) or scaling (c) and our resizing approach (d)](image-url)
III. OVERVIEW

The specification is a vector-based model containing the position and the shape of all objects present in the landscape. As shown in Figure 1, this vector-based model is used as input to create a 3D scene according to this specification.

In general, a 3D landscape is created from a vector model, using procedural approaches, or by a combination of both. Our method fits in the third category. Our input and output are vector models, and we have some procedural approaches in the resizing pipeline.

Figure 3 shows the overview of the landscape creation (3a) and highlight our resizing pipeline (3b). In this paper we will only discuss the third step of this pipeline shown in Figure 3a. We assume that the model of each element is previously available. For this purpose, there are a lot of methods to generate these models [6]. The second step regards to the creation of the specification model. It is an initial version (or a base version) of the landscape created by the user. He should place each object of each element in the landscape domain, and define some parameters according to the insertion and the removal approach. This model is the input and output of the resizing method described in Figure 3b. Finally, we use the specification to create a 3D landscape visualized in some specific viewer (as well as, the visualization will be not approached in this paper).

One type of element may be copied in different positions of the specification. For instance, some kind of a tree may be placed at many positions of the scene. Because of this, we will refer to elements as the set of objects (each one placed at different coordinates).

Figure 3b shows the resizing pipeline. The enlargement and the shrinking are performed similarly. We first choose which element will be removed or inserted. After, we choose which object of this element will be removed, or where we will insert a new object. And so, we perform the insertion or removal.

This operation can create some holes in the model. Because of this, we have to adjust the entire model to this operation. This adjustment is a shifting of all objects after paths, created using Dynamic Programming, in the landscape region.

Even if we deal with a vector model, we will perform all operations in the pixel space (an image). Thus, we will create a planar map with a respective shape of each object.

We use the same 3D model (and the respective 2D shape) for all objects related to a specific kind of element in all experiments shown in this paper. However, it is not a real restriction of our method. It is possible to use more than one shape for the objects of the same element. It is only necessary to specify this model and its planar domain shape.

The planar shape of the object can be the shape of the element base or some other representation. In general, this map will consist of points, simple polygons or free form closed curves. Each one is associated to a 3D object such as a landform, a building, a tree, a checkpoint, etc.

Furthermore, we will have some objects represented by a set of connected curves. This kind of representation is well suited to rivers and roads.

Beyond the shape, we can also set some properties in the specification. These properties can be related to the resizing process, such as insertion and removal constraints, the term of increasing mask, or other parameter for control.

The operations that will be introduced will be performed in the pixel space. Most of these operations concern to the exterior area of the specification, i.e., the regions outside of all objects region (2D shapes). Basically, these operations are: path creation, computation of the distance field, insertion and removal of objects.
IV. LANDSCAPES RESIZING

In this section, we will discuss the resizing pipeline shown in Figure 3b. We will focus on the choice of the element to be inserted or removed, and on the necessary adjustments of the model to keep the initial aesthetic.

The choices related to the objects will be better discussed in the Section V-D. There, we will show some approaches to choose the object which will be removed, and other to choose where we can insert an object. Nevertheless, it is possible to use more specific approaches, as described in Section II.

A. Shrinking of Landscapes

The landscape shrinking (or decreasing) reduces its dimensions while keeping consistent the overall appearance. This method consists in repeatedly choosing of objects which will be removed, its removal, and the adapting of the model to the removal. The Algorithm 1 describes this method.

The basic approach to choose which object will be removed aims to keep the proportion of each type of objects (the amount of objects related to an element). This is a Greedy Algorithm. We first define a set $C$ of objects which may be removed. Thus, we choose one object of this element to be removed. We can perform a random choice, or use a better approach discussed in Section V-D. These procedures are performed in the functions chooseElement and chooseObject respectively, in Algorithm 1.

We define the set of candidate elements which can be removed as:

$$C = \{i; R_1(i) > (1.0 - \epsilon) \max_j R_1(j)\}$$

The function $R_1(i) = E_{curr}(i)/E_{init}(i)$ is the ratio of the current amount of objects of this element $E_{curr}(i)$ over the respective initial amount $E_{init}(i)$. And $\epsilon > 0$ is a small tolerance to enable the insertion of elements with values close to the maximum.

After the choice of the object, we remove it in two steps. First, we remove the object from the respective element list. Nevertheless, this removal creates a hole in the model. Thus, the second step is the adjustment of the specification to remove (or shrink) this hole. It is performed in the function compactSpecification in the Algorithm 1.

The compaction of the model is the main procedure of the decreasing method. It is introduced in Algorithm 2. This model is based on the removal of paths.

The path creation is performed using Dynamic Programming (DP). A path is a set of points, growing horizontally (Figure 5a) or vertically (Figure 5b). The $x$-th vertical path will begin at point $(x,0)$. As well, the $y$-th horizontal path will begin at point $(0,y)$. In each step, we have to decide the next path point, between $k$ possibilities ($k$ should be odd to keep the symmetry). Figure 5 shows an example for $k = 3$. In this case, to perform the horizontal growth, if the current point is $(x,y)$ (red pixel in Figure 5a) the next one will be $(x+1,y-1)$, $(x+1,y)$ or $(x+1,y+1)$ (blue pixels in Figure 5a). The vertical growth is analogous, as shown in Figure 5b.

The cost table of the DP procedure is created based on a mask in the pixel space. The mask maps the area which contains the model in an image (with some predetermined

**Algorithm 1** Decrease:

1: procedure DECREASE(desired_area)
2: area = calculateArea()
3: while area > desired_area do
4:   e = chooseElement()
5:   o = chooseObject(e)
6:   removeObject(o)
7:   compactSpecification()
8:   area = calculateArea()
9: end while
10: end procedure

**Algorithm 2** Compact Specification:

1: procedure COMPACTSPECIFICATION
2: path = calculatePath()
3: while path.isValid() do
4:   removePath(path)
5:   path = calculatePaths()
6: end while
7: end procedure
The choice of elements is again performed to keep the proportion of the amount of element’s objects in the specification. This is also a Greedy Algorithm. In this case, the set \( C \) of the candidate elements which can be inserted is:

\[
C = \{i; R_2(i) > (1.0 - \epsilon) \max_j R_2(j)\}
\]

where the ratio function is \( R_2(i) = R_1(i)^{-1} = \frac{E_{area}(i)}{E_{insert}(i)} \).

The \texttt{choosePosition} define where the object will be inserted. An approach is to insert a new element in the point farthest of all objects. We obtain this information looking for the maximum value in the distance field created in the exterior regions of specification.

After choosing the element and the position of the object, we check if there is enough space to insert it. It is performed by comparing two masks: the specification mask and another similar containing only the new object. It is performed in the method \texttt{intersects} in the Algorithm 3.

If it has some intersection between the internal regions we have to adjust the model. A basic approach is to perform a vertical or a horizontal shift of two steps for all elements after this position, and an analogous shift of one step for the position, until there is no more intersection. Nevertheless, this approach creates some spurious artifacts (such as holes) and destroys some desired patterns in the scene.

A better approach is to get the path more distant of all objects (a path passing on the ridges of the distance field). But, this path has to pass for the chosen position. To achieve it, we divide the path creation in two DP problems. The first problem calculates a path in the region below of the point (growing in a bottom-up way), and the other one calculates the path in the region above of it (growing in a top-down way), both passing for this point. The final path is a concatenation of the path created in the below region, with the inverse of the path created in the region above.

We can constraint the tables of DP problems (cost and neighbors) to the viable region, i.e., the region where it is possible to have a piece of a path which contains the given point. It is a triangular region, such that each row depends on \( k \) (amount of possible neighbors of each path point).

When enough space has been available by shifts, the object can be inserted. We repeat this procedure until the desired new size is obtained. Figure 6c illustrates this procedure.

### Algorithm 3 Increase:

1. \texttt{procedure INCREASE(desired\_area)}
2. \hspace{1em} \texttt{area = calculateArea()}  
3. \hspace{1em} \texttt{while area < desired\_area do}
4. \hspace{2em} \texttt{e = chooseElements()}  
5. \hspace{2em} \texttt{pos = choosePosition()}  
6. \hspace{2em} \texttt{while intersects() do}
7. \hspace{3em} \texttt{pos = shiftElements(pos)}  
8. \hspace{2em} \texttt{end while}
9. \hspace{2em} \texttt{insertElement(e, pos)}  
10. \hspace{2em} \texttt{area = calculateArea()}  
11. \hspace{2em} \texttt{end while}
12. \texttt{end procedure}
V. IMPLEMENTATION DETAILS

In this section we will discuss some implementation details about our method.

A. Dynamic Programming Approach

We use Dynamic Programming to calculate the paths to be removed and to be inserted.

In general, methods that create paths using DP should to set the constant $k$ carefully because of the local details. In our case, we do not have to deal with details. Because of this, we can use large values for $k$. The unique constraint is related to the processing cost, because larger $k$ values imply more processing. We are taking $k$ as 10% of the minimal dimension of the map.

The DP cost table is created to favor paths in the exterior region. The cost to insert a new path point is zero for points being outside of these regions and a big positive value otherwise. Thus, using DP, we calculate paths that do not intersect these regions (paths with cost equals to zero).

B. Object Mask

All operations proposed are based on the object mask. The first idea of the mask is to define the region where the 3D model will be positioned in the landscape. Another desired effect is to avoid two objects being too close after resizing.

We use the mask to decide if a point is inside of some object region. This image, that can be efficiently rendered, avoids a geometric approach (in general, expensive) to perform this decision. Figure 7 shows an example of the use of the mask.

To avoid that the objects being glued, we scale the object region during mask creation. Because of this, during the creation of DP cost table, we avoid the paths pass between two near objects. Of course, it will decrease the amount of possible paths, and thus, it will decrease the compactability.

To get a strip with a fixed width (in pixel space) we opt for scale regions using the distance field. If the distance of a pixel is less than a threshold then it is included into the mask. A geometric scaling does not create this enlargement of a strip with fixed width in non-convex cases.

Figure 7 shows three examples of different masks. Figure 7a shows the original model. The following images show the size reduction using mask increasing of $inc$ pixels. Observe that the objects in Figure 7d are not glued as in Figure 7b. However, the size decreasing was smaller.

C. Distance Field

We use a distance field to create the cost table of DP problem for the creation of paths that will be inserted, and to increase the masks. In both problems, we do not need an extremely precise distance value.

There are many methods to create the distance field. In our case, an approximation of this field is good enough. Thus, we opt for a distance field created by successive morphological dilations of the objects, as shown in Figure Figure 7h.

After changing the landscape model, it is possible to update the distance field performing local modification of the regions where the inserted or removed path passes.

D. Choosing Objects

We have mentioned that the object insertion and removal can be randomly performed. Nevertheless, during the creation of the specification the user can have used some criteria to spread the objects.

An example of insertion criterion is the objects of some element can have a higher incidence probability when they are close to objects of another element. Figure 9 shows an example where objects represented by the green points were inserted closed to some kind of objects.

To insert the element $e$ that is more likely to be close to the elements in the set $A$ we perform the following steps:

1) Create a mask with the elements of $A$
2) Choose $n$ candidates between the $e$'s objects
3) Create the Distance Field for these objects
4) Choose the candidate with the greatest distance

The first step ensures the new object will be close enough to the target. The last step uses the distance as a weight for the choice. Bigger distance bigger the probability of choosing the object. This maximum distance criterion avoids big changes in the scene, because the new object will be placed in the position with more space, and thus will be necessary less or no shifts.

As well, the choice of which object will be removed is very important to ensure all holes in the scene will disappear after of the path removal. For this purpose, a goal is to keep the distribution of the removed objects as well spread in the specification space as possible. This criterion maximizes the power of path removal.

Given an element e chosen according to the criterion presented in Section IV-A, the first step is to choose the n the candidates between the e’s objects (the set C). Then, we have to solve an optimization problem, which the objective function is defined according to some criterion.

We can use the maximum distance (in distance field) in the neighborhood of a given object. This neighborhood can be a simple circle centered in the object position, or a bigger region created using the increased mask approach.

Given an element, if all objects have the same shape then the first approach is fair, i.e. the external neighborhood area is the same for all objects. But, in the case of some element has objects with different shapes it is better to use the second approach, when the external area of each object will be a belt produced using the region increasing.

To choose objects using this criterion we have to solve an optimization problem based on the following objective function:

$$W(e.o(i)) = \alpha N(e.o(i)) + \beta O(e.o(i)) + \gamma E(e,i) - \delta R(e,i)$$

where $N$ is a function of the maximum distance in the neighborhood of $e.o(i)$, $O$ is the sum of the distance between the $i$-th object and the others $e$’s objects, $E$ is the sum of the distance between this object and all other objects of elements similar to $e$ (for instance, we can have more than one elements of the tree type), and $R$ is the sum of the distance between this object and all other removed objects of the elements similar to $e$. Observe that we want to maximize the last term, so we have to subtract the weighted value of $R$.

The choice of the weights is empirical. We opt for the following rules:

- $\alpha$ is directly proportional to the log of the amount of $e$’s objects
- $\beta$ is inversely proportional to the log of the amount of $e$’s objects
- $\gamma$ is inversely proportional to the log of the amount of objects of the elements similar to $e$
- $\delta$ is inversely proportional to the log of the amount of removed objects of the elements similar to $e$

Finally, this approach is based on the following optimization problem:

$$o = \arg\min_{i \in C} W(e.o(i))$$

As previously mentioned, our technique is easily extendable to add other insertion and removal criterion. For this purpose, it is only necessary to change the function $W$. Those criteria can be used according to some specific requirement related to the landscape creation.
VI. RESULTS

A constant problem in procedural modeling approaches for landscapes creation is the definition of metrics for the evaluation of the quality of the results. The visual aspect is the most relevant metric for the most techniques. Some approaches consider mathematical or statistical models related to some natural phenomena. In our approach, the visual aspect is very important, but we will use other metrics to reinforce the quality of our results.

All metrics presented in this paper aims to preserve the proportion of the objects of each element after the resizing. Furthermore, these metrics should preserve the overall appearance. In this case, we can quantify this aesthetic property, for example, by using the mean of the distance between each pair of objects of the same type, maximum distance (in distance field), and the average of the maximum distance in the neighborhood of each object.

The introduced approach to choose the position of the objects was based in these metrics and the obtained results keep the original values (within a small error). Figure 11 shows an example where the original model (11b) was shrunk (11a) and enlarged (11c). Table 1 shows how these values were changed by the resizing. The last three rows mean the percentage of the amount of the objects of each type of element.

In some landscape models, some objects are more important than others (for instance, a hill is more important than a common tree). The result shown in Figure 9 was created considering this fact. Figure 9a shows a base specification, with the most important elements. The final specification (Figure 9b) is an extended version, under which we add other less important objects. The resizing was performed in the base model and kept the most important structures of the scene (the shrunk version shown in Figure 9c). As well, after the resizing, we add other objects to compose the final scene (9d).

In the following subsections, we will discuss about the results related to some variations of our resizing method.

A. Resizing Curves

In general, the most elements in a landscape can be represented by a polygon, a free form closed curve or a point. On the other hand, the best representation for some elements can be a curve (or a set of curves). It happens, for instance, with the representation of rivers and roads.

This kind of element is not well fitted in our resizing approach. The main reason is, it is very easy to have a landscape where there are no paths that do not cross the curve (e.g. Figure 10). Hence, we do not add these curves in the mask, and thus, we can have paths crossing the curve. On the other hand, because our curve representation is the entire set of points of the curve (i.e., we do not use a control point representation) the shifts are performed at the same way. Nevertheless, it is possible to create some artifacts in the curve.

An improvement for this method is to represent the curve as a spline curve and perform the changes of the control point position carefully to avoid overlapping. Another possibility is to keep a low resolution curve and perform midpoint displacement subdivision [7] to add detail in the river (but again, performing it carefully to avoid overlapping).

<table>
<thead>
<tr>
<th></th>
<th>Shrinked</th>
<th>Original</th>
<th>Enlarged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>525 x 352</td>
<td>900 x 600</td>
<td>1350 x 869</td>
</tr>
<tr>
<td>Aspect Ratio</td>
<td>1.49</td>
<td>1.5</td>
<td>1.55</td>
</tr>
<tr>
<td>Max distance</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>mean distance</td>
<td>13.3099</td>
<td>12.1935</td>
<td>12.4643</td>
</tr>
<tr>
<td>Free forms</td>
<td>0.882353</td>
<td>1.11628</td>
<td>1.46341</td>
</tr>
<tr>
<td>Rectangles</td>
<td>11.8487</td>
<td>12.186</td>
<td>11.9512</td>
</tr>
<tr>
<td>Points</td>
<td>87.2689</td>
<td>86.6977</td>
<td>86.5854</td>
</tr>
</tbody>
</table>

Table 1: Resizing metrics

Fig. 9. Using a basic model in the resizing, and after that, inserting detail in the scene

Fig. 10. Objects represented by curves
B. Retargeting of Landscapes

In general, landscapes are presented by a rectangular model. In the case of planets, in general the model is represented by a set of rectangular charts (an atlas). However, it is possible to work with a different kind of shapes. For instance, it happens when we are working with a limited model, such as, an island or a biome.

Our method can be easily adapted to deal with non-rectangular models. We deal with this situation using the following two approaches:

- Using a parameterization to transform the rectangular domain in the desired shape (and its inverse function)
- Using a closed free form curve to define the boundary of the landscape

In the first situation, we only have to change the space where the operations are performed, i.e., the creation of the increased masks and the distance field is performed in the model transformed by the parameterization. Nevertheless, the paths are defined in the rectangular domain. For this purpose, we create the cost table of the DP problem evaluating the transformed points, i.e., each point of the discretization of the domain is transformed, and then, the cost is evaluated in the image of this function.

Figure 12 shows an example of the original model, (12a) transformed by a given parameterization (12c), being enlarged (12b and 12d) using this approach.

In the case of a free form curve defining the landscape boundary the difference is to constraint all operations to the specified region. It is performed using an auxiliary mask to define this region. For this purpose, the DP problem is adapted: the points in the area outside of the landscape region receive a penalty (much smaller than the value used in the interior of the object’s region). We perform the resizing while we can create paths majorly inserted in the landscape region. Figure 13 shows an example of the shrinking of a free form area.

C. Composed Objects

There are some contexts where we can handle with composed objects, i.e., objects composed by a set of other objects. This kind of objects can represent more sophisticated elements, as cities, chain of mountains, forests, etc. It introduces a concept of multiresolution in our approach.

D. Resizing Vector Images

This technique is easily adapted to resize vector-images. The unique modification is to change the planar shape used in masks for some component of the image.

In the image case, it is possible to extend this method to add some semantic in shifts operations. For instance, to limit the movement of some objects to some part of the image (e.g. some objects can be constrained to stay below of the skyline), or decrease the size of the shape when it get closer of the skyline.
difficult problem in this area. Thus, it could be used for the creation and editing of virtual worlds from a high level perspective: a very multiresolution. Moreover, by recursively creating and remove objects, we obtained a powerful tool for the resizing to produce an equivalent result.

Figure 14 shows an example where we shrunk an initial vector image. In this example, we only decreased the width to keep the depth sensation. After the deformation using the same approach used for landscape resizing, we used the specification model to create an image. We replaced each mask for a small image creating the final model.

VII. Conclusion

In this work we introduced a method to resize a landscape specification keeping the overall appearance. This method is based on the insertion and removal of objects followed by a scene adjustment (enlargement or shrinking). We have adapted a complex approach used for image and mesh editing to a simpler context: a vector landscape model. For this purpose, we obtained good results using a simpler approach.

We have introduced simple and effective methods to insert and remove objects in the scene. Moreover, our method can be easily extended to use the state of the art techniques for spreading objects in a scene.

The main application of our method is to change the dimensions of a vector model to adapt the respective virtual environment, for instance, to be explored by a different number of characters. Furthermore, it is possible to use this approach to create large landscapes with the same overall appearance of a small model created by the user.

A natural future work of this research is to use more complex constraints for the insertion and removal of objects. We can perform these operations keeping a structured layout.

A drawback of our method is the resizing is not invertible. Thus, it is not possible to back to the exact previous configuration after the resizing. However, all modifications keep the initial appearance.

Furthermore this method is sensitive to the choice of some parameters: the mask increasing, the attraction (the incidence of some kind of objects closer of objects of some specific element) and the $k$ parameter of the DP problem. These parameters have to be coherent to the initial configuration for the resizing to produce an equivalent result.

By combining our approach with good methods to insert and remove objects, we obtained a powerful tool for the creation of huge landscapes. Moreover, by recursively creating structured elements, the proposed approach can be applied in multiresolution. Thus, it could be used for the creation and editing of virtual worlds from a high level perspective: a very difficult problem in this area.

ACKNOWLEDGMENT

This research has been developed at Visgraf Lab - IMPA, which is sponsored by CNPq, FAPERJ, FINEP, and at LIRIS - CNRS.

Furthermore, the first author was being supported by CAPES to visit the LIRIS for the development of this research.

REFERENCES