Graphic rendering of synchronized objects specification
v.1.04

Grame
Centre national de création musicale

May 16, 2014

Introduction

An augmented music score is a graphic space that supports music scores and arbitrary graphic resources, including real-time elements, and providing time synchronization between all the score components. Each of those elements possesses a time position and a mapping that bounds their graphical dimensions to the time space.

In a context of interaction between elements (and possibly in the future with other musical processes), we define two types of dependencies between elements : the hierarchy and the synchronization. The first one on a mainly graphical base, and the second one on a time base, influencing the graphical representation.

The aim of this specification is to define the behavior of multiple or chained synchronizations, involving objects with children and slaves.

1 Two types of relations between objects

In order to understand the problematics of graphic rendering for the synchronized objects, we should first try to describe the two possibilities of interaction between objects in an augmented score : the parent-child relation, and the master-slave relation.

1.1 Hierarchy (parent-child relation)

"A is parent of B" will be symbolized by : $A \sqsupset B$

"B is child of A" will be symbolized by : $B \sqsubset A$

With the 1.04 version of INScore, any object of the scene can have children objects, each one corresponding to a subnode in their OSC address.

This is a purely graphical relation : the child object is drawn by its parent and in its parent's relative coordinates. It has no meaning in the time space.

As the hierarchy between objects defines their OSC path, we can therefore agree that a single object can only have a single parent (but several children).

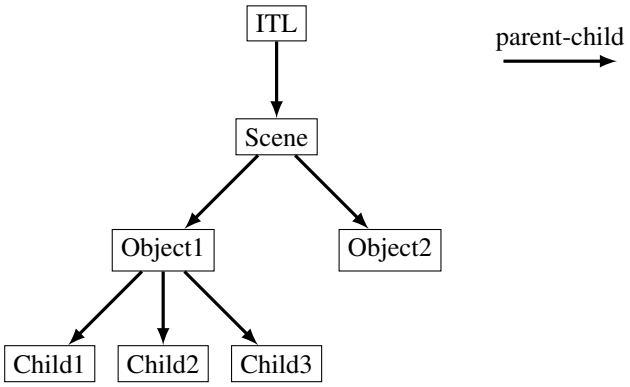


Figure 1: The parent-child hierarchy

1.2 Synchronization (master-slave relation)

"A is master of B" will be symbolized by : $A \leftarrow\!\!\prec B$

"B is slave of A" will be symbolized by : $B \succ\!\!\rightarrow A$

Time synchronization represents the possibility to graphically synchronize the augmented score components in order to align the graphic sections of the score components that correspond to equivalent time spaces. Each object on the augmented score has a time position and a duration. The other objects on its level of hierarchy (objects having the same parent) can then be synchronized on it, which means that they will be positioned, and possibly stretched, to fit their time position and duration relatively to their master's, and depending on the synchronization mode.

Both of the slave and master objects should have their own time to graphic mapping, that can then be used to create a graphic to graphic mapping corresponding to the direct link between the two elements.

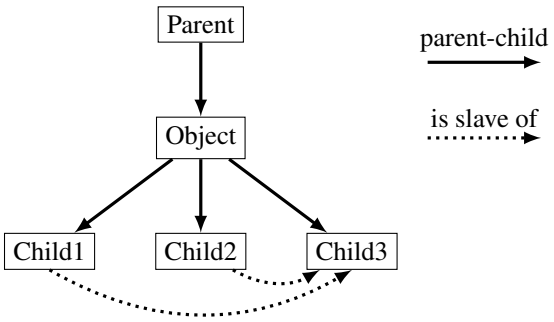


Figure 2: The slave-master synchronization

A master can have several slaves and a slave several masters. Each master-slave relation should have its own options that are :

- *syncHow* that indicates if the relation should be in absolute or relative time,
- *syncPos* that indicates if the slave should be placed on the top, on the bottom or over the master,
- *syncStretch* that can be :

- none, if there is no stretching and the slave's drawing only depends on its time position, but not on its duration,
- *h* for a horizontal stretching to fit the slave's duration,
- *v* for a vertical stretching (with no time meaning),
- *hv* for a vertical and horizontal stretching.

This graphic to graphic mapping defines graphical segments in the slave (source) element, that will correspond to graphical segments in the master (destination) element. The set of all those segments is called segmentation.

In practice, the graphic to graphic mapping is built by calculating a list of pairs of graphical segments (or rectangles), each one of these pairs representing the path from the slave's to the master's spaces.

Without specification, the default segmentation will be a unique segment corresponding to the object's bounding box.

The graphical segmentation of the object A will be represented by Seg_A

The bounding box of the object A will be represented by S_A

The time segmentation of the object A will be represented by T_A

The time to graphic mapping of the object A will be represented by $M_A = Seg_A \times T_A$

The graphic to graphic mapping of a synchronization between two elements will be represented by $M_{slave/master}$

The date (time position) of the object A will be represented by t_A and its duration (time segment) by d_A.

The origin point of a rectangle S_A will be represented by (x_{S_A}, y_{S_A}), its x-segment by w_{S_A} and its y-segment by h_{S_A}.

2 Multiple synchronizations

As said before, it is possible for a slave to have several masters. This means that it will be drawn as many times as the number of masters, and have as many mappings (a list of lists of pairs of rectangles !). The object's appearance will vary depending on its relation with each of its masters (stretch or not, placed on the top or on the bottom, in a relative or absolute time...).

```
/ITL/scene/parent/sync child1 child2;
/ITL/scene/parent/sync child1 child3 h;
/ITL/scene/parent/sync child1 child4 hv;
```

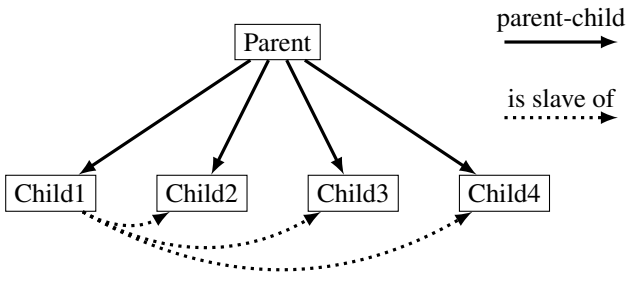


Figure 3: Example of multiple synchronization

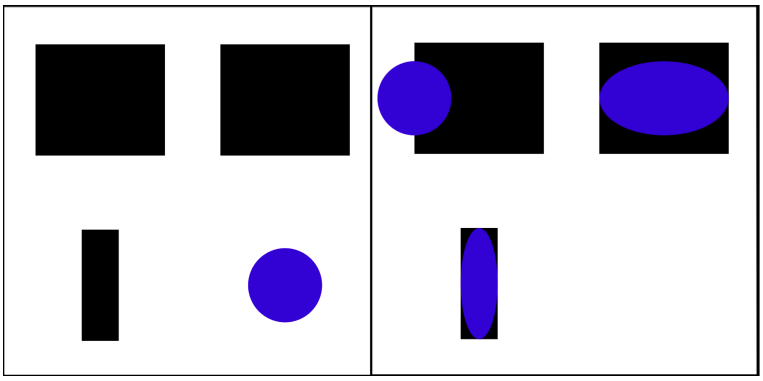


Figure 4: The purple circle is synchronized on the three other objects

3 Synchronization on objects with children or slaves

Now that we have defined the main characteristics of both graphic and time hierarchies, we should try to specify the behavior of synchronized objects, that themselves may have children and/or slaves : What should we do with those children and slaves ? Should they be drawn with their parent or master ? Should they be stretched with it ? Mapped ?

The object A with all its children and slaves will be represented by Å.

3.1 Bounding Boxes of parent and children

First of all, we should define the rectangle $S_{\mathring{A}}$ corresponding to the set of the object with its n children and slaves.

$$S_{\mathring{A}} = \{\bigcup_{i=1}^{n} S_i\} \bigcup S_A$$

The union of segments used here is defined as follows :

$$I \bigcup J = [\min\{\min_{i\in I}, \min_{j\in J}\}, \max\{\max_{i\in I}, \max_{j\in J}\}[$$

This works for one a more dimensions. The resulting n-dimensional segment being the union of the projections in each dimension.

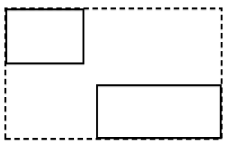


Figure 5: Union of two rectangles (graphical segments)

The children having the possibility of being drawn outside of their parent's bound, the general case is :

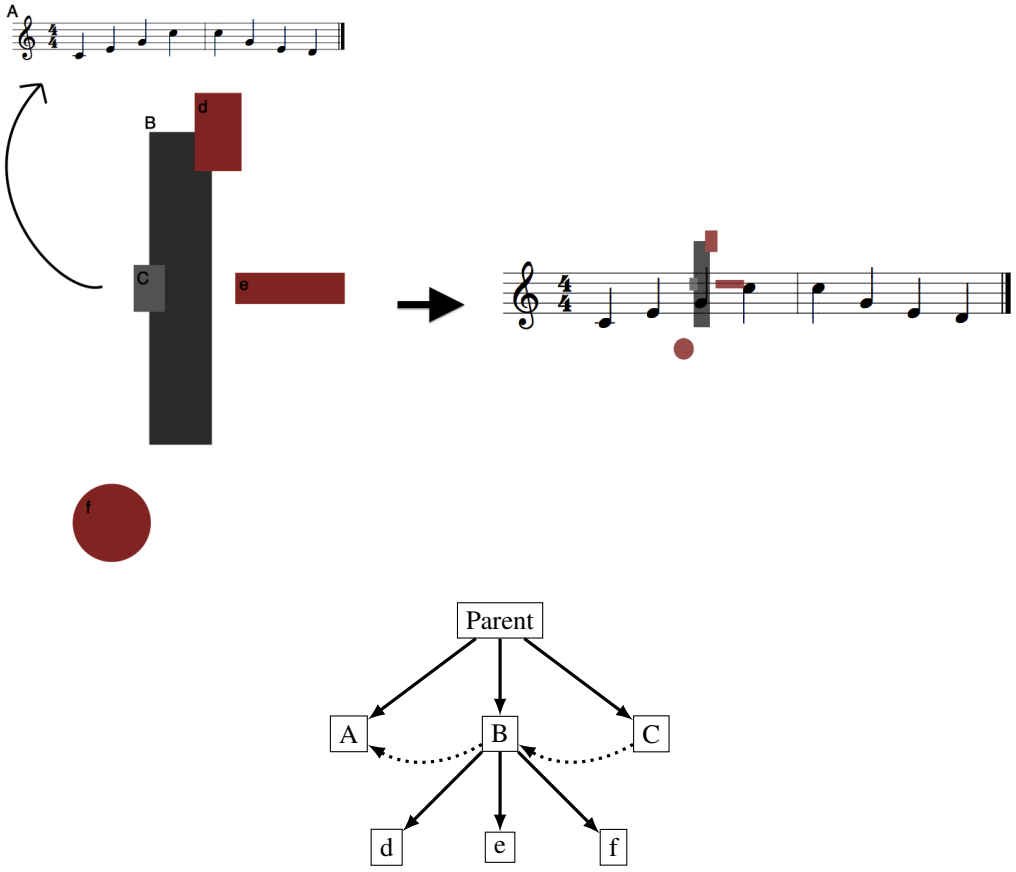
$$i \sqsubset A \not\Rightarrow S_i \subset S_A \quad \text{and} \quad S_A \subseteq S_{\mathring{A}}$$

and so :

$$S_{\mathring{A}} \bigcap S_A = S_A$$

3.2 Case of the default mapping for the slave

Let's take the following case of an object B synchronized on a score A and having a slave C and children d, e and f.



$B \rightarrowtail A$
$B \sqsupset (d, e, f)$
$B \rightarrowtail C$

Figure 6: Synchronization of an object with children and slaves

In the case of a default mapping, $Seg_B = \{S_B\}$. We are then dealing with a synchronization without horizontal stretching (no stretching or with a vertical stretching).

We know that the segment (or, in this case, bounding box) S_B is linked, through a graphic to graphic mapping, to another segment S'_B that depends on the master object A :

$$(S_B, S'_B) \in M_{B/A} \text{ with } M_{B/A} = Seg_B \times Seg_A$$

The aim here is to define this segment S'_B, but also to find the corresponding segment $S'_{\mathring{B}}$ linked to $S_{\mathring{B}}$, the bounding box of $\mathring{B}$.

3.2.1 Variety of a segment

To define our destination segments S_B and $S'_{\mathring{B}}$, we will appeal to the concepts of continuous mapping and variety [1].

For $\theta : [0,1] \to [0,1]$ and $I = [a,b[$, we name variety of I, the set of points $\mathcal{V}(I,\theta)$ from I defined by :

$$\mathcal{V}(I,\theta) = \{(1-\theta(t)).a + \theta(t).b \mid t \in [0,1[\}$$

Intuitively, the variety of an interval expresses the relationship between this interval and its variety using a function θ defined on $[0,1[$.

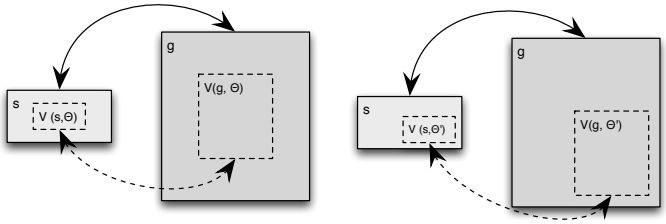


Figure 7: Varieties of two graphical segments

3.2.2 Definition of $S'_{\mathring{B}}$

Let's now remember that the INScore scene has time representation on its horizontal (x-axis) dimension. The vertical one being purely graphical.

If we look for the position and dimension of our destination segment S'_B, we should first try to find the relation between the time position of B, t_B, and an x-position in the mapping M_A.

As long as we have a time segment D_A in T_A (the time segmentation) that contains the time position t_B, we know that we have a graphical segment S_A in Seg_A (the graphical segmentation) that will contain our slave's position. If there is no such segment in T_A, the slave will simply not be drawn.

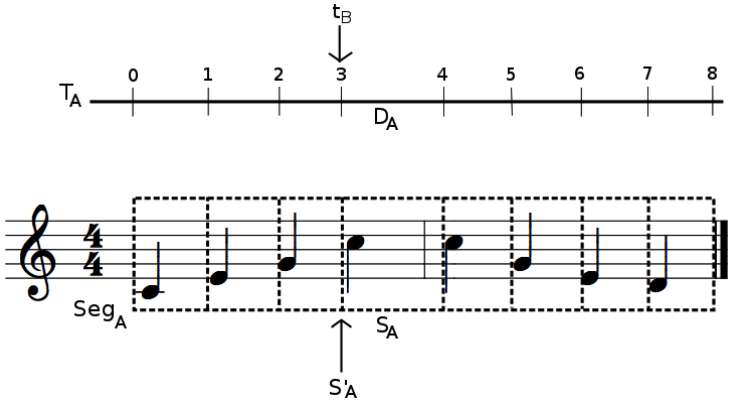


Figure 8: Relation between time and graphical mappings

We can see the time position t_B as a time segment $d_B = [t_B, t_B[$ that is a variety of D_A. We can then deduce a graphical segment S'_A corresponding to the time segment d_B :

$$d_B = \mathcal{V}(D_A, \theta)$$
$$S'_A = \mathcal{V}(S_A, (\theta, 1))$$

This relation gives us a one-dimensional segment S'_A with :

- the x position corresponding to the time position t_B,
- a null x size
- the y position centered on S_A
- and a y size corresponding to the height of S_A.

In order to find our destination segment S'_B, we will define it as :

- with its x origin aligned on S'_A,
- with a x size corresponding to the source S_B width,
- with y position and size depending on the vertical stretch and on the synchronization option "syncPos" (see Figure 9).

$$S'_B = [x_0, x_1[\times [y_0, y_1[\;\Big|\; \begin{cases} x_0 = x_{S'_A} - (1 + x_{S_B}) \times w_{S_B}/2 \\ x_1 = x_{S'_A} + (1 - x_{S_B}) \times w_{S_B}/2 \end{cases} \;\Big|\; \begin{cases} y_0 = y_{S'_B} - (1 + y_{S_B}) \times h_{S'_B}/2 \\ y_1 = y_{S'_B} + (1 - y_{S_B}) \times h_{S'_B}/2 \end{cases} \;\Big|\; \begin{cases} y_{S'_B} = \begin{cases} y_{0S'_A} - h_{S'_B}/2 & \text{if syncTop} \\ y_{S'_A} & \text{if syncOver} \\ y_{1S'_A} + h_{S'_B}/2 & \text{if syncBottom} \end{cases} \\ h_{S'_B} = \begin{cases} h_{S_B} & \text{if no stretch} \\ h_{S'_A} & \text{if vertical stretch} \end{cases} \end{cases}$$

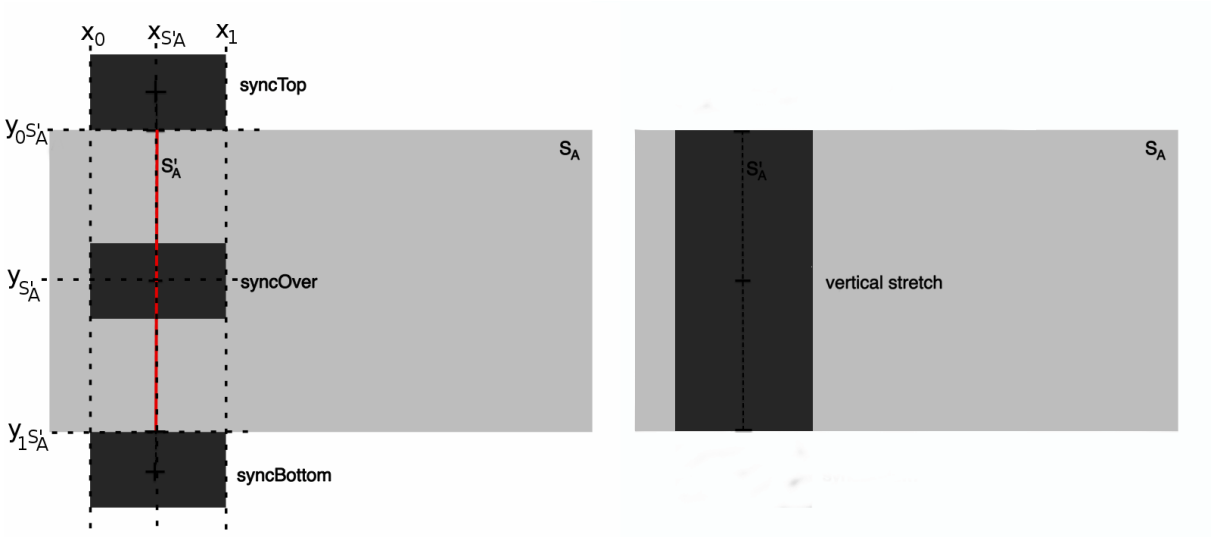


Figure 9: Different possibilities for the destination segment S'_B

3.2.3 Definition of $S'_{\mathring{B}}$

We can see that, if we know S_B (the object's bounding box) and $S_{\mathring{B}}$ (the bounding box of the object and all its children and slaves), we can define the relation between them, which is the variety.
Furthermore, the mapping $M_{B/A}$ gives us the destination segment S'_B, so we can deduce the segment $S'_{\mathring{B}}$ by applying the same relation between S'_B and $S'_{\mathring{B}}$ than between S_B and $S_{\mathring{B}}$.

for :

$$(S_B, S'_B) \in M_{B/A} \text{ and } M_{B/A} = Seg_B \times Seg_A$$

we have :

$$(S_{\mathring{B}}, S'_{\mathring{B}}) \mid \begin{cases} S_{\mathring{B}} \cap S_B = S_B \\ \exists \theta \mid \begin{cases} S_B = \mathcal{V}(S_{\mathring{B}}, \theta) \\ S'_B = \mathcal{V}(S'_{\mathring{B}}, \theta) \end{cases} \end{cases}$$

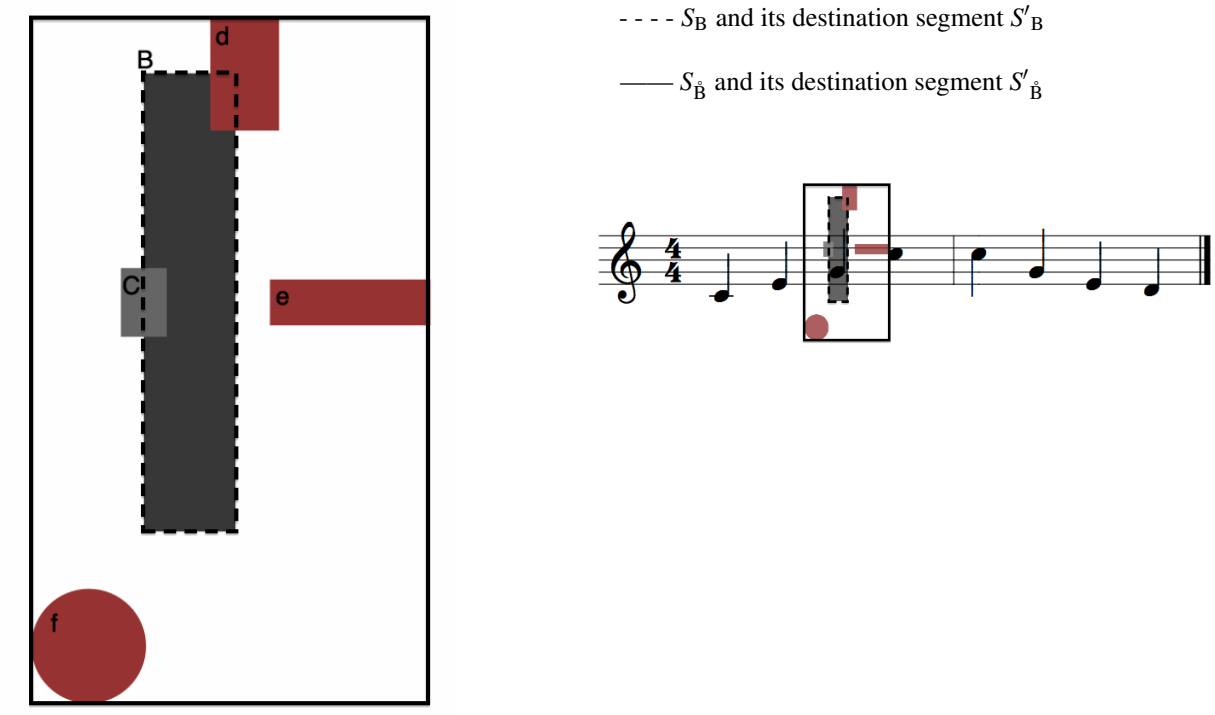


Figure 10: The source and destination graphical segments

3.3 Case of stretching (the slave is mapped)

In the case of a slave with mapping that should be stretched horizontally to fit its parent's mapping, it has been decided to take only into account the children that are in the object's bound. As our object can be partially mapped, and then partially drawn in its master's mapping, there are no more reasons to extrapolate the drawing of its children than the drawing of the rest (not in the mapping) of the object itself.

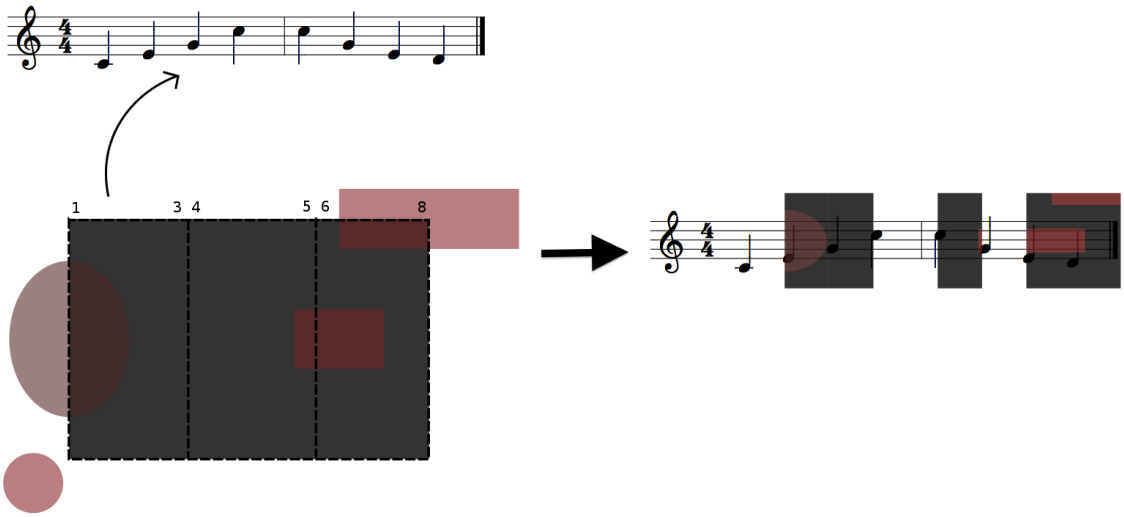


Figure 11: Case of a slave with mapping

4 Updates

Once those questions of mapping resolved, we should finally try to define the order of the updates.

Indeed, the computation and the drawing of our elements have to be done in a certain order, so that every step can have all the informations needed. For the mapping updater, we need to know for each object the positions and dimensions of its children and slaves, but for the view updater, the masters must be drawn before the slaves. This means that we should "climb" the hierarchy from the bottom (objects without any slave) to the top (object without any master) for the Model, and do the opposite for the View.

References

[1] Dominique Fober, Frederic Bevilacqua, and Roland Assous. Formalizing segments and their relationships in music, sound and gestures representations. *(submitted to) Computer Music Journal*, 2012.