Continuous Constraint Programming - Project

The project can be done in a group (max. 2 students) who should present a report with a brief discussion of the options taken in the implementation. The report must be sent by email to the lecturer (jcrc@fct.unl.pt) together with the code implemented no later than 2 Jan 2022.

Implement a program that executes a sequence of commands defined in a text file named "commands.txt". Each line of the file starts with an identifier of the command (a single word) followed by the parameters needed to execute the command (separated by single spaces).

Whenever you find a command that is not implemented by your program, skip to the next command.

When the result of the command is to draw a 2D object, a new figure in Vibes should be generated with a title identifying the command and its parameters.

The command BOUNDING will always be in the first line of the file (occurring only once), affecting the visualization of all the other commands. To draw all the figures generated in the next commands assume a precision of 0.01, that is, do not split intervals with sizes smaller than 0.01.

In the following, each command is described in detail:

- BOUNDING *xmin xmax ymin ymax*

(where *xmin*, *xmax*, *ymin*, *ymax* are real values with *xmin*<*xmax* and *ymin*<*ymax*)
This command defines the Cartesian coordinates of a bounding box that delimits all the figures generated in the next commands. Nothing is draw outside the box: $[xmin, xmax] \times [ymin, ymax]$.
In the following examples we assume that the first command was: BOUNDING 0 10 0 10

- POLYGON *name*

(where *name* is the name of a polygon and "*name*.txt" identifies a text file defining a convex polygon as a sequence of *n* vertices. The file contains *n* lines, each line *i*, representing vertice *i*, has 2 real numbers x_i and y_i, separated by a space. It is assumed that there is an edge between each two consecutive vertices and an edge between the last vertice and the first. The 2D object may be identified in future commands by its name)
This command draws the visible part of the polygon in the bounding box defined previously in the first command. Regions of the bounding box that are proved to be inside the polygon (inner regions) should be drawn in green (green boxes with black borders), whereas regions that may contain points inside the polygon (boundary regions) should be drawn in red (red boxes with red borders).

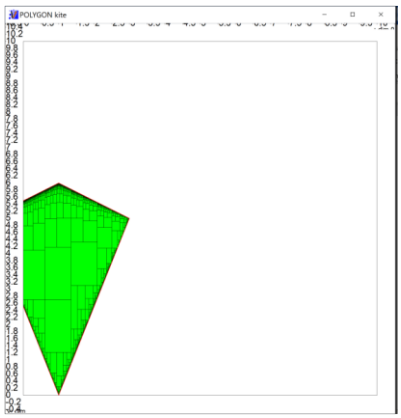
Example: POLYGON kite

with the following text file,

kite.txt

```
1 0
-1 5
1 6
3 5
```

produces the window:



- SHAPE *name*

(where *name* is the name of a shape and "*name*.txt" identifies a text file defining a generic 2D shape as a constraint system in the Minibex language. This system contains 2 unbounded variables, x and y, and a set of inequality constraints defining a non-null area. The 2D object may be identified in future commands by its name)
This command draws the visible part of the 2D object in the bounding box defined previously in the first command. Regions of the bounding box that are proved to be inside the shape (inner regions) should be drawn in blue (blue boxes with black borders), whereas regions that may contain points inside the shape (boundary regions) should be drawn in red (red boxes with red borders).

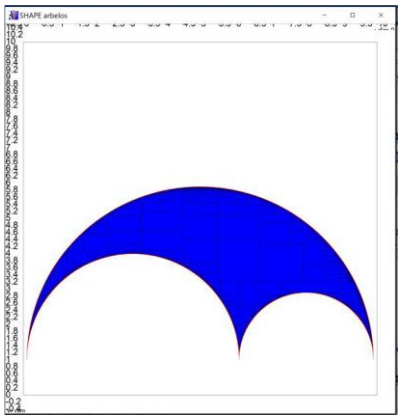
Example: SHAPE arbelos

with the following text file:

arbelos.txt

```
constants
        r1 = 4.9;
        r2 = 3.0;
        c1 = 5.0;
        c2 = 1.0;
variables
        x in [-oo,oo];
        y in [-oo,oo];
constraints
        (x-c1)^2+(y-c2)^2 <= r1^2;
        (x-(c1-r1+r2))^2+(y-c2)^2 >= r2^2;
        (x-(c1+r2))^2+(y-c2)^2 >= (r1-r2)^2;
        y >= c2;
end
```

produces the window:

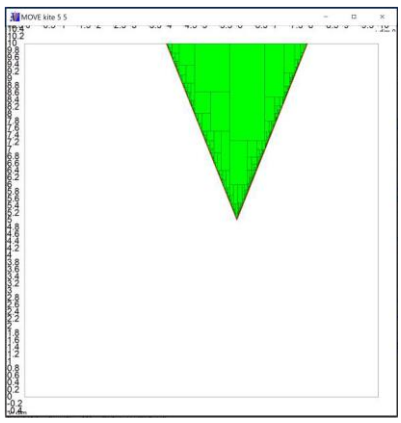


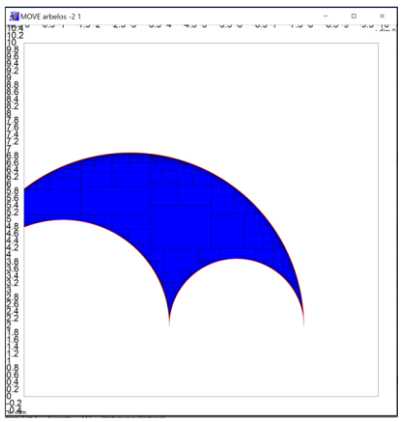
- MOVE *name tx ty*

(where *name* is the name of a 2D object previously defined by a POLYGON or SHAPE command and *tx* and *ty* are real values).
This command draws the visible part of the 2D object after performing a translation $\vec{t} = (tx, ty)$ relative to the original definition. Regions of the bounding box that are proved to be inside the object (inner regions) should be drawn in the color of the object (green if is a polygon or blue if is a shape), whereas regions that may contain points inside the object (boundary regions) should be drawn in red.

Example: MOVE kite 5 5
MOVE arbelos -2 1

produce the windows:



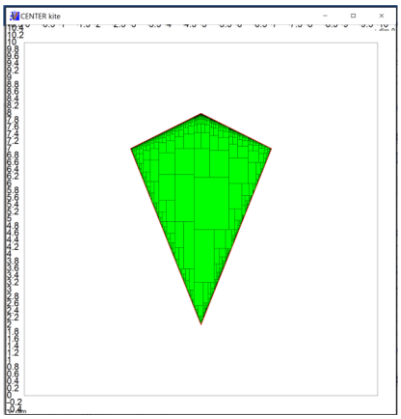


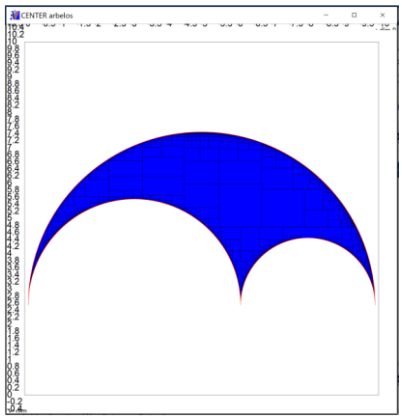
- CENTER *name*

(where *name* is the name of a 2D object previously defined by a command POLYGON or SHAPE)
This command draws the 2D object completely inside the bounding box and centered. This should be achieved by finding and applying an appropriate MOVE. If no such move is possible nothing is drawn.

Example: CENTER kite
CENTER arbelos

produce the windows:



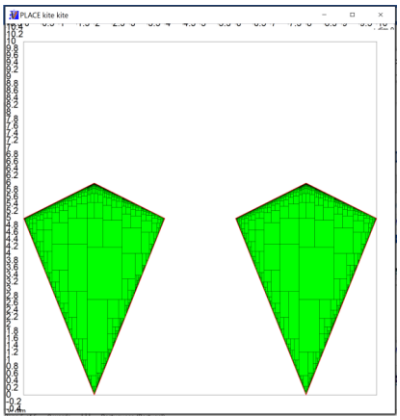


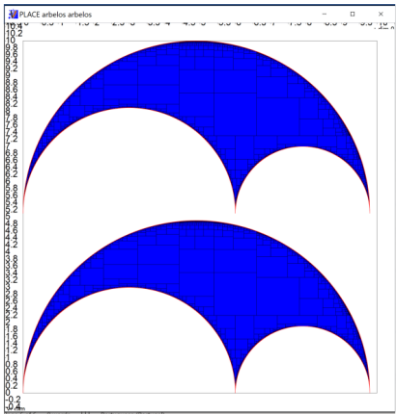
- PLACE *name1 name2*

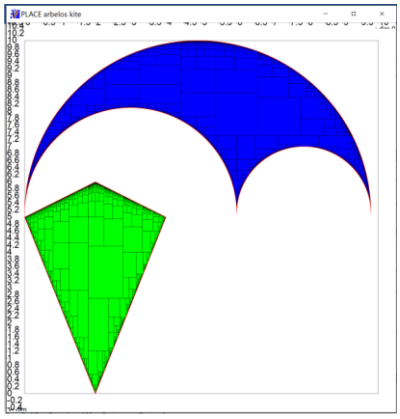
(where *name1* and *name2* are the names of 2D objects previously defined by POLYGON or SHAPE commands)
This command draws both 2D objects completely inside the bounding box without overlapping. This should be achieved by finding and applying appropriate MOVE commands for each object. If no such moves are possible nothing is drawn.

Example: PLACE kite kite
PLACE arbelos arbelos
PLACE arbelos kite

may produce the windows (there are many other configurations equally correct):







To produce all previous examples the commands text file should be:

commands.txt

```
BOUNDING  0 10 0 10
POLYGON kite
SHAPE arbelos
MOVE kite 5 5
MOVE arbelos -2 1
CENTER kite
CENTER arbelos
PLACE kite kite
PLACE arbelos arbelos
PLACE arbelos kite
```