Animated proofs made easy

Frédéric Gourdeau
fredg@mat.ulaval.ca

and

Guillaume Paré
guilpare@globetrotter.qc.ca

Université Laval, Québec, Canada

English summary

The aim of the course was to enable users of Cabri, who were neither experts nor novices, to construct animated proofs by themselves. In order to achieve this, four macros (available separately or as part of a tool bar) were constructed by the authors. These were shared with the participants and their basic functioning was explained.

After a first day of introduction and a guided first construction, participants constructed one or two animated proofs. The instructions for these constructions are in the two documents which follow this summary. They are complete and self-explanatory and should enable the reader to understand what was done in the course.

The method and the tools we proposed proved sufficient to achieve our goal. We have included some figures realised by participants after the course to illustrate what was achieved.

We hope that you will be able to use our construction tools to fruition. However, we ask that any figure built using these tools (or the method) be freely available to everyone. This, we think, is one way to ensure that dynamic geometry benefits more people.

List of Cabri files available

Examples of animated proofs:
1- Pyth-A.fig
2- Pyth-B.fig
3- CercleA.fig
4- CercleB.fig
5- Rotation.fig
6- Exemple1.fig

Constructions realised by participants:
1- AirePReg.fig: figure constructed by Monique Morel[1]
2- Aire1.fig: figure constructed by Alicia Noemí Fayó[2]
3- Aire2.fig: idem
4- Aire3.fig: idem
5- Aire4.fig: idem

[1] Monique Morel, mmorel@cstrois-lacs.qc.ca , teacher of mathematics, École secondaire Vaudreuil, Commission scolaire des Trois-Lacs, Québec, Canada
[2] Alicia Noemí Fayó, aliciafayo@ciudad.com.ar, professor of mathematics and computer science, Grupo de.Investigación Matemática XVIII, R. Argentina.

Construction of macros:
1- ConsP1.fig
2- ConsP2.fig
3- ConsPdeT.fig
4- ConsPdeR.fig
5- ConsPoly.fig

Macros:
1- Point_de_rotation.mac
2- Point_de_translation.mac
3- Point_conditionnel.mac
4- Polyconstruction.mac

Tool bar:
1- Outils.men

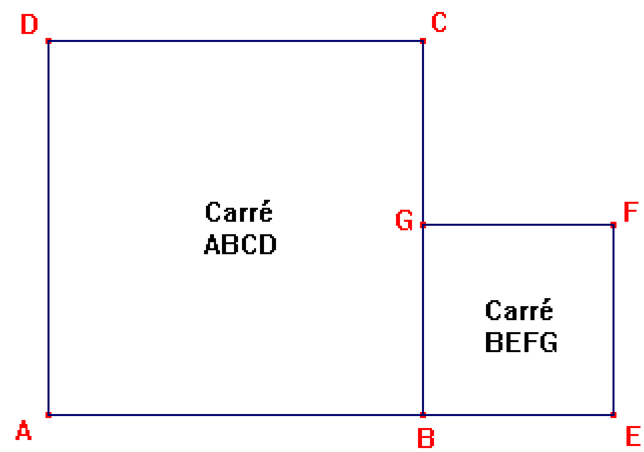
Document 1 – First construction of an animated proof of Pythagoras Theorem

Follow these steps to construct an animated proof of Pythagoras' Theorem similar to Pyth-A.fig

First part:
<u>The initial squares</u>

We want to construct two squares with different dimensions and an adjacent side. To do so, you may proceed as follow:

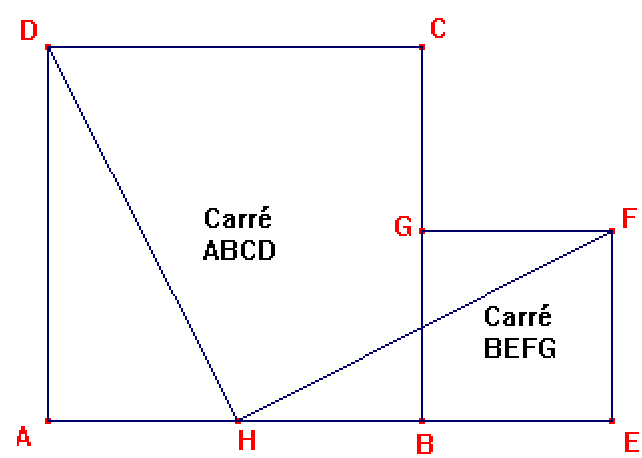
1. Construct a line at the bottom of the screen. This will be your base line for the squares.
2. Place two points on this line to form the basis of square *ABCD*. Now complete the square using perpendiculars and circles.
3. Construct another square *BEFG*, making sure to use the same point *B* for both squares.
4. Hide the superfluous elements while keeping the vertices of the squares.



Second part:
<u>The initial triangles</u>

We wish to show that the sum of the areas of the two squares built on the sides of a right-angled triangle (with lengths equal to that of *AB* and *BE,* respectively) is equal to the area of the square built on the hypotenuse. To create the hypotenuse, and two triangles, do as follow:

5. With the compass, measure one side of square *BEFG* and report it at point *A*. Create a point, which you will name *H,* at the intersection of *AB* and the stroke of the compass.
6. Trace segments *DH* and *HF*.



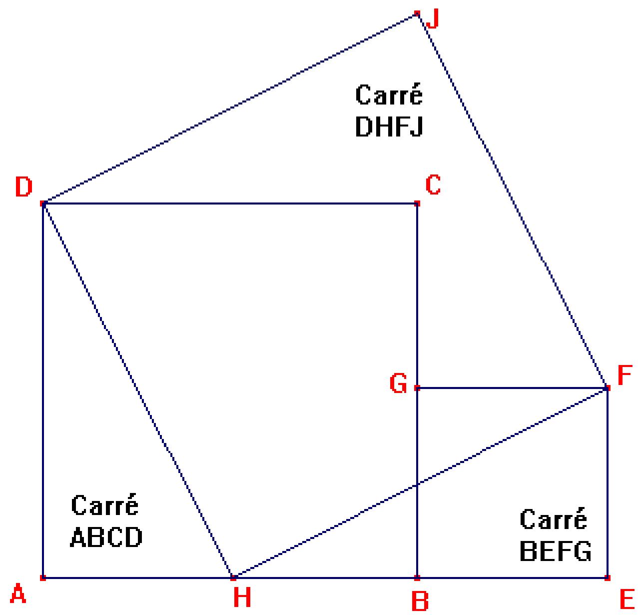
Third part:
<u>The square of the hypotenuse</u>

Construct the square with side *DH* (or *HF*).

7. Trace the perpendicular to *DH* through *D*.
8. Trace the perpendicular to *HF* through *F*.
9. Name the point of intersection of these two perpendiculars *J*.
10. You can now construct the square *DHFJ*.

This completes the construction of most of the elements needed for the animated proof.

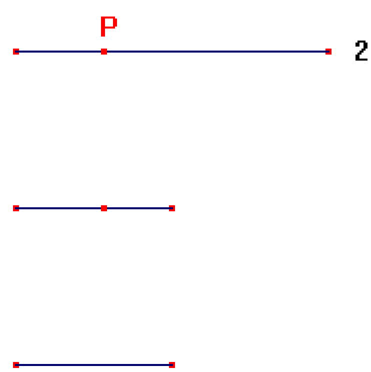
The proof proceeds by filing the square *DHFJ* with the squares *ABCD* and *BEFG*. To do so, we need to place triangles *AHD* and *HEF* in square *DHFJ*.



Fourth part:
<u>Beginning the animation</u>

We now start the animation. We are planning two steps.

11. Type in 2, as a number.
12. Construct a segment and, on that segment, a point *P*.
13. Apply the macro ***Polyconstruction,*** to *P*, its segment and the number 2.

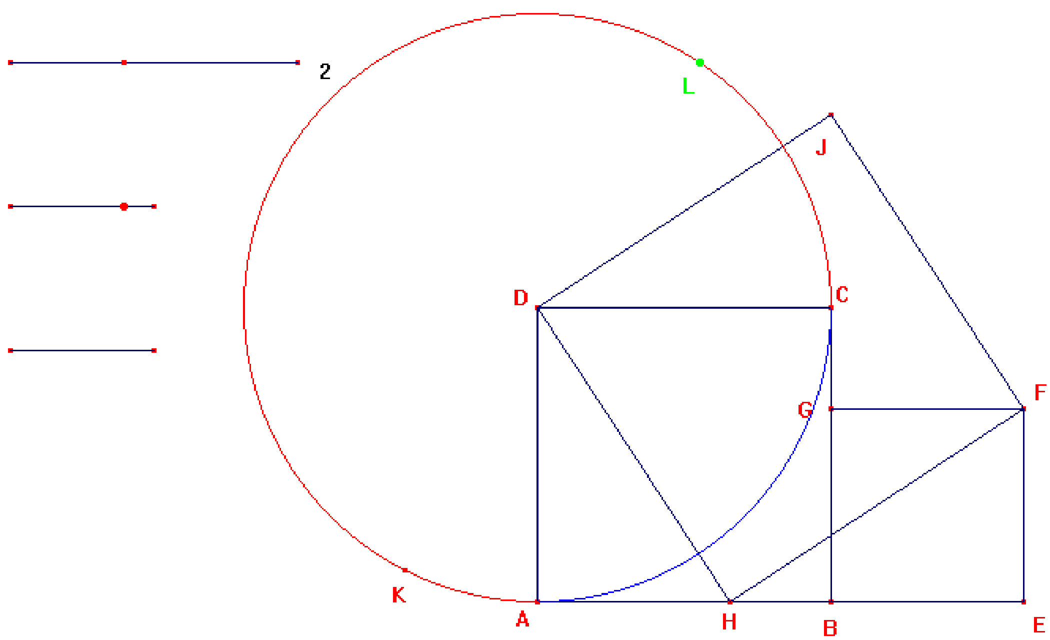


This will give two segments and, on each of these segments, a point will exist in succession according to the position of *P*. This enables the creation of an animation in two steps. Move point *P* in such a way that a (moving) point *P1* appears on the first small segment. It is not necessary to give a name (*P1*) to this point: we do so to make the explanations easier to follow.

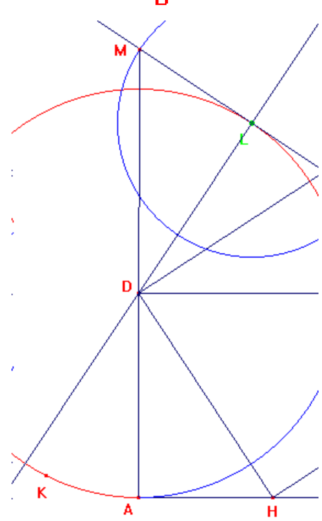
Fifth part:
Rotation of triangle *AHD*

To fill in square *DHFJ*, we will first make a rotation of triangle *AHD* around its vertex *D*. Note that we choose the rotation to be done clockwise so that the rotating triangle does not hide the rest of the figure. Make sure to move point *P* in such a way that a (moving) point *P1* appears on the first small segment.

14. The macro ***Point de rotation*** uses a circular arc which we must now construct. Trace a circle with centre *D* and radius *DA*.
15. Place a point *K* on this circle between *A* and *C* (clockwise). Construct the arc *AKC*.
16. Use the macro ***Point de rotation*** to construct the rotation of point *A*: apply the macro to *P1*, its segment and the arc *AKC*. Name the new point *L*. It will be on the arc.



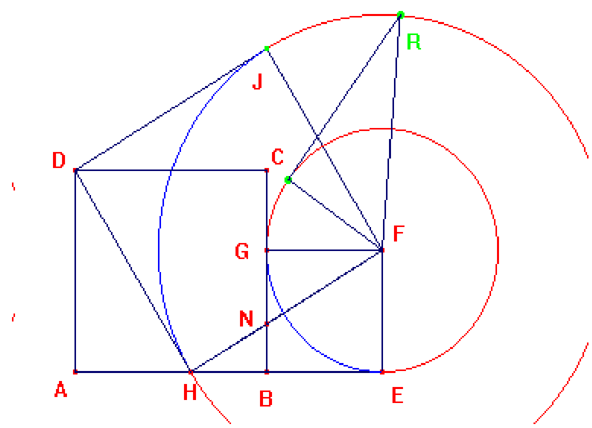
17. Trace the line *DL*.
18. Trace the perpendicular to this line through L.
19. Using the compass, report the distance *AH* from point *L*.
20. At the intersection of the compass stroke and the perpendicular to *DL*, construct the point *M*.
21. Construct triangle *DLM*.



Sixth part:
<u>Rotation of triangle *FHE*</u>

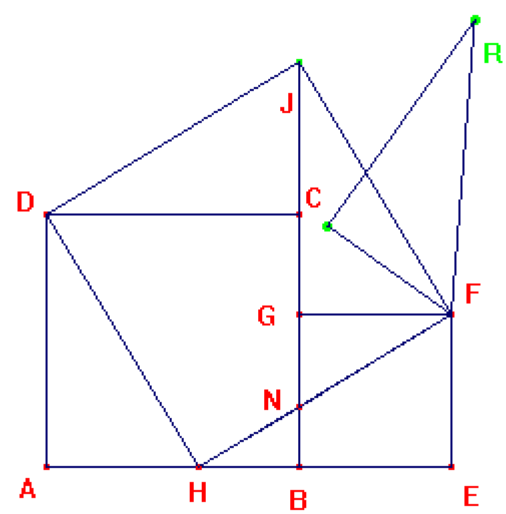
In this second step, we will rotate triangle *FHE* until it fills in what is left of square *DHFJ*. Make sure to move point *P* in such a way that a (moving) point *P2* appears on the second small segment.

22. Name *N* the point of intersection of *GB* and *FH*.
23. Construct the circle with centre *F* and radius *FH*. Construct the arc *HJ* (counter-clockwise).
24. Use macro ***Point de rotation*** to construct the rotation of point *H*: apply this macro to *P2*, its segment and the arc *HJ*. Name the new point *R*.
25. Construct a triangle congruent to *HEF* with *RF* as a side. This can be achieved using ***Point de rotation*** with point *E*, or otherwise. Let *KRF* denote this triangle.



Seventh part:
<u>Triangle *AHD*</u>

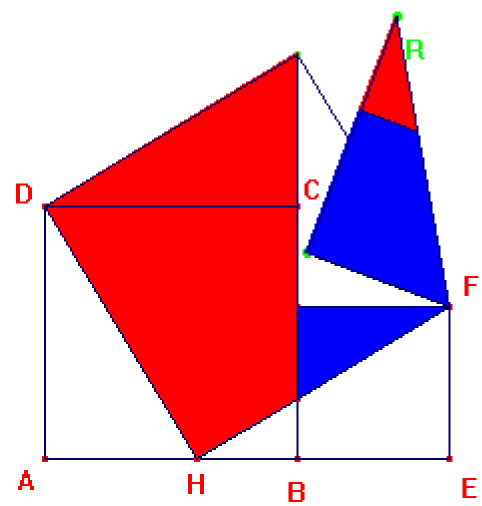
It would be useful if triangle *DCJ* (obtained after rotation) remained visible during this second step. Thus it must be constructed conditionally to point *P2*.



26. Use macro ***Point conditionnel*** (applied to *P2* and *J*) to obtain a point *J2* superposed to *J* and which exist only when *P2* exists.
27. Construct triangle *DCJ2* using point *J2* as a vertex.

Eighth part:
<u>Colours</u>

We can fill in the polygons so that they look better. To keep track of the two initial squares, we can fill in the polygons which are in square *ABCD* in red, and those in square *BEFG* in blue.



28. Using macro ***Point conditionnel***, construct in step 1 (i.e. conditionally to point *P1*) a triangle superposed to *NHB,* in red. Do the same for polygon *FNBE* in blue.
29. Colour triangle *DLM* of the first step in red.
30. In the second step, use the compass to mark length *HB* from vertex *R*. Trace a perpendicular to *KR* (see step 25 for point *K*) at the intersection between the circle (obtained from the compass tool) and *FR*. This yields the two points which are the images of *B* and *N* during the rotation (*B'* and *N'*). Construct and then colour triangle *RB'N'* in red and polygon *FN'B'E'* in blue.
31. Colour triangle *DJC* constructed in the seventh part in red.
32. Finally, construct and colour polygon *DHNC* in red and triangle *NFG* in blue. These should always remain visible, and thus no macro is used here.

Ninth part:
<u>Explanations</u>

To insert a text which can be different at each step, do as follow:

33. Place an anchor point *Q* where you would like the text to appear.
34. For each step, use macro ***Point conditionnel*** applied to the appropriate point (*P1* or *P2*) and *Q.*
35. Name the new point created superposed to *Q* with the explanations for that step.

Tenth part:
<u>Presentation</u>

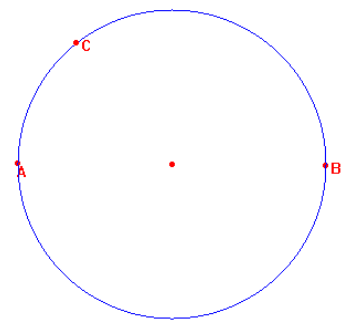
It is now time to hide what needs to be, including the segments corresponding to each step. Make sure that you hide objects and points for each of the two steps by moving point P. If the file will be handled by students, the figure can be made tougher to modify by fixing points; hiding points; and creating points superposed to other points, and then hiding the initial points, eventually naming the new points (which are fixed).

The animated construction will not work if square *ABCD* is larger than square *BEFG.* This can be arranged by going through the construction while starting from a smaller square *ABCD.*

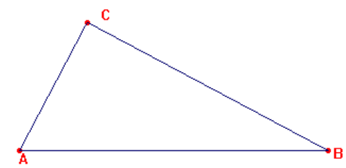
Document 2 – Second construction of an animated proof of Pythagoras' Theorem

Follow these steps to construct an animated proof of Pythagoras' Theorem similar to Pyth-B.fig

1. First part: the initial elements

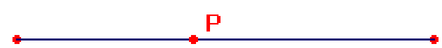


Firstly, we construct the elements which will remain present for the whole animation. In the case at hand, we need a right-angled triangle *ABC*, with hypotenuse *AB*. There are many ways to construct such a triangle. One way is to construct two points *A* and *B*, and then a circle centred at the midpoint of *AB* which passes through *A* (and *B*). Point *C* can then be chosen as any other point on the circle.



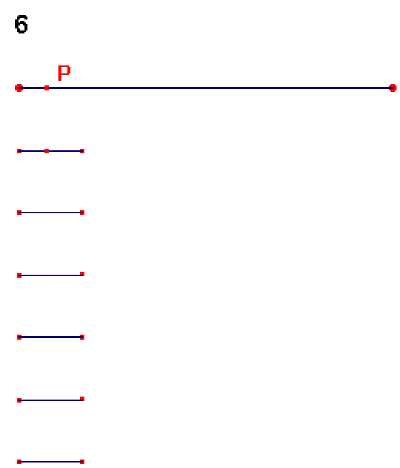
Triangle *ABC* can now be constructed. We then hide the circle and its centre.

2. Beginning the animation.



Construct a segment and a point *P* on this segment: *P* is the cursor, the point which will control the animation.

3. Number of steps of the animation.

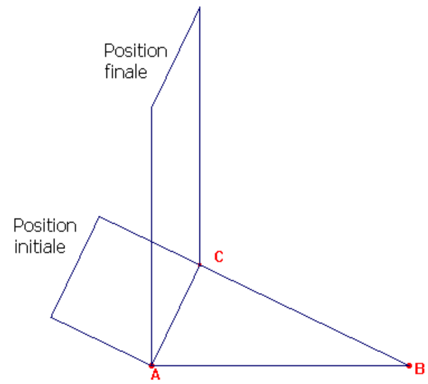


To start constructing the steps of the animation, we must first determine the number of steps. We are planning 6 steps. Type in the number 6 and then use macro ***Polyconstruction*** applied to the segment, point *P* and the number 6. This will give six segments and, on each of this segment, a point will exist in succession according to the position of *P*. This enables the creation of an animation in six steps.

Move point *P* in such a way that a (moving) point *P1* appears on the first small segment. It is not necessary to give a name (*P1*) to this point: we do so to make the explanations easier to follow.

4. **Animation – first step**: a square built on AC must become a parallelogram as it slides to reach the final position (see fig.).

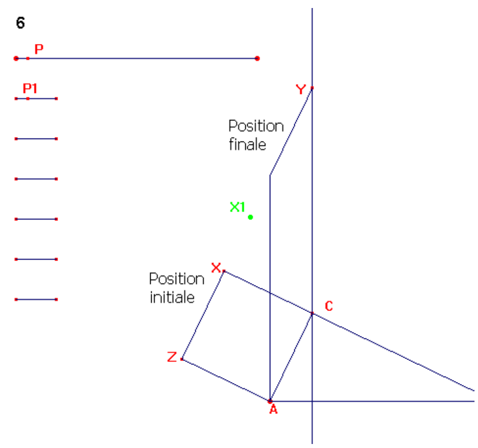
   To do so, we must first construct a square on AC and the perpendicular to AB through C.



5. Consider points X and Y on the figure. Y is the intersection of the perpendicular to AB through C and line ZX.

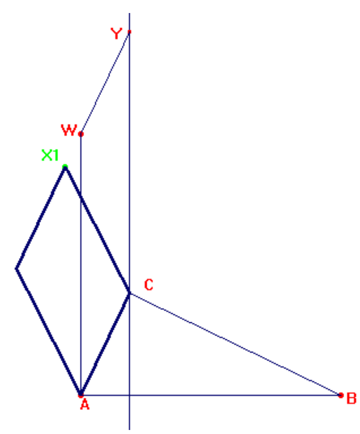
   A point $X1$, one of the vertices of the parallelogram, should move from X to Y: to construct $X1$, apply macro ***Point de translation*** to $P1$, the segment to which it belongs, X and Y.

   Explanation: as P moves, so does $P1$, and as $P1$ moves, so does $X1$. When $P1$ no longer exists (for instance, during the second part of the animation), $X1$ doesn't exist either.



6. Construct the parallelogram determined by the three vertices A, C and $X1$ (which can be done with two stroke of the compass). Let us note that when $P1$ ceases to exist, the parallelogram also ceases to exist.

   It is now possible to hide the elements which are no longer useful. However, some of these elements may turn out to be useful at a later stage and it may be preferable to wait if one is unsure. For instance, point W (see fig.) will be useful: it is the point of intersection of line ZX with the perpendicular to AB through A.

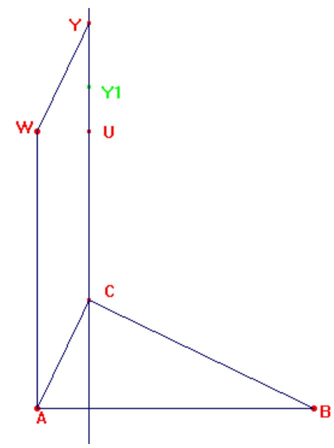


7. **Animation – second step.**

   Two vertices of the parallelogram (*Y* and *C*) must slide until we have a rectangle with one side on *AB*.

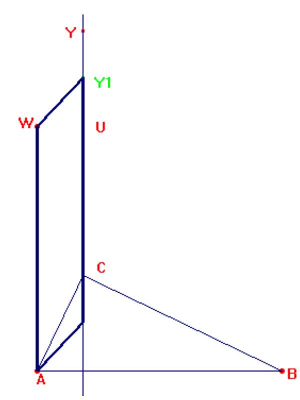
   Move point *P* in such a way that a (moving) point *P2* appears on the second small segment. Construct point *U* (*UW* is parallel to *AB*). Then, using macro ***Point de translation*** applied to *P2*, the segment to which it belongs, *Y* and *U*), we construct a point *Y1* that will move from *Y* to *U* as *P2* moves from one extremity of the segment to the other.

   As *P* moves, so does *P2*, and as *P2* moves, so does *Y1*. When *P2* no longer exists (for instance, during the second part of the animation), *Y1* doesn't exist either.



8. Construct the parallelogram determined by the three vertices *A*, *W* and *Y1*.

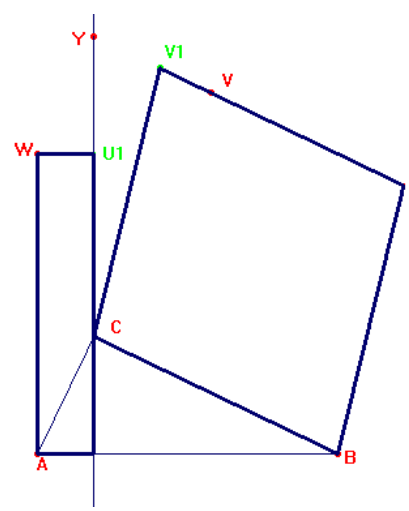
   This completes the second step of the animation. Hide what needs to be.



9. **Animation – third step.**

   Move point *P* in such a way that a (moving) point *P3* appears on the third small segment. In a way similar to that used for the square built on side *AC,* construct a square and a (moving) parallelogram on side *BC*. One vertex of the parallelogram, call it *V1,* will move from *V* to *Y* (where *V* is one of the vertex of the square built on *BC*).

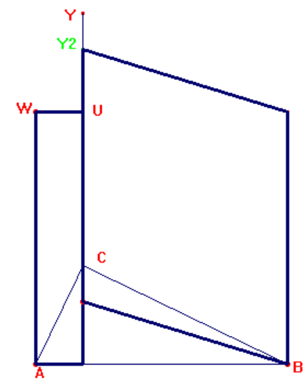
   The rectangle (determined by *W*, *U* and *A*) obtained at the end of the previous step should appear during this third step. Thus it must be constructed conditionally to point *P3*. To do so, use macro ***Point conditionnel***



applied to *P3* and *U*. This will produce a new point *U1* which will exist and be superposed to *U* as long as *P3* exists. Now construct the rectangle with *U1* as a vertex.

10. **Fourth step.** Move point *P* in such a way that a (moving) point *P4* appears on the fourth small segment. Repeat the construction done in the second step for the appropriate parallelogram: on the figure, *Y*2 moves from *Y* to U.

    The rectangle determined by *W*, *U* and *A* should appear during this fourth step. Thus it must be constructed conditionally to point *P4*.

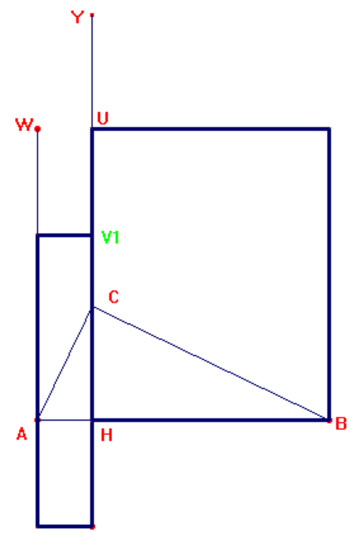


11. **Fifth step.** Move point *P* in such a way that a (moving) point *P5* appears on the fifth small segment.

    Consider *U* and *H*. Point V1 will need to move from *U* to *H*: to construct *V1,* apply macro ***Point de translation*** to P5, its segment, *U* and *H*.

    Construct a rectangle congruent to *UHAW,* with one vertex being *V1* (see figure). There are many ways to achieve this.

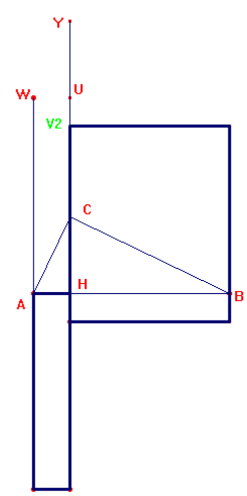
    The rectangle determined by *U*, *H* and *B* should appear during this step. Thus it must be constructed conditionally to point *P5*.



12. **Sixth step.** Move point *P* in such a way that a (moving) point *P6* appears on the sixth small segment.

    Repeat the construction done in the previous step in order for the rectangle on the right to move, conditionally to *P6*. Point *V2* will move from *U* to *H*.

    The rectangle obtained at the end of the last step should remain present during this step. Thus it must be constructed conditionally to point *P6*.

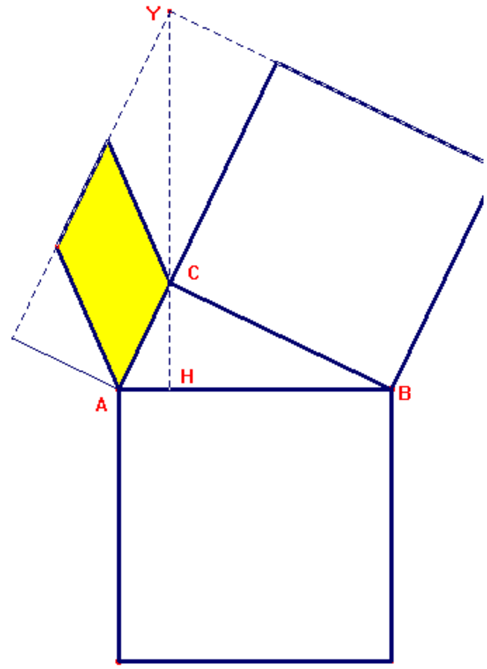


13. Colours and fixed elements. We can add elements which should be present during the whole animation: here, the three squares built on the three sides of the triangle should be added.

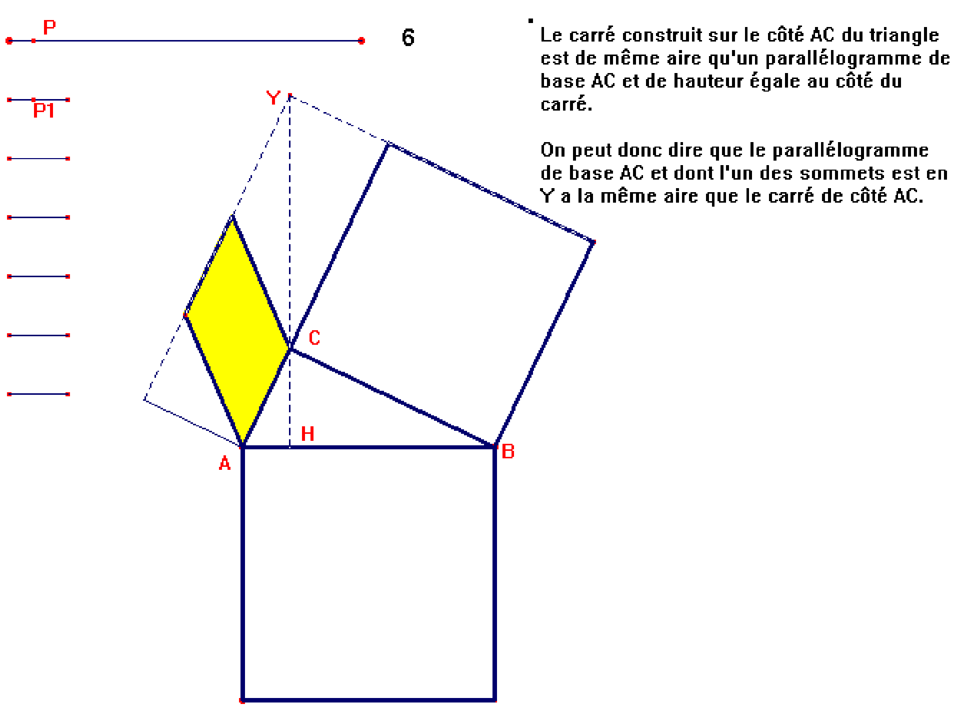
It is also useful to fill some polygons with colours. This must be done at each step, moving P to ensure that all the steps are covered.

Note

Colours may hide some important geometrical features and should be used with this in mind.



14. Explanatory text. It is possible to write an accompanying text which will change at each step of the animation. To do so, create an anchor point *Q* where the text will appear. Then, create a conditional point superposed to *Q* for each of the six steps (for instance, for step 1, apply macro ***Point conditionnel*** to *P1* and *Q*) and label this conditional point with the explanatory text. Once this is done, hide the anchor point *Q*.



15. Final clean up and resistance. It is now time to hide what needs to be, including the six segments corresponding to the six steps. Make sure that you hide objects and points for each of the six steps by moving point P. If the file will be handled by students, the figure can be made tougher to modify by fixing points; hiding points; and creating points superposed to other points, and then hiding the initial points, eventually naming the new points (which are fixed).