```
# This is the script "guion"
# Note: v23 is the vector from V2 to V3, v34 the vector from V3 to V4, etc.

# 1. UPDATE POSITION OF THE ADJACENT VERTICES TO V1.

SetValue(V2, Intersect(Ray(V1,V2), Sphere(V1,1)))
SetValue(V3, Intersect(Ray(V1,V3), Sphere(V1,1)))
SetValue(V4, Intersect(Ray(V1,V4), Sphere(V1,1)))
UpdateConstruction()


# 2. TRANSMIT THE POSITIONS TO THE FOLLOWING THREE VERTICES.

# 2.1 Most of the time, V5, V6 and V7 will occupy the positions P1, P2, P3 which are calculated below:

SetValue(C, (V3+V2)/2)
SetValue(Q1, Intersect(Line(C, v23), PerpendicularPlane(V5, v23)))
SetValue(P1, Intersect(Ray(C, C+V5-Q1), Circle(C, abs(C-V1), v23)))

SetValue(C, (V4+V3)/2)
SetValue(Q1, Intersect(Line(C, v34), PerpendicularPlane(V6, v34)))
SetValue(P2, Intersect(Ray(C, C+V6-Q1), Circle(C, abs(C-V1), v34)))

SetValue(C, (V2+V4)/2)
SetValue(Q1, Intersect(Line(C, v42), PerpendicularPlane(V7, v42)))
SetValue(P3, Intersect(Ray(C, C+V7-Q1), Circle(C, abs(C-V1), v42)))

# 2.2 We look at the special cases of vertex coincidence:

SetValue(P1, If(V2==V3, Intersect(Ray(V2,V5), Sphere(V2,1)), P1))
SetValue(P2, If(V3==V4, Intersect(Ray(V3,V6), Sphere(V3,1)), P2))
SetValue(P3, If(V4==V2, Intersect(Ray(V4,V7), Sphere(V4,1)), P3))

# 2.3 If the circumradius r of triangle P1 P2 P3 is greater than 1, the positions of P1, P2 and P3 must be
readjusted:

SetValue(C, Centre(Circle(P1,P2,P3)))
SetValue(r, If(P1==P2, abs(P1-P3)/2, P2==P3, abs(P2-P1)/2, P3==P1, abs(P3-P2)/2, abs(C-P1)))
SetValue(C, Centre(Circle(V2,V3,V4)))
SetValue(P', Intersect(Ray(C,2C-V1), Sphere(C,1)))

# 2.3.1 Of the two intersection points, choose as P1 the one closest to V5:

SetValue(C, (V3+V2)/2)
SetValue(pc, {Intersect(Sphere(P',1), Circle(C, abs(C-V1), v23))})
```

```
SetValue(Q1, pc(1))
SetValue(Q2, pc(2))
SetValue(Q1, If(abs(P1-Q2)<abs(P1-Q1), Q2, Q1))
SetValue(V5, If(r > 1, Q1, P1))

# 2.3.2 Of the two intersection points, choose as P2 the one closest to V6:

SetValue(C, (V4+V3)/2)
SetValue(pc, {Intersect(Sphere(P',1), Circle(C, abs(C-V1), v34))})
SetValue(Q1, pc(1))
SetValue(Q2, pc(2))
SetValue(Q1, If(abs(P2-Q2)<abs(P2-Q1), Q2, Q1))
SetValue(V6, If(r > 1, Q1, P2))

# 2.3.3 Of the two intersection points, choose as P3 the one closest to V7:

SetValue(C, (V2+V4)/2)
SetValue(pc, {Intersect(Sphere(P',1), Circle(C, abs(C-V1), v42))})
SetValue(Q1, pc(1))
SetValue(Q2, pc(2))
SetValue(Q1, If(abs(P3-Q2)<abs(P3-Q1), Q2, Q1))
SetValue(V7, If(r > 1, Q1, P3))


# 3. DETERMINES THE POSITION OF V8 (THE OPPOSITE VERTEX OF V1).

# 3.1 First case. V5, V6 and V7 different (if they are different they cannot be aligned):

SetValue(C, Centre(Circle(V5,V6,V7)))
SetValue(v, PerpendicularVector(Plane(V5,V6,V7)))
SetValue(pc, {Intersect(Line(C, v), Sphere(V5,1))})
SetValue(Q1, pc(1))
SetValue(Q2, pc(2))
SetValue(Q1, If(abs(V8-Q2)<abs(V8-Q1), Q2, Q1))
SetValue(V8, If(V5!=V6 && V5!=V7 && V7!=V6, Q1, V8))

# 3.2 Second case. V5 matches V6 but not V7:

SetValue(C, (V5+V7)/2)
SetValue(Q1, Intersect(Line(C, v57), PerpendicularPlane(V8, v57)))
SetValue(P1, Intersect(Ray(C, C+V8-Q1), Circle(C, abs(C-V2), v57)))
SetValue(V8, If(V5==V6 && V5!=V7, P1, V8))

# 3.3 Third case. V7 matches V5 but not V6, or V7 matches V6 but not V5:
```

```
SetValue(C, (V5+V6)/2)
SetValue(Q1, Intersect(Line(C, v56), PerpendicularPlane(V8, v56)))
SetValue(P1, Intersect(Ray(C, C+V8-Q1), Circle(C, abs(C-V3), v56)))
SetValue(V8, If((V7==V5 || V7==V6) && V5!=V6, P1, V8))
```

3.4 Fourth case. Match V5, V6 and V7:

```
SetValue(V8, If(V5==V6 && V5==V7, Intersect(Ray(V5,V8), Sphere(V5,1)), V8))
UpdateConstruction()
```