La note technique sur AutoCAD P&ID et Plant 3D

34

20.02.2022

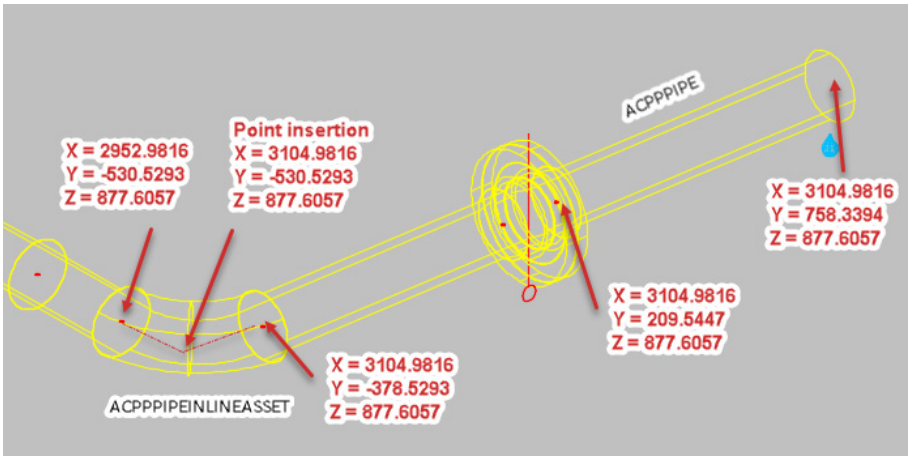
www.dovaq.fr

Un peu de programmation AutoLISP sur des objets PLANT 3D

Bien que le langage AutoLISP ne soit pas le langage idéal pour aborder des objets Plant 3D, il n'en reste pas moins qu'il permet de récupérer certaines informations mémorisées dans ces types d'objet.

Pour cela, il faut interroger l'objet Plant 3D pour entrer dans ses informations.

Prenons l'exemple d'un tuyau et d'un coude pour retrouver leurs points de coordonnées.



Démarrons par les informations liées au tuyau.

Par cette commande, on va pouvoir récupérer ses données primaires :

(Setq SELECT_OBJ (Entget (Car (Entsel))))

Choix de l'objet: ((-1 . <Nom d'entité: 408ecc80>) (0 . "ACPPPIPE") (5 . "6A0") (102 . "{ACAD_REACTORS") (330 . <Nom d'entité: 408eccb0>) (102 . "}") (330 . <Nom d'entité: 392c29f0>) (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (62 . 2) (100 . "AcPpDb3dPart") (90 . 65794) (90 . 0) (290 . 0) (290 . 0) (10 0.0 0.0 0.0) (100 . "AcPpDb3dSegment") (100 . "AcPpDb3dPipe") (10 3104.98 758.339 877.606) (11 3104.98 209.545 877.606) (41 . 57.15) (42 . 0.0) (290 . 0) (340 . <Nom d'entité: 0>) (46 . 0.0) (47 . 0.0))

On retrouve des clés DXF qui indiquent des "propriétés", telles que le calque : clé "DXF 8" (8 . "0"), le type d'objet : clé "DXF 0" (0 . "ACPPPIPE"), etc.

Les clés DXF 10 et 11 correspondent à des coordonnées. Ici on retrouve deux fois la clé "DXF 10" (10 0.0 0.0 0.0) ") (10 3104.98 758.339 877.606). En fait, pour cet objet-là, le tuyau, les coordonnées sont notées par les clés "DXF 10" et "DXF 11" placées à la fin de la liste de toutes les clés.

AutoCAD Plant 3D et AutoCAD P&ID sont des marques déposées par Autodesk - Tous les produits cités peuvent être des marques déposées par leurs propriétaires respectifs.

Ces deux clés DXF peuvent être récupérées comme ceci :

(Setq PT1 (cdr (assoc 10 (reverse SELECT_OBJ))) ; Point de départ => (3104.98 758.339 877.606)
PT2 (cdr (assoc 11 (reverse SELECT_OBJ)))); Point de fin => (3104.98 209.545 877.606)

Le point de départ, ici nommé PT1, a donc comme coordonnée (X,Y,Z) 3104.98 758.339 877.606
Le point de fin, ici nommé PT2, a donc comme coordonnée (X,Y,Z) 3104.98 209.545 877.606

La fonction "REVERSE" inverse le sens de recherche des codes DFX de l'objet dans la liste mémorisée en "SELECT_OBJ".

La fonction "ASSOC" récupère la valeur d'une clé sous la forme de liste, par exemple "(10 3104.98 758.339 877.606)".

La fonction "CDR" extrait la valeur de la clé, par exemple (CDR "(10 3104.98 758.339 877.606)) retourne "(3104.98 758.339 877.606)".

Pour les autres types d'objets Plant 3D, par exemple un coude, une bride, un té, une vanne, les propriétés sont disposées différemment que celles des tuyaux.

Prenons l'exemple d'un coude.

(Setq SELECT_OBJ (Entget (Car (Entsel))))

Choix de l'objet: ((-1 . <Nom d'entité: 4eb143d0>) (0 . "ACPPPIPEINLINEASSET") (5 . "6A5") (102 . "{ACAD_REACTORS") (330 . <Nom d'entité: 4eb14400>) (102 . "}") (330 . <Nom d'entité: 32e611f0>) (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "Lignex") (62 . 2) (100 . "AcPpDb3dPart") (90 . 66050) (90 . 0) (290 . 0) (290 . 0) (10 0.0 0.0 0.0) (100 . "AcPpDb3dInlineAsset") (340 . <Nom d'entité: 7879820>) (40 . 0.0) (10 3104.98 -530.529 877.606) (210 0.0 1.0 0.0) (211 0.0 0.0 1.0) (90 . 0) (100 . "AcPpDb3dPipeInlineAsset"))

On a bien, depuis la fin de la liste, la clé "DXF 10" (10 3104.98 -530.529 877.606), mais point la clé "DXF 11". On retrouve les clés primaires, telles que le calque "DXF 8", le type d'objets "DXF 0", etc.

Cette clé "DXF 10" nous indique le point d'insertion de l'objet Plant 3D.

(Setq PT1 (cdr (assoc 10 (reverse SELECT_OBJ))))

Les points des ports de connexion ne se trouvent pas dans cette liste primaire, mais plus loin.

En fait, les informations sont mémorisées à partir d'un autre pointeur, défini par la clé "DXF 330" en première position de la liste.

(setq OBJ1 (entget (cdr (assoc 330 SELECT_OBJ))))

((-1 . <Nom d'entité: 4eb14400>) (0 . "ACDBASSOCDEPENDENCY") (330 . <Nom d'entité: 4eb155f0>) (5 . "6A8") (100 . "AcDbAssocDependency") (90 . 2) (90 . 0) (290 . 1) (290 . 1) (290 . 1) (290 . 1) (90 . 0) (330 . <Nom d'entité: 4eb143d0>) (290 . 0) (330 . <Nom d'entité: 0>) (330 . <Nom d'entité: 0>) (360 . <Nom d'entité: 4eb14410>) (90 . 13))

Dans cette liste, on retrouve d'autres pointeurs, entre autres celui dans la clé "DXF 360" qui va nous retourner les informations que l'on cherche, à savoir les clés "DXF 10" des différents ports de connexion de l'objet Plant 3D.

(setq OBJ_INFO (entget (cdr (assoc 360 OBJ1))))

www.dovaq.fr - contact@dominique-vaquand.com
Votre partenaire AutoCAD depuis plus de 25 ans

((-1 . <Nom d'entité: 4eb14410>) (0 . "ACPPPIPEDEPENDENCYBODY") (330 . <Nom d'entité: 4eb14400>) (5 . "6A9") (100 . "AcDbAssocDependencyBody") (90 . 1) (100 . "AcPpDb3dDependencyBody") (90 . 2) (90 . 2) (90 . 2) (1 . "S1") (10 3104.98 -378.529 877.606) (11 0.0 1.0 0.0) (290 . 0) (40 . 0.0) (40 . 0.0) (1 . "in") (2 . "") (290 . 0) (90 . 2) (1 . "S2") (10 2952.98 -530.529 877.606) (11 -1.0 6.12323e-17 0.0) (290 . 0) (40 . 0.0) (40 . 0.0) (1 . "in") (2 . "") (290 . 0) (90 . 2) (90 . 13907) (5 . "6A5") (90 . 0) (330 . <Nom d'entité: 4eb143d0>) (90 . 2) (1 . "S1") (10 3104.98 -378.529 877.606) (11 0.0 1.0 0.0) (290 . 0) (40 . 0.0) (40 . 0.0) (1 . "in") (2 . "") (290 . 0) (90 . 13907) (5 . "6AA") (90 . 0) (330 . <Nom d'entité: 4eb14420>) (90 . 2) (1 . "S1") (10 3104.98 -378.529 877.606) (11 0.0 -1.0 0.0) (290 . 0) (40 . 0.0) (40 . 0.0) (1 . "in") (2 . "") (290 . 0) (90 . 13907) (5 . "6A5") (90 . 0) (330 . <Nom d'entité: 4eb143d0>) (90 . 2) (1 . "S2") (10 2952.98 -530.529 877.606) (11 -1.0 6.12323e-17 0.0) (290 . 0) (40 . 0.0) (40 . 0.0) (1 . "in") (2 . "") (290 . 0) (90 . 13907) (5 . "6AD") (90 . 0) (330 . <Nom d'entité: 4eb14450>) (90 . 2) (1 . "S1") (10 2952.98 -530.529 877.606) (11 1.0 -6.12323e-17 0.0) (290 . 0) (40 . 0.0) (40 . 0.0) (1 . "in") (2 . "") (290 . 0))

En fait, dans cette catégorie de type d'objet Plant 3D, les coordonnées sont décrites selon les ports de connexion, à savoir ici "S1", "S2, et aussi "S3", "S4" s'il possède quatre ports.

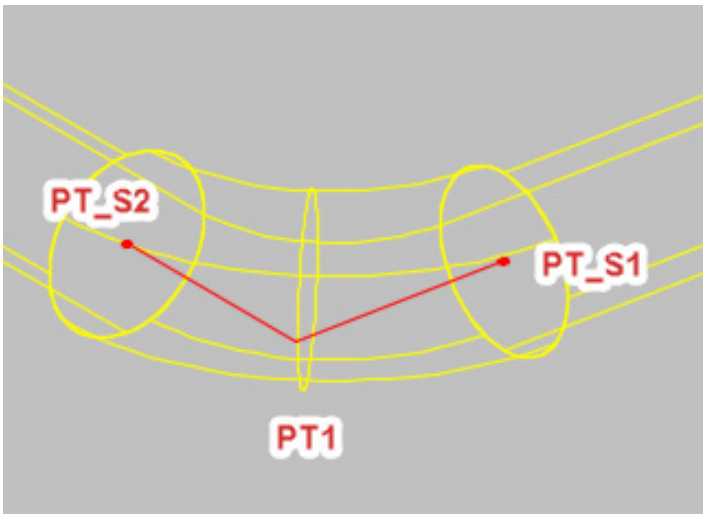
Notre coude n'en possédant que deux, on trouvera donc les ports "S1" et "S2"

Dans ce cas, il faut localiser la position d'une clé "DXF 1" qui définit un port pour retrouver de suite après la clé "DXF 10" de sa coordonnée finale.

```
(Setq PT_S1 (cdr (cadr (member (cons 1 "S1") obj_info)))) ; => (3104.98 -378.529 877.606)
(Setq PT_S2 (cdr (cadr (member (cons 1 "S2") obj_info)))) ; => (2952.98 -530.529 877.606)
```

Avec ces trois coordonnées "PT1" "PT_S1" et "PT_S2", on peut tracer, si on le souhaite, l'axe du coude par la commande :

```
(Command "_Line" PT_S1 PT1 PT_S2 "")
```



www.dovaq.fr - contact@dominique-vaquand.com
Votre partenaire AutoCAD depuis plus de 25 ans

Dans le langage AutoLisp, il est inclus des fonctions "VisualLisp". En utilisant certaines de ces fonctions, on peut récupérer d'autres informations plus pertinentes qu'en AutoLISP basique sur ces objets-là.

Par exemple :

```
(VL-LOAD-COM)
 (Setq SELECT_OBJ_VL (Vlax-Ename->Vla-Object (Car (entsel))))

(Vlax-Dump-Object SELECT_OBJ_VL)
```

Cette fonction renvoie les informations suivantes :

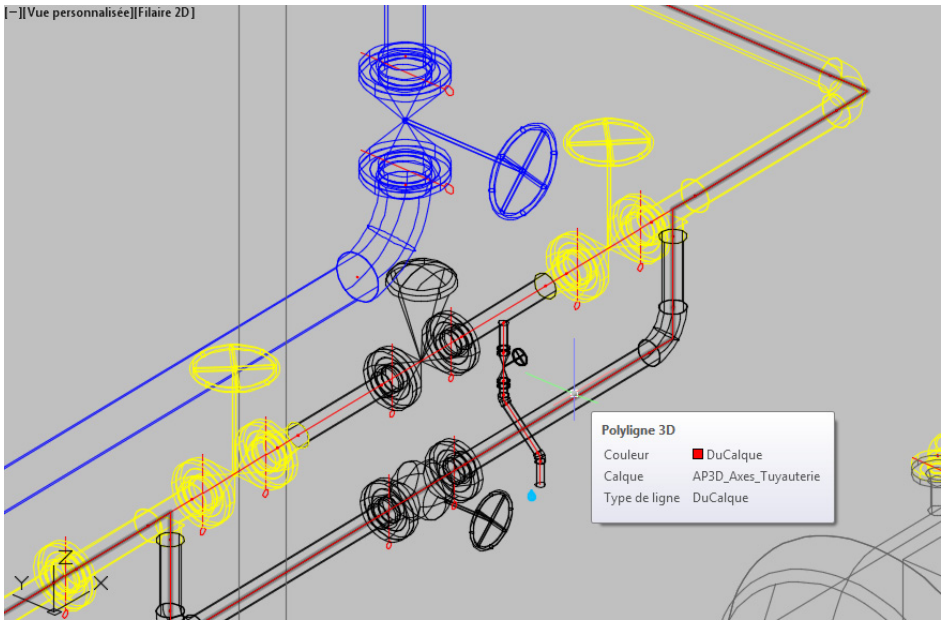
```
; IAcPnP3dPipe: Interface IAcPnP3dPipe
; Valeurs de propriétés:
;   Application (RO) = #<VLA-OBJECT IAcadApplication 000000014020d990>
;   CutLength (RO) = "548.794731"
;   Document (RO) = #<VLA-OBJECT IAcadDocument 00000000408716d8>
;   EndType (RO) = "PL"
;   EntityTransparency = "DuCalque"
;   Facing (RO) = ""
;   Handle (RO) = "6A0"
;   HasExtensionDictionary (RO) = 0
;   Hyperlinks (RO) = #<VLA-OBJECT IAcadHyperlinks 0000000058211a58>
;   InsulationThickness (RO) = ""
;   InsulationType (RO) = ""
;   Layer = "0"
;   LineNumberTag (RO) = "?"
;   Linetype = "ByLayer"
;   LinetypeScale = 1.0
;   Lineweight = -1
;   Material = "ByLayer"
;   MaterialCode (RO) = "1.0037"
;   ObjectID (RO) = 42
;   ObjectName (RO) = "AcPpDb3dPipe"
;   OwnerID (RO) = 43
;   PartFamilyLongDesc (RO) = "Pipe DIN 2448"
;   PlotStyleName = "Couleur_ 2"
;   PnPClassName (RO) = "Pipe"
;   PnPGuid (RO) = "d8347f14-d489-4909-9d8d-88888aeb4d53"
;   PnPMaterial (RO) = ""
;   PressureClass (RO) = ""
;   Schedule (RO) = ""
;   Service (RO) = ""
;   Size (RO) = "4\""
;   Spec (RO) = "10HC01"
;   Status (RO) = "New"
;   Tag (RO) = "4\"-10HC01-?-?"
;   TieInNumber (RO) = ""
;   TrueColor = #<VLA-OBJECT IAcadAcCmColor 0000000058211d50>
;   Visible = -1
;   WallThickness (RO) = "3.6"
```

Dans cette liste de propriétés, on retrouve pas mal d'informations intéressantes telle que l'étiquette (Tag), le diamètre (Size), le service (Service), le statut (Status), etc.

Pour récupérer une de ces propriétés, nous allons nous servir de la fonction "VLAX-GET-PROPERTY". Par exemple pour récupérer l'étiquette (Tag) :

```
(Setq ETIQ (Vlax-Get-Property SELECT_OBJ_VL "TAG"))
"4\"-10HC01-?-?"
```

Pour illustrer ce numéro, voici un exemple d'un programme qui trace une ligne d'axe sur des conduites de tuyauteries.




```
;|
Dominique VAQUAND
20.02.2020
=====================================================
Trace un trait d'axe sur des conduites de tuyauteries
=====================================================
|;


(defun c:AP3D_AxesT (/ Selection  CalqueCourant
                       Osmode  OBJ        OBJ_1    OBJ_2
                       X_LignesAxes       PT_10    Index_S
                       Index      PT_S1   PT_S2    PT_S3
                    )

(setvar "cmdecho" 0)

; Sélection des objets PLANT 3D
  (setq Selection (ssget (list (cons 0 "ACPPP*,ACPPC*"))))

  (setq Index_S 0)

; Si des objets sont sélectionnés

  (if Selection

    (progn

; Mémorisation des paramètres de départ

      (setq CalqueCourant (getvar "clayer"))
      (setq Osmode (getvar "osmode"))
      (setvar "osmode" 0)
      (command "_ucs" "")

; Gestiion Erreur

      (defun *error* (s)
      (setvar "osmode" osmode)
      (setvar "clayer" CalqueCourant)
      (command "_UCS" "_p")
      (setq *error* nil)
```

```
      )

; Création du calque AP3D_Axes_Tuyauterie s'il n'existe pas ou met courant ce calque
      (if (= (tblsearch "layer" "AP3D_Axes_Tuyauterie"))
      (command "_layer" "_T" "AP3D_Axes_Tuyauterie" "_UN" "AP3D_Axes_Tuyauterie" "")
      )

      (command "_layer" "_m" "AP3D_Axes_Tuyauterie" "_co" "1" "" "" )

; Boucle sur le nombre d'objet PLANT 3D sélectionnés

      (repeat (sslength Selection)

      ; récupération des codes DXF primaires

      (setq OBJ (entget (ssname Selection Index_S)))

      ; Mémorisation de la coordonnée départ ou point d'insertion
      (setq PT_10 (cdr (assoc 10 (reverse OBJ))))

      ; Mémorisatiobn des pointeurs
      (setq OBJ_1 (entget (cdr (assoc 330 OBJ))))

      ; traitement pour tracer la ligne d'axe

      (TraiteObj obj_1 360 obj PT_10)


      (setq index_S (1+ index_S))

      )           ;repeat
    )
  ) ;if

  ; Les lignes d'axes ont été créées, on les joint ensemble pour créer des polylignes 2D ou 3D

  (setq         X_LignesAxes
      (ssget    "x"
                (list (cons 0 "LINE")
                      (cons 8 "AP3D_Axes_Tuyauterie")
                )
      )
  )
  (initcommandversion)
  (command "_join" X_LignesAxes "")

; Remise des paramètres de départ

  (setvar "osmode" osmode)
  (setvar "clayer" CalqueCourant)
  (command "_UCS" "_p")

  (princ)
)

;|
==========================
Création des lignes d'axe
==========================
|;

(defun TraiteObj (Obj1x    Codedxf  objx     PT_10x   /        PT_S1
                  PT_S2     PT_S3    PT_S4    PT_S5    PT_S6    index
                  OBJ_2
                 )


  ;; Tyuyau = (0 . "ACPPPIPE")
  ;; Composants = (0 . "ACPPPIPEINLINEASSET") ou (0 . "ACPPCONNECTOR")
```

```
(setq index 0)
(setq          PT_S1 nil
    PT_S2 nil
    PT_S3 nil
    PT_S4 nil
    PT_S5 nil
    PT_S6 nil
)

; si l'objet PLANT 3D est un tuyau on récupére les coordonnées des ports S1 et S2
; directement dals la liste des clés primaires

(if (= (cdr (assoc 0 objx)) "ACPPPIPE")
  (progn
    (setq pt_s1 (cdr (assoc 10 (reverse objx)))
        pt_s2 (cdr (assoc 11 (reverse objx)))
    )
    (if (> (distance pt_s1 pt_s2) 0)
    (command "_line" pt_s1 pt_s2 "")
    )
  )

; Sinon on recherche les ports S1, S2, ... depuis un nouveau pointeur
; une fois les ports S1, S2, ... trouvés on récupère la coordonnées du port
  (progn

    (setq OBJ_2 (entget (cdr (assoc Codedxf OBJ1x))))

; on boucle sur la liste pour récupérer la coordonnées du port

    (repeat (length OBJ_2)

    (if (= (cdr (nth index OBJ_2)) "S1")
      (if (= PT_S1 nil)
        (setq PT_S1 (cdr (nth (1+ index) OBJ_2)))
      )
    )

    (if (= (cdr (nth index OBJ_2)) "S2")
      (if (= PT_S2 nil)
        (setq PT_S2 (cdr (nth (1+ index) OBJ_2)))
      )
    )

    (if (= (cdr (nth index OBJ_2)) "S3")
      (if (= PT_S3 nil)
        (setq PT_S3 (cdr (nth (1+ index) OBJ_2)))
      )
    )

    (if (= (cdr (nth index OBJ_2)) "S4")
      (if (= PT_S4 nil)
        (setq PT_S4 (cdr (nth (1+ index) OBJ_2)))
      )
    )

    (if (= (cdr (nth index OBJ_2)) "S5")
      (if (= PT_S5 nil)
        (setq PT_S5 (cdr (nth (1+ index) OBJ_2)))
      )
    )

    (if (= (cdr (nth index OBJ_2)) "S6")
      (if (= PT_S6 nil)
        (setq PT_S6 (cdr (nth (1+ index) OBJ_2)))
      )
    )

    (setq index (1+ index))

    )           ;repeat
```

```
; Si la coordonnée du port existe alors on trace une ligne d'axe
; du point d'insertion ou point du port

    (if (and (/= pt_s1 nil) (> (distance pt_10 pt_s1) 0))
    (command "_line" pt_10 pt_s1 "")
    )

    (if (and (/= pt_s2 nil) (> (distance pt_10 pt_s2) 0))
    (command "_line" pt_10 pt_s2 "")
    )

    (if (and (/= pt_s3 nil) (> (distance pt_10 pt_s3) 0))
    (command "_line" pt_10 pt_s3 "")
    )

    (if (and (/= pt_s4 nil) (> (distance pt_10 pt_s4) 0))
    (command "_line" pt_10 pt_s4 "")
    )


    (if (and (/= pt_s5 nil) (> (distance pt_10 pt_s5) 0))
    (command "_line" pt_10 pt_s5 "")
    )

    (if (and (/= pt_s6 nil) (> (distance pt_10 pt_s6) 0))
    (command "_line" pt_10 pt_s6 "")
    )

   )
  ) ;if

(princ)

) ;defun


;|
Efface tous les objets placés sur le calque "AP3D_Axes_Tuyauterie"
|;

(defun c:AP3D_EffacerAxesT ()

  (command "_erase"
        (ssget "x" (list (cons 8 "AP3D_Axes_Tuyauterie")))
        ""
  )
  (princ)
)
```

Ce programme est sous la forme d'un fichier nommé "Ap3D_AxesT.lsp". Il contient deux commandes :

- **AP3D_AxesT** pour créer le trait d'axe en sélectionnant des objets Plant 3D.
  Le trait d'axe se place automatiquement sur le calque "AP3D_Axes_Tuyauterie"

- **AP3D_EffacerAxesT** pour effacer tous les objets contenus sur le calque "AP3D_Axes_Tuyauterie"

Ce programme n'est qu'un exemple de traitement directement effectuer à partir du langage AutoLISP. Il a pour objectif de vous montrer comment sont mémorisées des propriétés de PLANT 3D et comment les utiliser.
De par ce langage, il n'est pas possible de modifier des propriétés d'objets Plant 3D. Dans ce cas, il faudra alors travailler avec des langages DotNet, soit en C# ou en Visual Basic, soit en C++.

Notes :

Si vous désirez découvrir le langage AutoLISP pour créer vos propres applications, je vous invite à :

- Lire le numéro 53 des "Cahiers d'AutoCAD"
  https://dovaq.fr/download/633/

ou à visiter sur les sites suivants :

- Introduction à AutoLISP (par Gilles Chanteau)
  https://programmation.developpez.com/tutoriels/autolisp/introduction-autolisp/#LI

- DA-Code (par Didier Aveline)
  https://www.da-code.fr/