S8 ELC D-6
PLM - Maquette Numérique
Rapport

Jeu d'échec 3D avec CATIA et VBScript

Élèves :
Alexandre Teixeira
Deborah Attie

Enseignants :
Paul Clozel
Didier Lacour

7 avril 2017

Table des matières

Table des figures

1 Objectif

Dans le cadre du sujet PLM Maquette Numérique de l'École Centrale Lyon, nous sommes invités à réaliser un projet en utilisant différents outils disponibles sur CATIA. Le projet de ce rapport avait comme objectif la création d'un jeu d'échec en 3D en utilisant les outils de *Part Design* pour la création des pièces et VBA pour l'implémentation de la logique et organisation des informations du jeu ainsi que la programmation de l'interface pour le joueur. Étant donnée la simplicité de notre projet en termes de contraintes, nous avons décidé d'utiliser *Assembly Design* avec VBA. Pour comprendre le fonctionnement et le développement de ce projet, ce compte rendu contient les pièces créées et les logiques derrière le script VBA.

2 Installation

Pour jouer ce jeu il faut avoir CATIA V5 2014 ou plus récent. Pour commencer, téléchargez le jeu dans le lien suivant sur GitHub : `https://github.com/ateixeira0163/Jeu-d-Echec-avec-VBACatia`. Ensuite, on extrait les fichiers dans le répertoire souhaité. Il suffit d'ouvrir le fichier JeuDEchec.Product. Avec la fenêtre ouverte, on charge sur `Outils → Macros` la Macro `JeuDeEchec.vbcatia` et on exécute la Macro `JeuDeEchec`.

3 Création des pièces

Pour la fabrication des pièces nous avons utilisé les outils de Part Design. Le jeu est composé d'un tableau et de six pièces différentes. Les commandes basiques que nous avons utilisés ont été la poche, l'extrusion, la révolution et le congé arête. En raison de leur simplicité, nous ne considérons pas nécessaire d'expliquer leur mode d'emploi.

Premièrement, nous avons commencé par le tableau pour ensuite pouvoir décider la taille des pièces. Pour faciliter la programmation du VBA, nous avons adopté un système des coordonnés. Cela veut dire que les parallélépipèdes qui constituent le tableau ont été nommés comme cubeXY, où le XY est la coordonné du parallélépipède.

Pour les pièces, nous avons adopté la une seule base pour le corps des pièces afin de définir un style et garder une homogénéité. Nous avons commencé par le plus simple, le pion. Pour cette étape, nous avons choisi des commandes très simples. Par ailleurs, La révolution a eu un rôle important car nous l'avons utilisé pour faire le corps et, avec un demi-cercle, la partie supérieure (boule). Avec les commandes que nous avons déjà mentionné, nous avons fabriqué la tour, le roi, la reine et le fou. Pour finir et donner plus de fluidité aux pièces, nous avons beaucoup employé le congé arête.

La dernière pièce, le cavalier, a été la plus difficile et complexe. Nous avons bien cherché les meilleurs exemples pour nous donner une vision de comment modéliser son design. Les défis que nous avons aperçu étaient de le rendre plus réel et d'insérer des détails qui caractérisent mieux un cavalier. De plus, nous avons eu un peu de difficulté avec la taille de cette pièce.

Ainsi, afin de répondre aux besoins de cette pièce et inclure des détails qui donnent une impression de mouvement, nous avons ajouté des yeux ainsi que le crin de cheval. Pour les yeux, nous avons utilisé la poche avec une profondeur relativement petite et au

fond nous avons fait un congé arête plus grand que le reste, avec l'intention de les laisser plus arrondie et de créer une sensation de véritables yeux.

Par ailleurs, nous avons fait une esquisse avec spline avec la finalité de couper une partie des bords et les arrondir en conséquence. Avec cette esquisse, nous avons réalisé une poche, créant donc, des faces ondulés. Le commande utilisé pour répliquer de l'autre côté ces détails a été la symétrie en 3D.

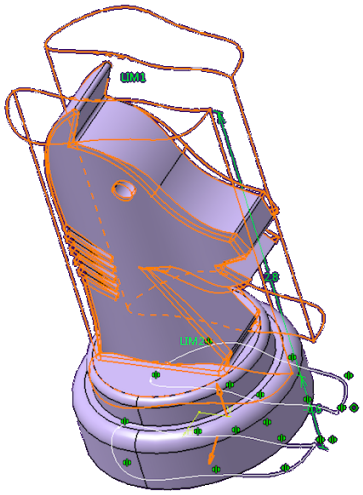


FIGURE 1 – Esquisse de la poche pour faire les faces du cavalier ondulés

En outre, pour qu'il ait la bonne taille, nous avions eu besoin de refaire le cavalier. La première fois que nous l'avons conçu, nous n'étions pas concernés par cet aspect car nous étions distraits par le côté esthétique. Pourtant, la deuxième fois, nous avons prêté plus d'attention aux dimensions. Pour finir, nous avons mis plusieurs congés pour rendre la pièce lisse.

En résumé, dans l'ensemble, les pièces n'étaient pas très difficiles à confectionner. Le plus stimulant était de donner aux pièces les détails dont elles avaient besoin pour avoir un style unique et réaliste.

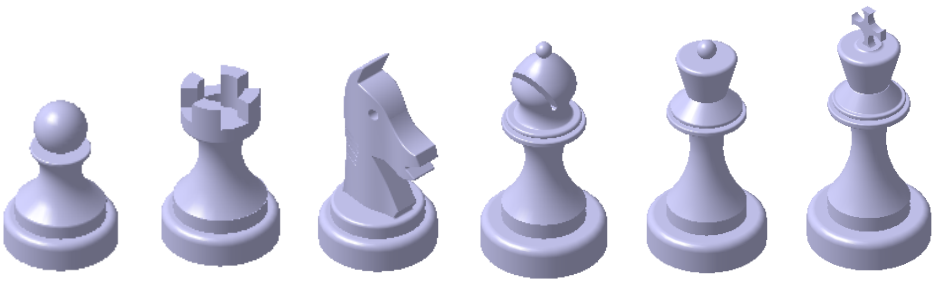


FIGURE 2 – Les six pièces finies

4 Script VBA

4.1 Initialisation - `CATMain()`

Afin d'organiser les positions initiales dans l'échiquier et initialiser les variables nécessaires pour l'exécution du code, nous avons initialisé ces paramètres dans la fonction `CATMain()` parce qu'elle est la première fonction à être exécuté. Les sections suivantes sont organisés en ordre d'exécution pour faciliter la compréhension du code. Seule les parties importantes seront expliqués dans ces sections. Le code complet peut être trouvé dans l'annexe A.

Compte tenu des variables qui doivent être utilisées dans différents fichiers, un module a été créé pour garder ces variables publiques (comme montré dans le script 8 de l'annexe A.1). L'utilisation des ces variables sera expliqué en détails dans les prochaines sections.

4.1.1 Noms des joueurs

Pour commencer, on récupère les noms de joueurs pour afficher après la possession des tours. Par défaut on a "Player 1" et "Player 2" dans le cas où rien n'est inséré ou si les noms égales.

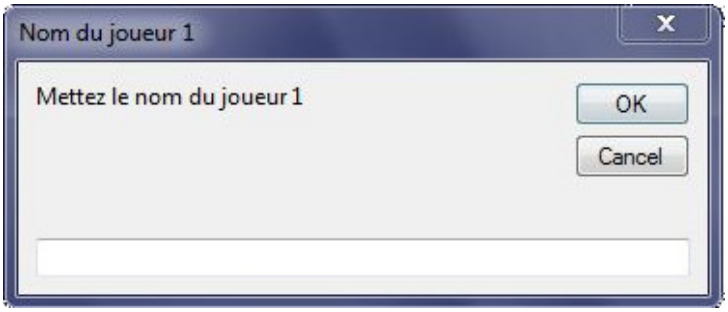


FIGURE 3 – Fenêtre affiché pour insérer les noms

```
playerName1 = InputBox("Mettez le nom du joueur 1")
playerName2 = InputBox("Mettez le nom du joueur 2")
If playerName1 = "" Then
    playerName1 = "Player 1"
End If
If playerName2 = "" Or playerName2 = playerName1 Then
    playerName2 = "Player 2"
End If
```

Script 1 – Récupération des noms des joueurs

4.1.2 Définition des paramètres des pièces

Afin de pouvoir gérer les positions des pièces par rapport à l'échiquier chaque pièce possède trois contraintes liés à la première case de l'échiquier comme montré dans la figure 4. Les contraintes de position pour X et Y (`CorpsPosX` et `CorpsPosY`) sont définis par rapport au centre des pièces et le côté de la première case de l'échiquier et la contrainte de position selon Z (`CorpsPosZ`) sert à faire l'animation du déplacement est défini pas rapport à la surface supérieure de la première case de l'échiquier.

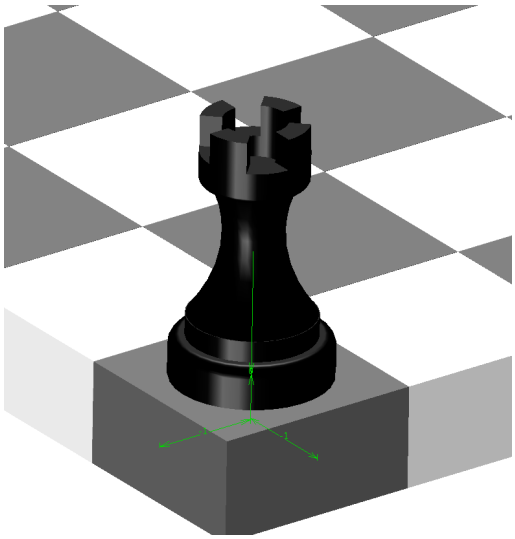


FIGURE 4 – Contraintes des position des pièces par rapport à la première case de l'échiquier

Ensuite, il y a 3 paramètres supplémentaires pour chaque pièce, à quelle joueur la pièce appartient (`CorpsPlayer`), le type de la pièce (`CorpsType`) qui peut être donc `"roi"`, `"pion"`... et une variable pour garder si le premier mouvement à déjà été réalisé (`PionPremierMouvement`) comme montré dans le script 2.

```
For i = 0 To 31    'On initialise avec premier mouvement pas réalisé
    PionPremierMouvement(i) = False
Next i

[...] 'D'autres pièces qui suivent la même logique ci-dessous

CorpsPosX(8) = 0                'On défini que le corps d'indice 8 occupe
CorpsPosY(8) = 0                'la case de position (0,0) de l'échiquier
CorpsPlayer(8) = playerName1    'Cette pièce appartient au joueur 1
CorpsType(8) = "tour"           'C'est une tour

[...] 'D'autres pièces qui suivent la même logique ci-dessus
```

Script 2 – VBScript contenant la définition des paramètres des pièces

Néanmoins, ces paramètres ne sont pas encore liés aux contraintes définis au préalable dans le produit CATIA. Pour le faire, il est nécessaire récupérer le document actuel (le produit `JeuDEchec`), les contraintes dans ce produit et ensuite appliquer la valeur de la contrainte correspondante selon défini dans les paramètres. Le code 3 contient cette association.

Maintenant, on utilise les listes `CorpsPos` pour la partie logique et les listes `deplacement` pour réaliser les changements sur le produit CATIA.

4.1.3 Fenêtre 2D

Afin de contrôler les pièces du jeu d'échec, une fenêtre 2D est crée qui contient 64 boutons, un pour chaque case de l'échiquier et aussi un label qui indique à qui est le tour de jouer qui change selon les noms insérés au début de la partie. Pour faire cette fenêtre

```
Dim productDocumentPrincipal As ProductDocument      'Charge le fichier
Set productDocumentPrincipal = CATIA.ActiveDocument 'CATIA en cours
Dim JeuDeEchec As Product                            'Charge le produit
Set JeuDeEchec = productDocumentPrincipal.Product    'en cours
Dim deplacements As Constraints  'Charge les contraintes de ce produit
Set deplacements = JeuDeEchec.Connections("CATIAConstraints")

Dim deplacementX(31) As Constraint  'On crée des listes de contraintes
Dim deplacementY(31) As Constraint  'mais pas encore associées au
Dim deplacementZ(31) As Constraint  'produit
Dim XValeur(31) As Length           'On crée des listes des valeurs
Dim YValeur(31) As Length           'mais pas encore associées au
Dim ZValeur(31) As Length           'produit

For i = 0 To 31   'Pour chaque corps
    'On associe les éléments de la liste deplacement (X,Y ou Z)
    'à une contrainte du produit
    Set deplacementX(i) = deplacements.Item("Decalage" & i & "X")
    Set deplacementY(i) = deplacements.Item("Decalage" & i & "Y")
    Set deplacementZ(i) = deplacements.Item("Decalage" & i & "Z")
    Set XValeur(i) = deplacementX(i).Dimension  'On associe les listes
    Set YValeur(i) = deplacementY(i).Dimension  'au paramètre Dimension
    Set ZValeur(i) = deplacementZ(i).Dimension  'des contraintes
    deplacementZ(i).Orientation = catCstOrientOpposite 'Orientation
    XValeur(i).Value = (CorpsPosX(i) * -2) - 1  'On associe les valeurs
    YValeur(i).Value = (CorpsPosY(i) * -2) - 1  'selon les paramètres
    ZValeur(i).Value = 0                        'définis au préalable
Next i

JeuDeEchec.Update        'On met à jour l'élément CATIA
FenetreJeuDeEchec.Show   'On affiche la fenêtre 2D pour les commandes
```

Script 3 – VBScript contenant la définition des paramètres des pièces

nous avons utilisé les *UserForms* sur VBScript. La fenêtre initiale est montré dans la figure 5. Les icônes utilisés pour représenter les pièces ont été créés par MadeByOliver sur la plateforme Flaticon.

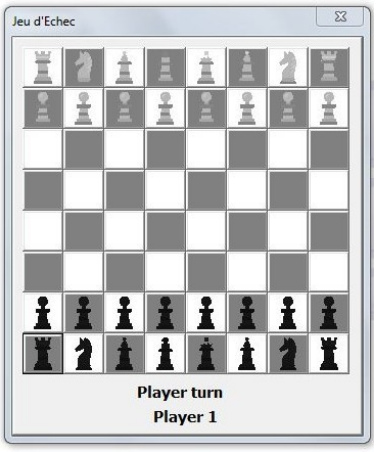


FIGURE 5 – `FenetreJeuDEchec` - Fenêtre 2D pour gérer les pièces du produit CATIA.

Chaque bouton est nommé comme `PosXY` (ex. : `Pos00` pour la case (0,0), `Pos01` pour la case (0,1) ...) est chaque bouton quand cliqué suit la logique indiqué dans le figure 6.

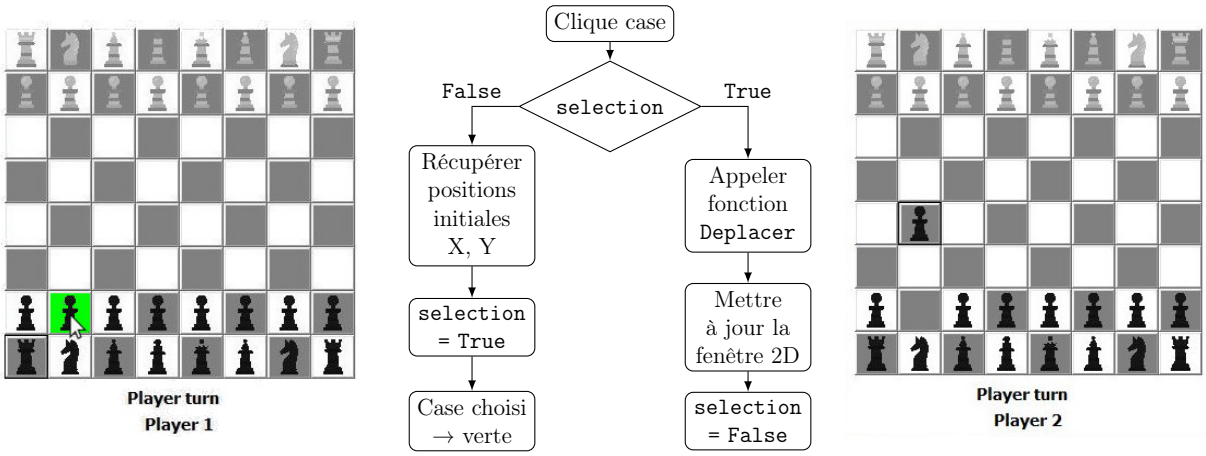


FIGURE 6 – Logique de décision pour un clique dans une case du tableau 2D

4.1.4 Fonction `Deplacer`

Comme montré précédemment, après le deuxième clique sur une case, la fonction `Deplacer` est appelé. Cette fonction a comme objectif réaliser le mouvement des pièces dans le produit CATIA en changeant les valeurs des contraintes. Pour vérifier que le mouvement est permis selon les règles du jeu d'échec, plusieurs règles doivent être vérifiés comme si la case est vide, si l'on a choisi notre propre pièce pour bouger et ainsi de suite. Le code complet se trouve dans l'annexe A.3. Afin de faciliter la logique utilisée, le graphe logique de la figure 7 montre comment le code agit selon les conditions données.

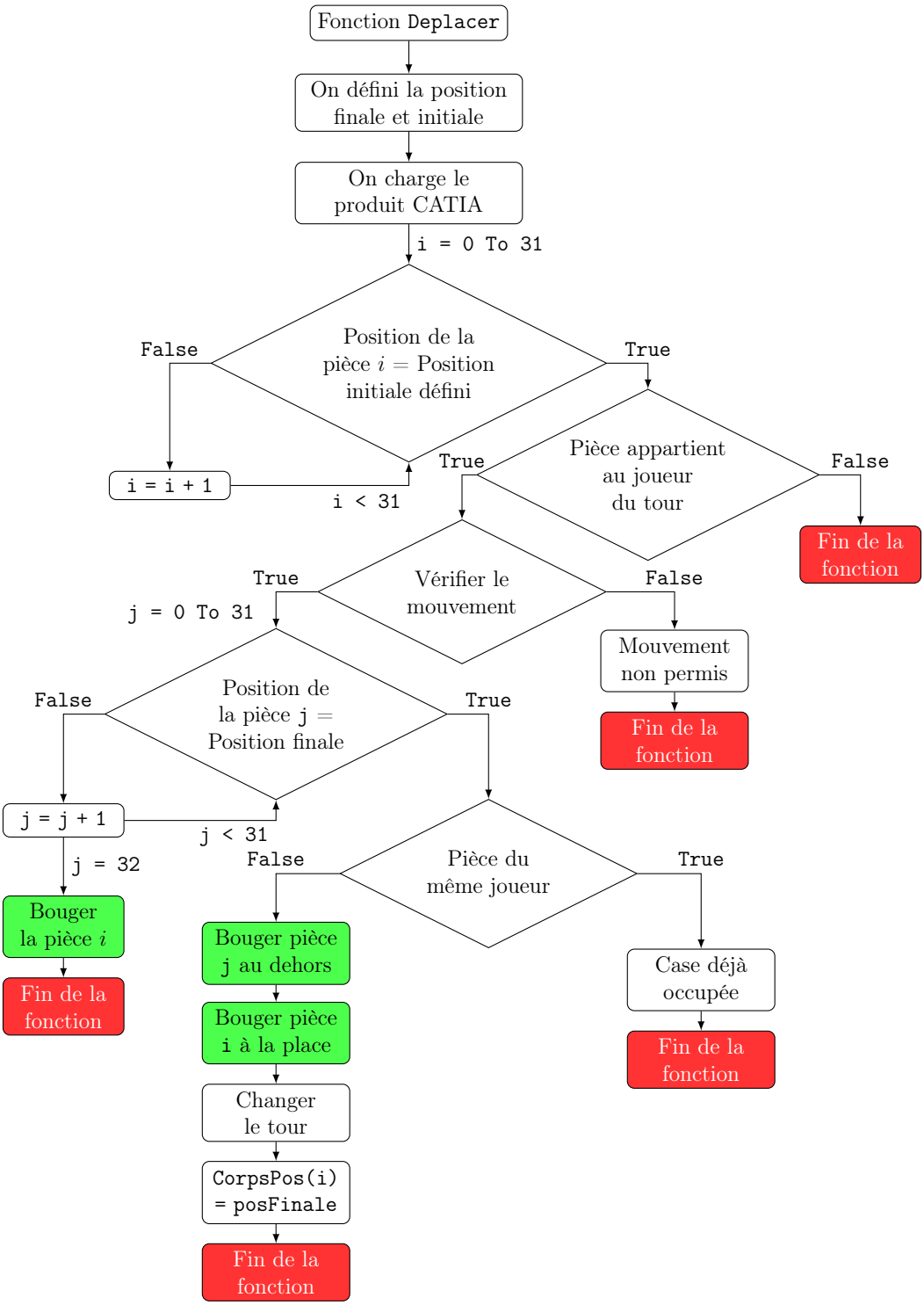


FIGURE 7 – Logique utilisé pour la fonction `Deplacer`

4.1.5 Fonction `mouvementVerif`

Comme montré dans la section précédente, une des étapes de vérification du mouvement consiste à analyser si le mouvement demandé est d'accord avec les règles du jeu d'échec. La fonction `mouvementVerif` a donc comme but vérifier, étant donnée la pièce à bouger (ex : roi, pion, cheval...), si le mouvement est permis ou pas vis-à-vis de la position

finale par rapport à l'initiale (ex : le fou ne peut bouger qu'en diagonales) et aussi des autres pièces (ex : la tour ne peut pas « sauter » d'autres pièces, il faut ainsi vérifier s'il y a des pièces au chemin).

Comme toutes les fonctions ont a peu près la même logique, nous allons utiliser l'exemple de la tour, comme montré dans la figure 8 où les cases vertes montrent où la pièce peut bouger et rouges où il y a d'autres pièces.

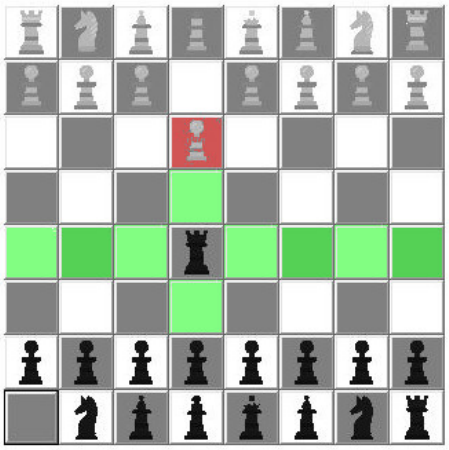


FIGURE 8 – Schéma pour vérifier le mouvement de la tour. Les cases vertes indiquent les mouvements possibles et la case rouge indique qu'une pièce se trouve au chemin d'un mouvement.

```
'---------------------------- TOUR ----------------------------

ElseIf CorpsType(i) = "tour" And ((posFinaleX = posInitialeX) Or (
    posFinaleY = posInitialeY)) Then
'La première ligne (elseif CorpsType(i)...) défini si le mouvement est d
    'accord sans prendre en compte les autres pièces
    For k = 1 To Abs(posFinaleX - posInitialeX + posFinaleY -
    posInitialeY) - 1
     'La boucle for vérifie s'il y a des pièces sur le chemin
       For j = 0 To 31
          If CorpsPosX(j) = posInitialeX + k * Sgn(posFinaleX -
    posInitialeX) And CorpsPosY(j) = posInitialeY + k * Sgn(posFinaleY -
    posInitialeY) Then
             mouvementPermis = False
             Exit Sub
          End If
       Next j
    Next k
mouvementPermis = True
```

Script 4 – VBScript contenant la définition des paramètres des pièces

4.1.6 Animations

Afin de faire un peu plus fluide le jeu, nous avons ajouté des animations lorsque les pièces changent de position. Pour le faire on utilise une boucle où à chaque itération on met à jour la fenêtre CATIA faisant ainsi une idée de mouvement. Le code 5 montre un

exemple d'implémentation utilisé. L'idée est de faire avancer les valeurs des contraintes de façon linéaire. Pour commencer, on monte la pièce (augmenter `CorpsPosZ`), ensuite on translate la pièce selon la direction finale (`CorpsPosX` et `CorpsPosY`) et finalement on descend la pièce (mettre à 0 `CorpsPosZ`).

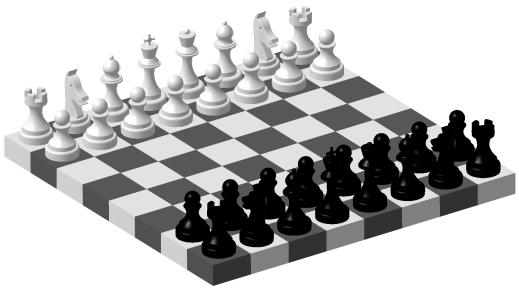
```
' --------------- Animation pour bouger une pièce ---------------

For T = 0 To tempsAnimation 'On fait monter la pièce
    doc.Connections("CATIAConstraints").Item("Decalage" & i & "Z").Dim
    ension.Value = 3 * T / tempsAnimation
    doc.Update
    DoEvents
    Next T

For T = 0 To tempsAnimation 'Animation pour la pièce i selon X et Y
    doc.Connections("CATIAConstraints").Item("Decalage" & i & "X").Dim
    ension.Value = valeurXFin * T / tempsAnimation + (tempsAnimation - T)
     * valeurXInit / tempsAnimation
     doc.Connections("CATIAConstraints").Item("Decalage" & i & "Y").Dim
    ension.Value = valeurYFin * T / tempsAnimation + (tempsAnimation - T)
     * valeurYInit / tempsAnimation
    doc.Update
    DoEvents
    Next T

For T = 0 To tempsAnimation 'On fait descendre la pièce
    doc.Connections("CATIAConstraints").Item("Decalage" & i & "Z").Dim
    ension.Value = (tempsAnimation - T) * 3 / tempsAnimation
    doc.Update
    DoEvents
    Next T
```

Script 5 – VBScript pour l'animation du mouvement d'une pièce

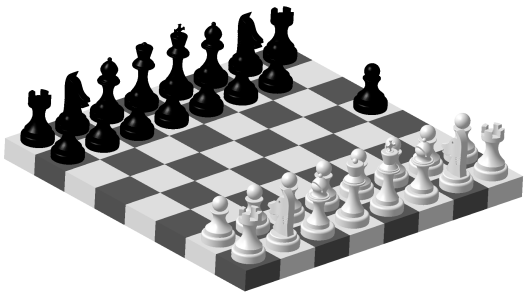
Néanmoins, lorsque l'on a plusieurs contraintes (dans ce cas 96 contraintes) et plusieurs parts (dans ce cas 32) le processeur n'est plus capable de réaliser avec une bonne qualité les transitions. Alors pour que le mouvement soit acceptable, on utilise `tempsAnimation = 3`. Mais si l'ordinateur en question possède un meilleur processeur graphique, ce paramètre peut être augmenté afin d'améliorer la fluidité du mouvement.

4.1.7 Cameras

Après la mise en œuvre de tout les règles et mouvements, nous avons cherché à faire le jeu encore plus agréable. A partir de cela, nous avons créé à partir des cameras et *viewpoints* de CATIA que le point de vue change selon le joueur. Afin de pouvoir accéder à la camera depuis tout les fonctions, nous l'avons mis comme étant un paramètre publique. Ensuite, on décrit tout les paramètres de la camera comme position, angle de vue etc... Pour finalement quand on change le tour, le point de vue change selon le joueur. Les deux points de vue sont montré dans les figures 9a et 9b. Le code 6 montre les parties des cameras dans les différentes fichiers.



(a) Point de vue du premier joueur



(b) Point de vue du deuxième joueur

```
'----- Pour acceder a la camera depuis tout les fonctions -----

Public fenetrePrincipale As SpecsAndGeomWindow 'Fenêtre principale
Public visionObs As Viewer3D        'Variable pour un observateur
Public camera3D1 As Camera3D        'Variable pour la première camera
Public camera3D2 As Camera3D        'Variable pour la deuxième camera
Public viewpoint3D1 As Viewpoint3D 'Viewpoint de la camera 1
Public viewpoint3D2 As Viewpoint3D 'Viewpoint de la camera 2

'--- Paramètres de la camera 1 ---
Public origineCamera3D1(2)     'Position de l'origine de la camera
Public directionVision3D1(2)   'Angle de vision
Public directionUp3D1(2)

'--- Paramètres de la camera 2 ---
Public origineCamera3D2(2)     'Semblable à la camera 1
Public directionVision3D2(2)
Public directionUp3D2(2)
```

Script 6 – VBScript contenant les paramètres pour la définition des points de vue dans le fichier `Variables`

```
'------ Configurations pour l'angle d'affichage ----------

Set fenetrePrincipale = CATIA.ActiveWindow      'Fenêtre principale
Set visionObs = fenetrePrincipale.ActiveViewer  'Observateur
Set camera3D1 = visionObs.NewCamera() 'Création de la camera 1
Set camera3D2 = visionObs.NewCamera() 'Création de la camera 2
Set viewpoint3D1 = camera3D1.Viewpoint3D  'Création du viewpoint 1
Set viewpoint3D2 = camera3D2.Viewpoint3D  'Création du viewpoint 2

[...] 'On défini les paramètres des cameras

'On défini la camera 1 pour commencer la partie
visionObs.Viewpoint3D = viewpoint3D1
```

Script 7 – VBScript contenant les paramètres pour la définition des points de vue dans le fichier `JeuDeEchec`

5 Produit final

Lors du développement de notre projet, concernant le VBA, nous avons eu des difficultés par rapport à mise en œuvre optimale des boutons (les cases dans la fenêtre 2D). Le problème était surtout l'impossibilité de mettre les boutons dans une liste pour ensuite appeler leurs paramètres dans une boucle. Cela nous a fait écrire pour chaque bouton son action. De plus, comme cité précédemment, nous avons rencontré des problèmes aussi par rapport à l'animation lorsqu'une pièce bougeait. Ce problème, issue de la quantité importante de contrainte dans le produit, a fait que notre animation soit moins fluide que souhaité.

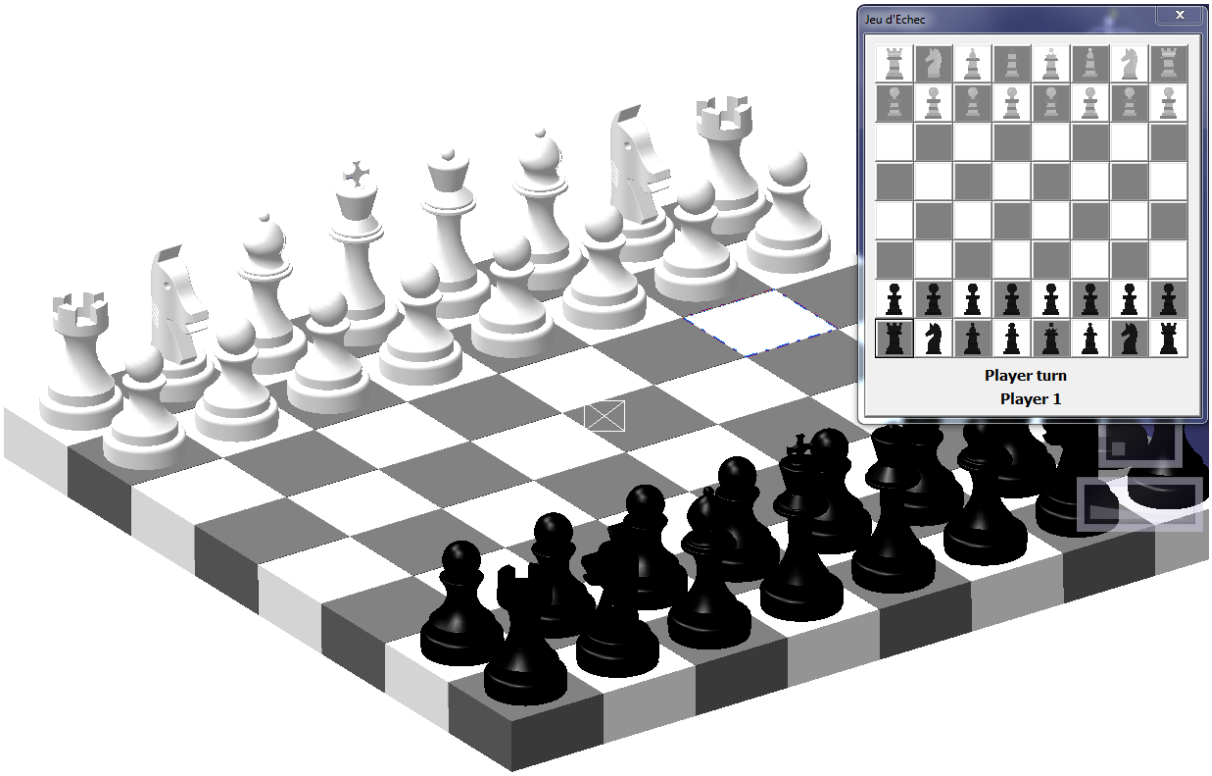


FIGURE 10 – Jeu d'échec construit en utilisant CATIA et VBA

La figure 10 montre le projet réalisé. On voit dans la figure les éléments cités précédemment. Nous avons vu au cours du développement de ce projet qu'il existe plusieurs outils disponibles actuellement pour faire de la maquettage numérique. Des nombreuses applications sont possibles en utilisant VBA et CATIA. En plus, plusieurs d'autres outils comme *Kinematics* sont très utiles pour modéliser divers systèmes.

Le développement nous a donc permis d'acquérir des compétences divers concernant CATIA et son intégration et manipulation avec VBA. Cette connaissance peut nous servir dans l'avenir pour l'automatisation de certains tâches lors d'un projet ingénieur ou faciliter l'utilisation de certains outils pour d'autres personnes.

A VBScript complet

A.1 Fichier `Variables`

```
'----------- Variables pour le mouvement ------------------------

Public posInitialeX As Integer      'Position X sélectionnée
Public posInitialeY As Integer      'Position Y sélectionnée
Public valeurXFin As Double         'Position X finale
Public valeurYFin As Double         'Position Y finale
Public tempsAnimation As Integer    'Nb d'itérations pour l'animation
Public selection As Boolean         'Variable de sélection
Public mouvementPermis As Boolean   'Variable pour validation mouvement
Public tourPlayer As String         'Pour changer les tours

'--------- Variables pour la définition des pièces ---------------

Public CorpsPlayer(31) As String    'De quel joueur est la pièce
Public CorpsPosX(31) As Integer     'Position X de la pièce
Public CorpsPosY(31) As Integer     'Position Y de la pièce
Public CorpsPosZ(31) As Integer     'Position Z de la pièce
Public CorpsType(31) As String      'Le type de la pièce
Public PionPremierMouvement(31) As Boolean  'Premier mouvement du pion
Public Dehors1 As Integer           'Position quand la pièce est enlévée
Public Dehors2 As Integer           'Aussi mais pour l'autre joueur
Public playerName1 As String        'Nom du premier joueur
Public playerName2 As String        'Nom du déuxième joueur

'----- Pour acceder a la camera depuis tout les fonctions -----

Public fenetrePrincipale As SpecsAndGeomWindow 'Var pour la fênetre prin
Public visionObs As Viewer3D                   'Var pour un observateur
Public camera3D1 As Camera3D                   'Variable 1ère camera
Public camera3D2 As Camera3D                   'Variable 2ème camera
Public viewpoint3D1 As Viewpoint3D             'Viewpoint camera 1
Public viewpoint3D2 As Viewpoint3D             'Viewpoint camera 2

'--- Paramètres de la camera 1 ------------------
Public origineCamera3D1(2)       'Position de l'origine de la camera
Public directionVision3D1(2)     'Angle de vision
Public directionUp3D1(2)

'--- Paramètres de la camera 2 ------------------
Public origineCamera3D2(2)       'Semblable à la camera 1
Public directionVision3D2(2)
Public directionUp3D2(2)

'---------------- Dossier -----------------------

Public chemin As String 'Pour récupérer le chemin du dossier
```

Script 8 – `Variables` - VBScript contenant les variables publiques

A.2 Fichier FenetreJeuDeEchec

```
'Pour chaque bouton on appele la fonction ActionButton
Private Sub Pos00_Click()
    Call ActionButton(0, 0, Pos00)
End Sub

[...]

Private Sub PosXY_Click()
    Call ActionButton(X, Y, PosXY)
    'Avec X et Y étant les coordonnées de l'échiquier
End Sub

[...]

Private Sub Pos77_Click()
    Call ActionButton(7, 7, Pos77)
End Sub

Public Sub Colorer()

   'On met tout l'échiquier aux couleurs initiales
    Pos00.BackColor = &H808080
    Pos01.BackColor = RGB(255, 255, 255)
    [...]
    PosXY.BackColor = &H808080
    PosX(Y+1).BackColor = RGB(255, 255, 255)
    'Semblable aux boutons
    [...]
    Pos76.BackColor = RGB(255, 255, 255)
    Pos77.BackColor = &H808080

End Sub

Public Sub ActionButton(ByVal posX, ByVal posY, ByRef Button)

    'Fonction appelé quand un bouton est cliqué
    If selection = True Then         'Si une case a déjà été selectionnéé
        Call Deplacer(posX, posY)    'On appelle la fonction deplacer
        Call Colorer                 'On colore tout les cases
        Call ChangeFigure(Button, posX, posY)'On met à jour les figures
        Exit Sub                     'On sort de la fonction
    End If

    If selection = False Then       'Pour sélectionner la pièce à bouger
        posInitialeX = posX         'On recupère la position initiale X
        posInitialeY = posY         'On recupère la position initiale Y
        selection = True            'Case sélectionnée
        Button.BackColor = RGB(0, 255, 0)'Case change à vert
        Exit Sub                    'On sort de la fonction
    End If

End Sub
```

```
Public Sub ChangeFigure(ByRef Button,ByVal posFinaleX,ByVal posFinaleY)
    'Fonction utilisé pour mettre à jour les figures dans l'échiquier 2D
    If mouvementPermis = True Then   'Si le mouvement est permis,
        If posInitialeX = 0 Then     'on enlève la figure du bouton
            If posInitialeY = 0 Then 'où était la pièce
                Pos00.Picture = LoadPicture(none)
            [...]
            ElseIf posInitialeY = 7 Then
                Pos07.Picture = LoadPicture(none)

        [...]

        ElseIf posInitialeX = 7 Then
            If posInitialeY = 0 Then
                Pos70.Picture = LoadPicture(none)
            [...]
                Pos77.Picture = LoadPicture(none)
            End If
        End If
    End If

    Dim playerInd As Integer    'Indice pour les noms des figures
    changeLabelName (tourPlayer)'On change le nom affiché pour le tour

    For i = 0 To 31
       'On regarde quelle pièce se trouve maintenant sur la case finale
        If posFinaleX = CorpsPosX(i) And posFinaleY = CorpsPosY(i) Then
            If CorpsPlayer(i) = playerName1 Then
            'Si elle appartient au premier joueur
                playerInd = 1    'alors indice = 1
            Else                 'sinon
                playerInd = 2    'indice = 2
            End If
            'On change la figure affiché selon le type de la pièce et le
    joueur (donc la couleur de la pièce) ex: roi1, pion2...
            Button.Picture = LoadPicture(chemin & "\ " & CorpsType(i) &
   playerInd & ".bmp")
            'A changer par rapport où se trouvent les figures
        End If
    Next i
    mouvementPermis = False 'On remet la variable à False
End Sub

Public Sub changeLabelName(ByVal nom As String)
'On appele cette fonction pour changer le nom affiché du tour
    playerLabel.Caption = nom 'On change la valeur du label
    playerLabel.TextAlign = fmTextAlignCenter 'On aligne le nom
    If nom = playerName1 Then       'Si c'est le premier joueur
        visionObs.Viewpoint3D = viewpoint3D1'Changer la vue à camera 1
    Else                                    'Sinon
        visionObs.Viewpoint3D = viewpoint3D2'Changer la vue à camera 2
    End If
End Sub
```

Script 9 – `FenetreJeuDeEchec` - VBScript contenant les actions pour la fenêtre 2D

A.3 Fichier JeuDechec

```
Sub Deplacer(ByVal posFinaleX, ByVal posFinaleY)
'Fonction pour deplacer les pièces dans l'échiquier
    valeurXInit = (posInitialeX * -2) - 1 'Calcul des contraintes
    valeurYInit = (posInitialeY * -2) - 1 'basés sur les positions
    valeurXFin = (posFinaleX * -2) - 1    'de l'échiquier
    valeurYFin = (posFinaleY * -2) - 1

    Set doc = CATIA.ActiveDocument.Product      'Document CATIA active

    For i = 0 To 31  'Pour toutes les pièces
      If CorpsPosX(i) = posInitialeX And CorpsPosY(i) = posInitialeY
         And CorpsPlayer(i) = tourPlayer Then
      'S'il y a une pièce dans la position intiale
      Call mouvementVerif(i, posFinaleX, posFinaleY)
        'On vérifie si le mouvement est possible
      If mouvementPermis = True Then     'Si c'est possible
         For j = 0 To 31                 'On regarde les autres pièces
            If i = j Then      'Si c'est la même pièce on fait rien
            Else
               If CorpsPosX(j) = posFinaleX And CorpsPosY(j) =
                  posFinaleY Then 'On cherche une pièce à la pos finale
                  If CorpsPlayer(i) = CorpsPlayer(j) Then
                  'Si la pièce appartient au même joueur
                  MsgBox "Case déjà occupée", vbOKOnly, Attention
                  'Alors on dit que la case est occuppée
                  mouvementPermis = False
                  'Le mouvement n'est donc pas permis
                  selection = False 'On deselectionne la case
                  Exit Sub          'On sort de la fonction
                  Else
                     For T = 0 To tempsAnimation
                     'Animation pour la pièce adversaire
                        doc.Connections("CATIAConstraints").Item("
                          Decalage" & j & "Z").Dimension.Value = 3
                           * T / tempsAnimation 'On monte selon Z
                        doc.Update 'On met à jour la visualisation
                        DoEvents   'pour créer une animation
                        Next T     'en utilisant des itérations

                        If CorpsPlayer(j) = playerName1 Then
                        'Même chose pour les autres directions
                           For T = 0 To tempsAnimation
                           'Et les autres pièces concernées
                           doc.Connections("CATIAConstraints").Item("
   Decalage" & j & "X").Dimension.Value = Dehors1 * T / tempsAnimation +
    (tempsAnimation - T) * valeurXFin / tempsAnimation
                           doc.Connections("CATIAConstraints").Item("
   Decalage" & j & "Y").Dimension.Value = 2 * T / tempsAnimation + (
   tempsAnimation - T) * valeurYFin / tempsAnimation
                           doc.Update
                           DoEvents
                           Next T
                           Dehors1 = Dehors1 - 2
               'Pour que les pièces qui sortent ne se superposent pas
```

```
                        If CorpsType(j) = "roi" Then
                        'Si la pièce éliminée est un roi
                        MsgBox "Le joueur " & playerName2 & "a gagné"
                        ,vbInformation, "Fin de la partie" 'Fin jeu
                        FenetreJeuDeEchec.Hide
                        'On ferme la fenêtre
                        End If
                    Else
                        For T = 0 To tempsAnimation
                        'Semblable mais pour l'autre joueur
                        doc.Connections("CATIAConstraints").Item("
Decalage" & j & "X").Dimension.Value = Dehors2 * T / tempsAnimation +
 (tempsAnimation - T) * valeurXFin / tempsAnimation
                        doc.Connections("CATIAConstraints").Item("
Decalage" & j & "Y").Dimension.Value = -17 * T / tempsAnimation + (
tempsAnimation - T) * valeurYFin / tempsAnimation
                        doc.Update 'la différence est que cette
                        DoEvents 'pièce va de l'autre côté
                        Next T
                        Dehors2 = Dehors2 - 2
                        If CorpsType(j) = "roi" Then
                        MsgBox "Le joueur " & playerName1 & "a gagné"
                            , vbInformation, "Fin de la partie"
                        FenetreJeuDeEchec.Hide
                        End If
                    End If
                For T = 0 To tempsAnimation 'On descend la pièce
                    doc.Connections("CATIAConstraints").Item("
Decalage" & j & "Z").Dimension.Value = (tempsAnimation - T) * 3 /
tempsAnimation + -1 * T / tempsAnimation
                    doc.Update
                    DoEvents
                    Next T
                    CorpsPosX(j) = 20 'Valeur seulement pour ne pas
                    CorpsPosY(j) = 0 'déranger les mouvements futurs
                End If
            End If
        End If
    Next j
        For T = 0 To tempsAnimation
        'On fait l'animation pour la pièce i selon Z
            doc.Connections("CATIAConstraints").Item("Decalage" &
i & "Z").Dimension.Value = 3 * T / tempsAnimation
            doc.Update
            DoEvents
            Next T
        For T = 0 To tempsAnimation
        'On fait l'animation pour la pièce i selon X et Y
            doc.Connections("CATIAConstraints").Item("Decalage" &
i & "X").Dimension.Value = valeurXFin * T / tempsAnimation + (
tempsAnimation - T) * valeurXInit / tempsAnimation
            doc.Connections("CATIAConstraints").Item("Decalage" &
i & "Y").Dimension.Value = valeurYFin * T / tempsAnimation + (
tempsAnimation - T) * valeurYInit / tempsAnimation
            doc.Update
            DoEvents
```

```
                Next T
            For T = 0 To tempsAnimation
            'On fait l'animation pour la pièce i selon Z
                doc.Connections("CATIAConstraints").Item("Decalage" &
   i & "Z").Dimension.Value = (tempsAnimation - T) * 3 / tempsAnimation
                doc.Update
                DoEvents
                Next T
            If tourPlayer = playerName1 Then    'On change le tour
                tourPlayer = playerName2
            Else
                tourPlayer = playerName1
            End If
            CorpsPosX(i) = posFinaleX
            'La pièce possède desormais la position finale
            CorpsPosY(i) = posFinaleY
            PionPremierMouvement(i) = True
            'Elle a réalisé certainement la premier mouvement
            selection = False 'On deselectionne les cases
            doc.Update        'On met à jour la fenêtre CATIA
            Exit Sub          'On sort de la fonction
        Else
            MsgBox "Mouvement non permis"
            'Si le mouvement n'est pas permis
        End If
            selection = False    'On deselectionne les cases
            doc.Update           'On met à jour
            Exit Sub             'On sort de la fonction
    End If
   Next i
   selection = False  'Si rien se passe alors on deselectionne les cases
   doc.Update         'On met à jour
End Sub


Sub mouvementVerif(ByVal i, ByVal posFinaleX, ByVal posFinaleY)
   'Fonction pour analyser si le mouvement est permis compte tenu de la
   direction et des autres pièces
   'if ... -> analyse si le mouvement est permis étant donnée seulement
   la position initiale et finale
      'for ... -> analyse pour toutes les pièces s'il y a une entre la
   position finale et intiale
         'S'il y en a, alors on peut pas bouger
      'Prochaine pièce
   'Fin du if

    '------------ Pion joueur 1 ------------
    If CorpsType(i) = "pion" And CorpsPlayer(i) = playerName1 And (Abs(
   posFinaleY - posInitialeY) < 2 And (posFinaleX - posInitialeX) = 1)
   Then
      If Abs(posFinaleY - posInitialeY) = 1 Then
         For j = 16 To 31
            If CorpsPosX(j) = posFinaleX And posFinaleY = CorpsPosY(j)
   Then
               mouvementPermis = True
            End If
```

```
        Next j
    Else
        mouvementPermis = True
    End If

 '--------------- Pion joueur 2 ---------------
ElseIf CorpsType(i) = "pion" And CorpsPlayer(i) = playerName2 And (
Abs(posFinaleY - posInitialeY) < 2 And (posFinaleX - posInitialeX) =
-1) Then
    If Abs(posFinaleY - posInitialeY) = 1 Then
        For j = 0 To 15
            If CorpsPosX(j) = posFinaleX And posFinaleY = CorpsPosY(j)
Then
                mouvementPermis = True
            End If
        Next j
    Else
        mouvementPermis = True
    End If

 '------- Pion joueur 1 premier mouvement ---------
ElseIf CorpsType(i) = "pion" And CorpsPlayer(i) = playerName1 And
posFinaleY = posInitialeY And (posFinaleX - posInitialeX) = 2 And
PionPremierMouvement(i) = False Then
    mouvementPermis = True
    PionPremierMouvement(i) = True

 '--------- Pion joueur 2 premier mouvement ---------
 ElseIf CorpsType(i) = "pion" And CorpsPlayer(i) = playerName2 And
posFinaleY = posInitialeY And (posFinaleX - posInitialeX) = -2 And
PionPremierMouvement(i) = False Then
     mouvementPermis = True
     PionPremierMouvement(i) = True

 '-------------------- Roi --------------------
ElseIf CorpsType(i) = "roi" And (Abs(posFinaleX - posInitialeX) <= 1
And Abs(posFinaleY - posInitialeY) <= 1) Then
    mouvementPermis = True

'-------------------- Fou --------------------
ElseIf CorpsType(i) = "fou" And (Abs(posFinaleX - posInitialeX) = Abs
(posFinaleY - posInitialeY)) And Abs(posInitialeX - posFinaleX) > 0
Then
    For k = 1 To Abs(posFinaleX - posInitialeX) - 1
        For j = 0 To 31
            If CorpsPosX(j) = posInitialeX + k * Sgn(posFinaleX -
posInitialeX) And CorpsPosY(j) = posInitialeY + k * Sgn(posFinaleY -
posInitialeY) Then
                mouvementPermis = False
                Exit Sub
            End If
        Next j
    Next k
    mouvementPermis = True
```

```
'----------------- Cheval -----------------
  ElseIf CorpsType(i) = "cheval" And ((Abs(posFinaleX - posInitialeX) =
   2 And Abs(posFinaleY - posInitialeY) = 1) Or (Abs(posFinaleX -
  posInitialeX) = 1 And Abs(posFinaleY - posInitialeY) = 2)) Then
     mouvementPermis = True

   '---------------- tour ----------------------
  ElseIf CorpsType(i) = "tour" And ((posFinaleX = posInitialeX) Or (
  posFinaleY = posInitialeY)) Then
     For k = 1 To Abs(posFinaleX - posInitialeX + posFinaleY -
  posInitialeY) - 1
        For j = 0 To 31
           If CorpsPosX(j) = posInitialeX + k * Sgn(posFinaleX -
  posInitialeX) And CorpsPosY(j) = posInitialeY + k * Sgn(posFinaleY -
  posInitialeY) Then
              mouvementPermis = False
              Exit Sub
           End If
        Next j
     Next k
     mouvementPermis = True
   '---------------- Reine --------------------
  ElseIf CorpsType(i) = "reine" And ((Abs(posFinaleX - posInitialeX) =
  Abs(posFinaleY - posInitialeY)) Or ((posFinaleX = posInitialeX) Or (
  posFinaleY = posInitialeY))) Then
     If Abs(posFinaleX - posInitialeX) = Abs(posFinaleY - posInitialeY)
   Then
        For k = 1 To Abs(posFinaleX - posInitialeX) - 1
           For j = 0 To 31
              If CorpsPosX(j) = posInitialeX + k * Sgn(posFinaleX -
  posInitialeX) And CorpsPosY(j) = posInitialeY + k * Sgn(posFinaleY -
  posInitialeY) Then
                 mouvementPermis = False
                 Exit Sub
              End If
           Next j
        Next k
     ElseIf posFinaleX = posInitialeX Or posFinaleY = posInitialeY Then
        For k = 1 To Abs(posFinaleX - posInitialeX + posFinaleY -
  posInitialeY) - 1
           For j = 0 To 31
              If CorpsPosX(j) = posInitialeX + k * Sgn(posFinaleX -
  posInitialeX) And CorpsPosY(j) = posInitialeY + k * Sgn(posFinaleY -
  posInitialeY) Then
                 mouvementPermis = False
                 Exit Sub
              End If
           Next j
        Next k
     End If
     mouvementPermis = True
   '--------- Si aucun des cas s'applique  ------------
  Else
     mouvementPermis = False
  End If
End Sub
```

```
Sub CATMain()

playerName1 = InputBox("Mettez le nom du joueur 1", "Nom du joueur 1")
playerName2 = InputBox("Mettez le nom du joueur 2", "Nom du joueur 2")
If playerName1 = "" Then      'Si on ne met rien
   playerName1 = "Player 1" 'Alors nom = Player 1
End If
If playerName2 = "" Or playerName2 = playerName1 Then
   playerName2 = "Player 2" 'la même chose
End If

For i = 0 To 31       'On initialise avec premier mouvement pas réalisé
   PionPremierMouvement(i) = False
Next i

' -------- 1er joueur -------
For i = 0 To 7                        'Pions du premier joueur
   CorpsPosX(i) = 1
   CorpsPosY(i) = i
   CorpsPlayer(i) = playerName1
   CorpsType(i) = "pion"
Next i

CorpsPosX(8) = 0                'On défini que le corps d'indice 8 occupe
CorpsPosY(8) = 0                'la case de position (0,0) de l'échiquier
CorpsPlayer(8) = playerName1    'Cette pièce appartient au joueur 1
CorpsType(8) = "tour"           'C'est une tour

CorpsPosX(9) = 0               'Les autres pièces suivent la même logique
CorpsPosY(9) = 1
CorpsPlayer(9) = playerName1
CorpsType(9) = "cheval"

CorpsPosX(10) = 0
CorpsPosY(10) = 2
CorpsPlayer(10) = playerName1
CorpsType(10) = "fou"

CorpsPosX(11) = 0
CorpsPosY(11) = 3
CorpsPlayer(11) = playerName1
CorpsType(11) = "roi"

CorpsPosX(12) = 0
CorpsPosY(12) = 4
CorpsPlayer(12) = playerName1
CorpsType(12) = "reine"

CorpsPosX(13) = 0
CorpsPosY(13) = 5
CorpsPlayer(13) = playerName1
CorpsType(13) = "fou"

CorpsPosX(14) = 0
CorpsPosY(14) = 6
CorpsPlayer(14) = playerName1
```

```
CorpsType(14) = "cheval"

CorpsPosX(15) = 0
CorpsPosY(15) = 7
CorpsPlayer(15) = playerName1
CorpsType(15) = "tour"

' -------- 2eme joueur -------
For i = 16 To 23                    'Pions du deuxieme joueur
   CorpsPosX(i) = 6
   CorpsPosY(i) = i - 16
   CorpsPlayer(i) = playerName2
   CorpsType(i) = "pion"
Next i

CorpsPosX(24) = 7
CorpsPosY(24) = 0
CorpsPlayer(24) = playerName2
CorpsType(24) = "tour"

CorpsPosX(25) = 7
CorpsPosY(25) = 1
CorpsPlayer(25) = playerName2
CorpsType(25) = "cheval"

CorpsPosX(26) = 7
CorpsPosY(26) = 2
CorpsPlayer(26) = playerName2
CorpsType(26) = "fou"

CorpsPosX(27) = 7
CorpsPosY(27) = 3
CorpsPlayer(27) = playerName2
CorpsType(27) = "roi"

CorpsPosX(28) = 7
CorpsPosY(28) = 4
CorpsPlayer(28) = playerName2
CorpsType(28) = "reine"

CorpsPosX(29) = 7
CorpsPosY(29) = 5
CorpsPlayer(29) = playerName2
CorpsType(29) = "fou"

CorpsPosX(30) = 7
CorpsPosY(30) = 6
CorpsPlayer(30) = playerName2
CorpsType(30) = "cheval"

CorpsPosX(31) = 7
CorpsPosY(31) = 7
CorpsPlayer(31) = playerName2
CorpsType(31) = "tour"
```

```
' --- Parametres ---

selection = False                'Aucune case n'est sélectionnée
mouvementPermis = False          'initialisation de variable
tourPlayer = playerName1         'Le premier joueur commence
tempsAnimation = 3               'On défini la quantité d'iterations
Dehors1 = 0                      'On initialise les variables
Dehors2 = 0

Dim productDocumentPrincipal As ProductDocument 'Charger le document
Set productDocumentPrincipal = CATIA.ActiveDocument 'CATIA en cours
chemin = productDocumentPrincipal.Path                'Chemin du produit
Dim JeuDeEchec As Product                          'Define le produit
Set JeuDeEchec = productDocumentPrincipal.Product  'en cours
Dim deplacements As Constraints            'On charge les contraintes
Set deplacements = JeuDeEchec.Connections("CATIAConstraints")

Dim deplacementX(31) As Constraint 'On crée des listes de contraintes
Dim deplacementY(31) As Constraint 'mais pas encore associées au produit
Dim deplacementZ(31) As Constraint
Dim XValeur(31) As Length 'On crée des listes de valeurs
Dim YValeur(31) As Length 'mais pas encore associées au produit
Dim ZValeur(31) As Length

For i = 0 To 31
   'On associe les contraintes aux éléments des listes
   Set deplacementX(i) = deplacements.Item("Decalage" & i & "X")
   Set deplacementY(i) = deplacements.Item("Decalage" & i & "Y")
   Set deplacementZ(i) = deplacements.Item("Decalage" & i & "Z")
   Set XValeur(i) = deplacementX(i).Dimension
   Set YValeur(i) = deplacementY(i).Dimension
   Set ZValeur(i) = deplacementZ(i).Dimension
   deplacementZ(i).Orientation = catCstOrientOpposite 'Orientation Z
   XValeur(i).Value = (CorpsPosX(i) * -2) - 1 'On associe les valeurs
   YValeur(i).Value = (CorpsPosY(i) * -2) - 1 'selon les paramètres
   ZValeur(i).Value = 0                       'définis au préalable
Next i

'---------- Configurations pour l'angle d'affichage ----------

Set fenetrePrincipale = CATIA.ActiveWindow 'Fenêtre principale
Set visionObs = fenetrePrincipale.ActiveViewer 'Observateur
Set camera3D1 = visionObs.NewCamera() 'On crée 2 nouvelles cameras
Set camera3D2 = visionObs.NewCamera()
Set viewpoint3D1 = camera3D1.Viewpoint3D 'On crée les viewpoints
Set viewpoint3D2 = camera3D2.Viewpoint3D 'des cameras

'--- Paramètres de la camera 1 ---

origineCamera3D1(0) = -8.973027 'On défini les paramètres de la camera 1
origineCamera3D1(1) = -16.130461
origineCamera3D1(2) = 15.830488
Set viewpoint3D1Variant = viewpoint3D1
viewpoint3D1Variant.PutOrigin origineCamera3D1  'Origine de la camera
```

```
directionVision3D1(0) = 0.509698
directionVision3D1(1) = 0.744311
directionVision3D1(2) = -0.43152
Set viewpoint3D1Variant = viewpoint3D1
viewpoint3D1Variant.PutSightDirection directionVision3D1  'Vision camera

directionUp3D1(0) = 0.250634
directionUp3D1(1) = 0.351352
directionUp3D1(2) = 0.902072
Set viewpoint3D1Variant = viewpoint3D1
viewpoint3D1Variant.PutUpDirection directionUp3D1

viewpoint3D1.FocusDistance = 30.75342 'Focus
viewpoint3D1.Zoom = 0.135442          'Zoom de la camera

'--- Paramètres de la camera 2 ---

origineCamera3D2(0) = 25.86628 'Semblable à la camera 1
origineCamera3D2(1) = 27.935162
origineCamera3D2(2) = 15.100451
Set viewpoint3D2Variant = viewpoint3D2
viewpoint3D2Variant.PutOrigin origineCamera3D2

directionVision3D2(0) = -0.604699
directionVision3D2(1) = -0.663898
directionVision3D2(2) = -0.440093
Set viewpoint3D2Variant = viewpoint3D2
viewpoint3D2Variant.PutSightDirection directionVision3D2

directionUp3D2(0) = -0.258623
directionUp3D2(1) = -0.361786
directionUp3D2(2) = 0.890391
Set viewpoint3D2Variant = viewpoint3D2
viewpoint3D2Variant.PutUpDirection directionUp3D2

viewpoint3D2.FocusDistance = 30.774475
viewpoint3D2.Zoom = 0.136432

' -----------------

visionObs.Viewpoint3D = viewpoint3D1 'Camera 1 pour début de partie
'On change le label pour le premier nom
Call FenetreJeuDeEchec.changeLabelName(playerName1)
JeuDeEchec.Update      'On met à jour la fenêtre CATIA
FenetreJeuDeEchec.Show 'On affiche la fenêtre 2D pour jouer

Dim camerasCrees As Cameras      'A la fin, on récupère les cameras
Set camerasCrees = productDocumentPrincipal.Cameras 'Pour les effacer

camerasCrees.Remove "camera000" 'On efface les cameras créées
camerasCrees.Remove "camera001"
End Sub
```