Table of Contents

Guide Technique - Positionnement des Immatriculations TBM

Version : 2.0 (Simplifié) **Date** : 19/12/2025 **Audience** : Intégrateurs tiers utilisant l'API Lumiscaphe WebRender **Objectif** : Placement dynamique de l'immatriculation sur avions TBM

Table des Matières

---

1. Introduction

1.1 Objectif

Ce guide décrit comment calculer et positionner les lettres d'immatriculation sur un avion TBM en 3D, en utilisant l'API Lumiscaphe WebRender.

Périmètre : - Calcul des positions 3D des lettres - Construction du payload JSON pour l'API - Gestion des largeurs variables des caractères

Hors périmètre : Couleurs, styles de lettres, schémas de peinture (voir autres guides).

1.2 Prérequis

Techniques : - APIs REST - Coordonnées 3D - Parsing XML - Géométrie de base

Accès : - API Lumiscaphe WebRender - Base de données XML TBM

---

2. Contexte Métier

2.1 Qu'est-ce qu'une Immatriculation ?

L'immatriculation est un numéro d'identification unique affiché sur le fuselage de l'avion.

Caractéristiques : - Format : 4 à 6 caractères alphanumériques - Exemples : `N960TB`, `F-HBCD`, `N123AB` - Position : Visible sur les deux flancs (gauche et droite)

2.2 Contraintes Techniques

Normes : - Hauteur minimale : 30 cm - Visibilité : Lettres contrastées - Espacement : 5 cm entre caractères

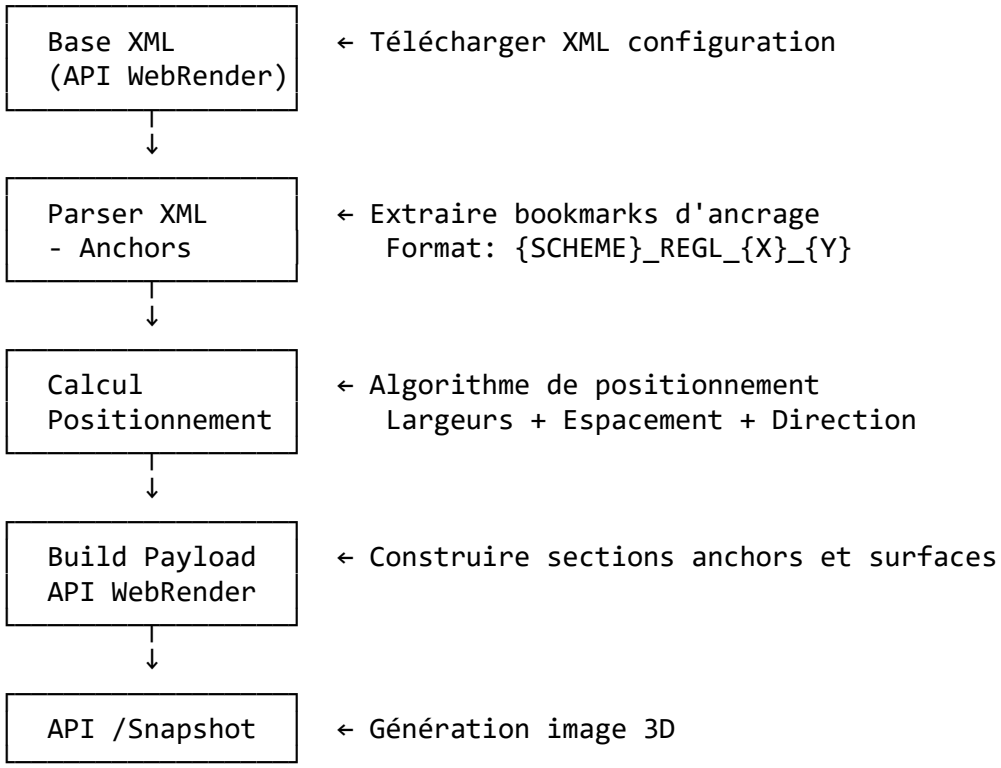
Complexité : - Largeurs variables : 'I' = 10 cm, 'M' = 30 cm - Positions de départ variables selon la configuration - Calcul dynamique requis

Règle fondamentale : Toutes les données doivent être lues dynamiquement depuis le XML (pas de valeurs hardcodées).

---

3. Architecture

3.1 Flux de Données



3.2 Coordonnées 3D

Repère avion : - **Axe X** : Longitudinal (avant → arrière) - **Axe Y** : Vertical (bas → haut) - **Axe Z** : Transversal (gauche → droite)

Surfaces d'immatriculation : - **RegL** : Registration Left (côté gauche) - **RegR** : Registration Right (côté droit)

---

4. Parsing des Bookmarks XML

4.1 Format des Bookmarks

Les bookmarks XML définissent les points de départ (anchors) pour positionner les lettres.

Format actuel (V0.6+) :

```
<Bookmark label="ZEPHIR_REGL_0.34_0.0" />
<Bookmark label="ZEPHIR_REGR_-0.34_0.0" />
```

Structure :

```
{PAINTSCHEME}_REGL_{StartX}_{Y}
{PAINTSCHEME}_REGR_{StartX}_{Y}
```

Composants : - `PAINTSCHEME` : Nom du schéma de peinture (ex: ZEPHIR, TEHUANO) - `REGL` / `REGR` : Left ou Right - `StartX` : Position X de la première lettre (en mètres) - `Y` : Position verticale commune à toutes les lettres (en mètres)

Notes : - Le signe de `StartX` encode la position (positif = droite, négatif = gauche) - Pour RegL : typiquement positif (ex: `0.34`) - Pour RegR : typiquement négatif (ex: `-0.34`)

4.2 Algorithme d'Extraction

Pseudo-code :

```
FONCTION extractAnchors(xmlDoc, paintScheme):
    anchors = { Left: null, Right: null, Y: 0.0 }

    # Rechercher tous les bookmarks
    bookmarks = xmlDoc.querySelectorAll('Bookmark, ConfigurationBookmark')

    targetLeft = paintScheme.toUpperCase() + "_REGL"
    targetRight = paintScheme.toUpperCase() + "_REGR"

    POUR CHAQUE bookmark DANS bookmarks:
        label = bookmark.getAttribute('label') OU bookmark.getAttribute('name
')
        label = label.toUpperCase()
```

```
        SI label commence par targetLeft:
            # Parser SCHEME_REGL_X_Y
            parts = label.split('_')
            SI longueur(parts) >= 4:
                startX = parseFloat(parts[2])  # 3ème élément
                y = parseFloat(parts[longueur(parts) - 1])  # Dernier élément
                anchors.Left = { Start: startX, Direction: 1.0 }
                anchors.Y = y

        SINON SI label commence par targetRight:
            parts = label.split('_')
            SI longueur(parts) >= 4:
                startX = parseFloat(parts[2])
                anchors.Right = { Start: startX, Direction: 1.0 }

    # Fallback si rien trouvé
    SI anchors.Left est null:
        anchors.Left = { Start: 0.34, Direction: 1.0 }
        anchors.Right = { Start: -0.34, Direction: 1.0 }
        anchors.Y = 0.0

    RETOURNER anchors
```

Exemple de parsing :

Bookmark : `ZEPHIR_REGL_0.34_0.0` - `parts = ["ZEPHIR", "REGL", "0.34", "0.0"]` - `startX = parts[2] = 0.34` - `y = parts[3] = 0.0` - Direction = `1.0` (écriture de gauche à droite)

---

5. Algorithme de Positionnement

5.1 Principe

Objectif : Calculer la position X absolue de chaque lettre.

Règle :

```
position_première_lettre = Start X
position_lettre_i = Start X + (offset × Direction)
```

Espacement : 5 cm (0.05 m) entre le bord droit d'une lettre et le bord gauche de la suivante.

5.2 Largeurs des Caractères

Les lettres n'ont pas toutes la même largeur :

Caractère	Largeur (m)	Caractère	Largeur (m)

Caractère	Largeur (m)	Caractère	Largeur (m)
I, 1, -	**0.10**	A-Z (sauf I,M,W)	0.20
M, W	**0.30**	0-9 (sauf 1)	0.20
DEFAULT	**0.20**		

Constantes :

```
SPACING = 0.05  // 5 cm
CHAR_WIDTHS = {
    'M': 0.30, 'W': 0.30,
    'I': 0.10, '1': 0.10, '-': 0.10,
    // A-Z sauf I, M, W : 0.20
    // 0-9 sauf 1 : 0.20
    DEFAULT: 0.20
}
```

5.3 Formule de Calcul

Pour la première lettre (index 0) :

```
position_0 = Start X
```

Pour les lettres suivantes (index i > 0) :

```
offset = (largeur_première_lettre / 2)
       + SOMME(largeurs_complètes des lettres 1 à i-1)
       + (nombre_espaces × SPACING)
       + (largeur_lettre_i / 2)

position_i = Start X + (offset × Direction)
```

Détail du calcul :

1. `half_first` = largeur lettre[0] ÷ 2
2. `sum_widths_middle` = SOMME des largeurs complètes (lettres 1 à i-1)
3. `num_spaces` = i
4. `half_current` = largeur lettre[i] ÷ 2
5. `offset = half_first + sum_widths_middle + (num_spaces × SPACING) + half_current`

5.4 Algorithme Complet

```
FONCTION calculateTransformsAbsolute(immatriculation, startX, direction):
    positions = []
    SPACING = 0.05

    POUR i DE 0 À longueur(immatriculation) - 1:
        SI i == 0:
            # Première lettre : centre exactement à startX
```

```
            positions.ajouter(startX)
        SINON:
            # 1. Moitié première lettre
            char0 = immatriculation[0]
            width0 = CHAR_WIDTHS[char0] OU DEFAULT
            half_first = width0 / 2.0

            # 2. Somme largeurs lettres intermédiaires (1 à i-1)
            sum_widths = 0.0
            POUR j DE 1 À i - 1:
                charJ = immatriculation[j]
                widthJ = CHAR_WIDTHS[charJ] OU DEFAULT
                sum_widths += widthJ

            # 3. Nombre d'espaces
            num_spaces = i

            # 4. Moitié lettre actuelle
            charI = immatriculation[i]
            widthI = CHAR_WIDTHS[charI] OU DEFAULT
            half_current = widthI / 2.0

            # 5. Calcul offset total
            offset = half_first + sum_widths + (num_spaces × SPACING) + half_
current

            # 6. Position absolue
            position = startX + (offset × direction)

            # 7. Arrondir à 4 décimales
            position = arrondir(position, 4)

            positions.ajouter(position)

    RETOURNER positions
```

Exemple visuel :

```
Immatriculation : N 9 6 0 T B
Largeurs        : 0.20  0.20  0.20  0.20  0.20  0.20
Espacement      :    0.05   0.05   0.05   0.05   0.05

Start X = 0.34, Direction = 1.0

Position[0] (N) = 0.34
Position[1] (9) = 0.34 + (0.10 + 0.0 + 0.05 + 0.10) × 1.0 = 0.59
Position[2] (6) = 0.34 + (0.10 + 0.20 + 0.10 + 0.10) × 1.0 = 0.84
Position[3] (0) = 0.34 + (0.10 + 0.40 + 0.15 + 0.10) × 1.0 = 1.09
```

```
Position[4] (T) = 0.34 + (0.10 + 0.60 + 0.20 + 0.10) × 1.0 = 1.34
Position[5] (B) = 0.34 + (0.10 + 0.80 + 0.25 + 0.10) × 1.0 = 1.59
```

---

6. Construction du Payload API

6.1 Structure Minimale

Le payload API pour le positionnement contient deux sections critiques :

```
{
  "scene": [{
    "database": "{DATABASE-GUID}",
    "configuration": "...",
    "surfaces": [
      {
        "tag": "RegL",
        "labels": [
          { "index": 0, "translation": { "x": 0.34, "y": 0.0 } },
          { "index": 1, "translation": { "x": 0.59, "y": 0.0 } },
          ...
        ]
      },
      {
        "tag": "RegR",
        "labels": [
          { "index": 0, "translation": { "x": -0.34, "y": 0.0 } },
          { "index": 1, "translation": { "x": -0.59, "y": 0.0 } },
          ...
        ]
      }
    ]
  }],
  "mode": { "image": { "camera": "{CAMERA-GUID}" } },
  "renderParameters": { ... },
  "encoder": { ... }
}
```

Note : Les sections `materials` et `materialMultiLayers` gèrent les textures et couleurs (voir autre documentation).

6.2 Génération de la Section `surfaces`

Algorithme :

```
FONCTION generateSurfaces(immatriculation, anchors):
    surfaces = []

    # Calculer positions Left
    positionsLeft = calculateTransformsAbsolute(
```

```
    immatriculation,
    anchors.Left.Start,
    anchors.Left.Direction
)

# Calculer positions Right
positionsRight = calculateTransformsAbsolute(
    immatriculation,
    anchors.Right.Start,
    anchors.Right.Direction
)

# Construire surface Left
labelsL = []
POUR i DE 0 À longueur(positionsLeft) - 1:
    labelsL.ajouter({
        "index": i,
        "translation": {
            "x": positionsLeft[i],
            "y": anchors.Y
        }
    })

surfaces.ajouter({
    "tag": "RegL",
    "labels": labelsL
})

# Construire surface Right
labelsR = []
POUR i DE 0 À longueur(positionsRight) - 1:
    labelsR.ajouter({
        "index": i,
        "translation": {
            "x": positionsRight[i],
            "y": anchors.Y
        }
    })

surfaces.ajouter({
    "tag": "RegR",
    "labels": labelsR
})

RETOURNER surfaces
```

6.3 Appel API

Endpoint : `POST https://wr-lumis3d.lumiscaphe.com/Snapshot`

Headers :

```
Content-Type: application/json
Authorization: Bearer {token}
```

Body : Payload JSON complet (avec surfaces calculées)

---

7. Exemple Concret

7.1 Données d'Entrée

Configuration : - Immatriculation : `N960TB` - Paint Scheme : `Zephir` - Bookmarks XML : - `ZEPHIR_REGL_0.34_0.0` → Start = 0.34, Y = 0.0 - `ZEPHIR_REGR_-0.34_0.0` → Start = -0.34, Y = 0.0

7.2 Calcul des Positions Left

Largeurs : - N : 0.20 m - 9 : 0.20 m - 6 : 0.20 m - 0 : 0.20 m - T : 0.20 m - B : 0.20 m

Calcul détaillé :

```
Lettre[0] (N) :
  position = 0.34

Lettre[1] (9) :
  half_first = 0.20 / 2 = 0.10
  sum_widths = 0.0
  num_spaces = 1
  half_current = 0.20 / 2 = 0.10
  offset = 0.10 + 0.0 + (1 × 0.05) + 0.10 = 0.25
  position = 0.34 + 0.25 = 0.59

Lettre[2] (6) :
  half_first = 0.10
  sum_widths = 0.20
  num_spaces = 2
  half_current = 0.10
  offset = 0.10 + 0.20 + (2 × 0.05) + 0.10 = 0.50
  position = 0.34 + 0.50 = 0.84

Lettre[3] (0) :
  offset = 0.10 + 0.40 + 0.15 + 0.10 = 0.75
  position = 0.34 + 0.75 = 1.09

Lettre[4] (T) :
  offset = 0.10 + 0.60 + 0.20 + 0.10 = 1.00
  position = 0.34 + 1.00 = 1.34
```

```
Lettre[5] (B) :
  offset = 0.10 + 0.80 + 0.25 + 0.10 = 1.25
  position = 0.34 + 1.25 = 1.59
```

Résultat : `[0.34, 0.59, 0.84, 1.09, 1.34, 1.59]`

7.3 Calcul des Positions Right

Même algorithme avec `Start = -0.34` :

Résultat : `[-0.34, -0.59, -0.84, -1.09, -1.34, -1.59]`

7.4 Payload JSON Résultant


```json
{
  "surfaces": [
    {
      "tag": "RegL",
      "labels": [
        { "index": 0, "translation": { "x": 0.34, "y": 0.0 } },
        { "index": 1, "translation": { "x": 0.59, "y": 0.0 } },
        { "index": 2, "translation": { "x": 0.84, "y": 0.0 } },
        { "index": 3, "translation": { "x": 1.09, "y": 0.0 } },
        { "index": 4, "translation": { "x": 1.34, "y": 0.0 } },
        { "index": 5, "translation": { "x": 1.59, "y": 0.0 } }
      ]
    },
    {
      "tag": "RegR",
      "labels": [
        { "index": 0, "translation": { "x": -0.34, "y": 0.0 } },
        { "index": 1, "translation": { "x": -0.59, "y": 0.0 } },
        { "index": 2, "translation": { "x": -0.84, "y": 0.0 } },
        { "index": 3, "translation": { "x": -1.09, "y": 0.0 } },
        { "index": 4, "translation": { "x": -1.34, "y": 0.0 } },
        { "index": 5, "translation": { "x": -1.59, "y": 0.0 } }
      ]
    }
  ]
}
```