INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES STRASBOURG

AUTODESK
Make Anything

Développer des plugins Civil 3D assisté par l'IA

Notice de rédaction de prompts optimisés pour Cursor

BERSON Guillaume
PELISSIER Jean

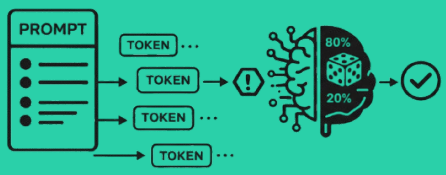
Janvier 2026

Qu’est-ce qu’un prompt ?

Le prompt n’est pas une simple question, mais une consigne qui guide tout le raisonnement que va suivre une IA.
Sa qualité conditionne directement la fiabilité et la pertinence du plugin généré.

Comment l’IA « comprend » un prompt

L’IA ne comprend pas le besoin humain.
Elle fonctionne en découpant le prompt en tokens (morceaux de mots) et en évaluant, à chaque étape, la probabilité du mot ou de l’action suivante.



Cursor analyse :

- Les mots utilisés
- leur ordre
- leur précision
- le contexte

Si une information est absente ou ambiguë, l’IA comble le vide en choisissant la solution qui est « statistiquement » la plus probable, ce qui peut l’amener à prendre une mauvaise direction.

Quel impact a-t-il sur le développement du plugin?

Temps de développement

Les prompts structurés dirigent l’IA dans la bonne direction dès le départ.
Cela évite les allers-retours inutiles liés aux incompréhensions, qui peuvent rallonger inutilement le temps de développement du plugin.

Qualité du code

Un prompt bien formulé favorise un développement clair et structuré.
Il conduit à un code mieux organisé, avec des commentaires pertinents et une séparation logique des fonctionnalités, facilitant ainsi la relecture et la compréhension par l’utilisateur.

Boucles d’erreurs

Si la logique du prompt n'est pas bien structurée, on s'expose à de possibles boucles d'erreurs dans lesquelles l'IA ne trouve pas ses erreurs et reste bloquée sur de fausses pistes. Cela donne donc un plugin même pas compilable.

Etapes de rédaction d'un prompt optimisé pour le développement de plugins Civil3D

1 Initialisation de l'environnement de développement

Avant de décrire la logique du plugin, il est nécessaire de demander à l'IA de structurer correctement l'environnement de développement.
Le prompt d'initialisation définit :

- L'environnement de travail : système d'exploitation, versions de Civil 3D et du SDK .NET.
- Les chemins de développement : racine du projet et accès aux bibliothèques Civil 3D (API).
- La structure attendue du projet :
    1. création du projet et des fichiers associés.
    2. organisation du code principal et gestion des erreurs.
    3. règles de compilation et résultat attendu (DLL).
    4. objectifs finaux et contraintes imposées à l'IA.

Sans un environnement défini, la génération et la compilation du plugin ne peuvent pas aboutir.

2 Contexte

On demande à l'IA de se comporter comme un développeur C# expert en plugins Civil 3D, chargé de concevoir un outil optimisant une mission précise.
Ce cadrage oriente ses choix techniques et sa logique de développement.

3 Objectif

Définir clairement le comportement attendu et l'objectif du plugin, en se concentrant sur le besoin métier plutôt que sur les aspects techniques.

4 Entrées

Expliciter les entrées fournies par le système ou l'utilisateur (fichiers, sélections, paramètres) et la manière dont le plugin les récupère.

7 Contraintes

Préciser les contraintes techniques et fonctionnelles à respecter (compatibilité Civil 3D, performances, gestion des erreurs), afin d'éviter que l'IA propose des solutions inadaptées.

6 Traitement

Décrire les principales étapes de traitement du plugin, dans un ordre logique, afin de guider le raisonnement de l'IA.
Éviter de décrire une logique de traitement si on ne la maîtrise pas

5 Sorties

Définir les sorties attendues du plugin (objets créés, messages, résultats visibles) et la forme sous laquelle elles sont produites dans Civil 3D.

8 Critères de validation

Définir des critères de validation clairs permettant de vérifier que le plugin répond correctement au besoin.
Ces critères portent sur le bon déroulement du traitement, la conformité du résultat obtenu, la clarté des messages affichés dans la ligne de commande.

Exemple : Prompt optimisé pour un plugin simple (création de polyligne)

CONTEXTE ENVIRONNEMENT DE TRAVAIL :

- OS : Windows 10 / 11
- Terminal par défaut : PowerShell
- .NET SDK installé : .NET 8 (dotnet CLI)
- AutoCAD version : 2026
- Civil 3D version : 2026

CONTEXTE DE DÉVELOPPEMENT :

- Racine principale du workspace : T:\Jean PELISSIER\PRT\Projet
- Tous les projets C# sont contenus dans cette racine.
- Chaque plugin doit être un projet SDK-style (Microsoft.NET.Sdk) ciblant net8.0-windows, plateforme x64.
- Les références AutoCAD/Civil 3D doivent être ajoutées directement dans chaque .csproj :
  - AutoCAD DLLs : C:\Program Files\Autodesk\AutoCAD 2026\ : acdbmgd.dll; acmgd.dll; accoremgd.dll
  - Civil 3D DLLs : C:\Program Files\Autodesk\AutoCAD 2026\C3D : AeccDbMgd.dll;AeccLandMgd.dll;AeccRoadwayMgd.dll ; AecBaseMgd.dll

Toutes les références doivent avoir <Private>False</Private> pour éviter la copie des DLLs.

STRUCTURE ATTENDUE :

1/ Création du projet :

- Nouveau dossier : T:\Jean PELISSIER\PRT\Projet\Plugin
- Créer un projet SDK-style net8.0-windows (C#, x64)
- Ajouter les fichiers nécessaires :
  - o Plugin.csproj avec toutes les références AutoCAD/Civil 3D configurées
  - o dossier /Properties avec AssemblyInfo.cs (optionnel mais recommandé)
  - o .vscode\tasks.json pour build rapide (dotnet restore + build)
  - o dossier /Commands avec le fichier principal de commande
  - o .editorconfig, .gitignore
  - o README.md avec documentation

2/ Code principal :

- Créer une classe Commands dans /Commands/Plugin.cs
- Gestion des erreurs (try/catch) et messages utilisateur clairs
- Utiliser Autodesk.AutoCAD.ApplicationServices.Editor pour les interactions utilisateur

3/ Compilation :

- Restaurer les dépendances : dotnet restore
- Compiler : dotnet build -c Release -p:Platform=x64
- Résultat attendu : build réussi, DLL dans bin\x64\Release\net8.0-windows\Plugin.dll
- Aucune exécution de Civil 3D ne doit être lancée automatiquement.

4/ Objectif final :

- Build totalement automatique
- Fichiers structurés proprement pour tests et évolutions
- Cursor doit corriger toutes les erreurs de build jusqu'à obtention d'un ""Build succeeded""
- Aucun lancement externe de Civil 3D n'est autorisé
- Utilisation exclusive de dotnet CLI (pas de MSBuild.exe)
- Chaque projet est autonome et contient toutes ses références dans son .csproj

DEMANDE :

1/ Contexte :

Tu es un développeur C# expert en plugins AutoCAD / Civil 3D. Tu dois concevoir un plugin Civil 3D fiable et robuste, destiné à un usage métier, en respectant les contraintes de l'API Autodesk et les bonnes pratiques .NET.

2/ Objectif :

Développer un plugin Civil 3D permettant de créer une polyligne 2D à partir de points définis par l'utilisateur, soit par saisie manuelle de coordonnées, soit par sélection graphique dans le dessin.

3/ Entrées :

Mode de saisie choisi par l'utilisateur :

- Saisie manuelle : coordonnées (X, Y) entrées dans la ligne de commande.
- Sélection graphique : points sélectionnés dans le dessin.
- Nombre de points variable (minimum 2).
- Interaction utilisateur via la ligne de commande Civil 3D.

4/ Sorties :

- Une polyligne 2D créée et visible dans le dessin Civil 3D.
- Messages clairs dans la ligne de commande :
  - o Confirmation de création
  - o Nombre de points utilisés
  - o Messages d'erreur explicites si nécessaire

5/ Traitement :

- Lancer la commande Civil 3D dédiée
- Demander à l'utilisateur de choisir le mode de saisie
- Récupérer les points dans l'ordre de saisie ou de sélection
- Vérifier la validité des points
- Créer une polyligne 2D AutoCAD à partir des points fournis
- Ajouter la polyligne dans le ModelSpace du dessin courant
- Gérer proprement les transactions et libérations de ressources.

6/ Contraintes :

- Compatibilité Civil 3D / AutoCAD 2026.
- Utilisation exclusive des API officielles Autodesk
- Aucune dépendance externe.
- Gestion robuste des erreurs utilisateur
- Ne pas créer de géométrie si les conditions minimales ne sont pas remplies.

7/ Critères de validation :

- Le plugin compile sans erreur et se charge correctement dans Civil 3D.
- La commande est disponible et exécutable.
- La polyligne générée correspond exactement aux points fournis.
- Les messages utilisateur sont clairs, cohérents et exploitables.
- Aucun crash ou comportement instable n'est observé.