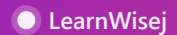


Exercice de code avec l'IA

Construisez l'exemple VisualOperationsStudio de ce module avec ChatGPT ou Claude à partir d'un prompt prêt à l'emploi qui oriente le modèle vers la spécification du module, puis relisez, exécutez et étendez ce qu'il vous renvoie.



Le même module, construit comme vous construirez vraiment.

Toutes les autres sections de ce module expliquent comment fonctionne « Canvas interactif : état, saisie, hit testing et performances ». Celle-ci vous demande de le produire — en pilotant un assistant avec la spécification du module elle-même, puis en vérifiant que le résultat s'exécute et fait ce que décrit l'énoncé du lab. C'est la méthode de travail que vous utiliserez sur de vrais projets, mise en pratique sur un sujet dont vous connaissez déjà la bonne réponse.

Ce que vous demandez

Transformez le bac à sable Canvas en éditeur de topologie pour VisualOperationsStudio. Créez des classes NodeModel et EdgeModel avec des identifiants stables et une géométrie en coordonnées monde, conservez la scène dans un objet TopologyScene stocké par session plutôt que dans un champ statique, et faites de RenderScene la seule méthode qui dessine. Ajoutez les conversions ToWorld et ToScreen pour le décalage de panoramique et le facteur de zoom courants, puis gérez MouseDown pour tester en coordonnées monde l'appartenance du point aux limites d'un nœud et positionner Selected, MouseMove pour faire glisser le nœud sélectionné ou déplacer la vue, et MouseUp pour terminer le geste — chacun modifiant l'état puis appelant RenderScene. Ajoutez un zoom à la molette ou par bouton centré sur le pointeur, ignorez au rendu les nœuds situés hors du rectangle visible, donnez aux utilisateurs du clavier un moyen de sélectionner et de déplacer un nœud, et relevez le trafic de requêtes que produit un glisser ainsi que le point à partir duquel vous déplaceriez la boucle vers du JavaScript client.

Le module demande ces livrables, et le prompt ci-dessous demande exactement ceux-là à l'assistant :

- ✓ NodeModel, EdgeModel et un TopologyScene par session, avec des identifiants stables et aucun état statique mutable
- ✓ Conversions ToWorld et ToScreen utilisées pour le hit testing, afin que la sélection reste correcte après un panoramique et un zoom
- ✓ Gestionnaires MouseDown, MouseMove et MouseUp qui modifient l'état puis appellent l'unique méthode RenderScene
- ✓ Élimination hors champ qui ignore les nœuds hors du rectangle visible, plus un parcours clavier pour sélectionner et déplacer un nœud
- ✓ InteractionNotes.md avec le trafic de requêtes mesuré pendant un glisser et le seuil de bascule côté client que vous appliqueriez

Pourquoi il est conçu ainsi

Le Module 4 a laissé VisualOperationsStudio avec un bac à sable Canvas: une méthode DrawPlayground que l'événement Redraw appelle, reconstruisant de zéro chaque tracé, dégradé et transformation. C'est correct, et c'est totalement inerte. Le personnel d'exploitation veut cliquer sur une pompe, la faire glisser à un endroit sensé, se déplacer dans l'usine et zoomer sur un groupe — et une méthode de dessin ne peut répondre à aucune de ces demandes, parce que le canvas du navigateur détient des pixels et que les pixels ignorent quelle forme est une pompe. Dès qu'une surface devient interactive, elle cesse d'être un moteur de rendu et devient une petite machine à états qui, accessoirement, dessine.

Voici les termes que l'assistant doit maîtriser. Si ce qu'il vous renvoie les emploie de façon approximative, il n'a pas compris le module :

Terme	Ce qu'il signifie ici
Modèle, Saisie, État, Rendu	Quatre couches et une seule règle entre elles: la saisie change l'état, et c'est seulement ensuite qu'une unique méthode de rendu transforme l'état courant en appels de dessin.
Objets de scène	Des classes C# comme NodeModel et EdgeModel qui portent l'identifiant stable, les limites en coordonnées monde, l'état et l'indicateur Selected que les pixels ne peuvent que suggérer.
État de scène par session	La scène appartient à la session d'un seul utilisateur; une liste static mutable donnerait à chaque visiteur les mêmes nœuds et la même sélection.
Événements de saisie de Control	Canvas dérive de Control: les événements souris (down, move, up), clic, double-clic ainsi que les événements tactiles et gestuels comme TouchStart, Tap, Pinch, Rotate, Swipe et Track sont tous disponibles.
Hit testing fondé sur le modèle	L'API Canvas documentée de Wisej n'expose ni isPointInPath ni isPointInStroke: « sur quoi l'utilisateur a-t-il cliqué? » se répond donc en comparant un point à une géométrie qui vous appartient.
Conversion écran vers monde	Si le rendu applique $\text{écran} = \text{monde} \times \text{zoom} + \text{pan}$, alors chaque coordonnée de pointeur entrante doit repasser par l'inverse avant d'être comparée à quoi que ce soit dans le modèle.
Élimination hors champ	Ignorer les objets de scène dont les limites ne coupent pas le rectangle monde visible, pour qu'une grande topologie ne coûte que ce qui est réellement à l'écran.

Construisez-le — choisissez votre assistant

Les deux boutons ouvrent un nouvel onglet avec le **même prompt prêt à l'emploi** déjà rempli. Le prompt oriente le modèle vers la spécification de ce module — un unique fichier Markdown qui contient la leçon, l'énoncé du lab et les livrables — pour qu'il travaille sur le vrai contenu au lieu de deviner de quoi parle le Module 5. Vous pouvez ouvrir vous-même [module-05.md](#) pour voir exactement ce qu'il lira.



Utiliser ChatGPT



Utiliser Claude

Ouvre chatgpt.com avec le prompt de ce module déjà chargé. Fonctionne avec GPT-5 ou Codex ; toute offre capable de naviguer sur le web récupérera la spécification.

Ouvre claude.ai avec le même prompt déjà chargé. Si vous utilisez Claude Code, vous pouvez plutôt le coller dans un terminal ouvert sur un dossier vide.



Envoyez-le tel quel et vous obtenez un projet, pas un extrait

Le prompt se termine en demandant toute la solution sous forme d'un unique `.zip`, avec les étapes pour la compiler et l'exécuter. Téléchargez-le, ouvrez la solution dans Visual Studio, appuyez sur F5 et VisualOperationsStudio s'ouvre sur `localhost`. Il vous faut le SDK .NET et le package NuGet `Wisej-4` ; tout le reste, le prompt demande à l'assistant de l'écrire lui-même.

Je suis le Module 5 - « Canvas interactif : état, saisie, hit testing et performances » - du cours LearnWisej « Dessin et peinture dans Wisej.NET », et je veux construire avec toi l'exemple de ce module. Réponds-moi en français.

Commence par lire la spécification du module. C'est un unique fichier Markdown (en anglais) qui contient la leçon complète, l'énoncé du lab et les livrables :

<https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-05.md>

Construis ensuite un projet Wisej.NET 4 complet que je puisse ouvrir dans Visual Studio :

1. Projet « VisualOperationsStudio », C#, net8.0 ou ultérieur, package NuGet Wisej-4 4.1.x, hôte Kestrel qui appelle app.UseWisej().
2. Construis ce que demande le lab de la spécification : Transformez le bac à sable Canvas en éditeur de topologie pour VisualOperationsStudio. Créez des classes NodeModel et EdgeModel avec des identifiants stables et une géométrie en coordonnées monde, conservez la scène dans un objet TopologyScene stocké par session plutôt que dans un champ statique, et faites de RenderScene la seule méthode qui dessine. Ajoutez les conversions ToWorld et ToScreen pour le décalage de panoramique et le facteur de zoom courants, puis gérez MouseDown pour tester en coordonnées monde l'appartenance du point aux limites d'un nœud et positionner Selected, MouseMove pour faire glisser le nœud sélectionné ou déplacer la vue, et MouseUp pour terminer le geste - chacun modifiant l'état puis appelant RenderScene. Ajoutez un zoom à la molette ou par bouton centré sur le pointeur, ignorez au rendu les nœuds situés hors du rectangle visible, donnez aux utilisateurs du clavier un moyen de sélectionner et de déplacer un nœud, et relevez le trafic de requêtes que produit un glisser ainsi que le point à partir duquel vous déplaceriez la boucle vers du JavaScript client.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - NodeModel, EdgeModel et un TopologyScene par session, avec des identifiants stables et aucun état statique mutable
 - Conversions ToWorld et ToScreen utilisées pour le hit testing, afin que la sélection reste correcte après un panoramique et un zoom
 - Gestionnaires MouseDown, MouseMove et MouseUp qui modifient l'état puis appellent l'unique méthode RenderScene
 - Élimination hors champ qui ignore les nœuds hors du rectangle visible, plus un parcours clavier pour sélectionner et déplacer un nœud
 - InteractionNotes.md avec le trafic de requêtes mesuré pendant un glisser et le seuil de bascule côté client que vous appliqueriez
4. Suis la spécification partout où elle nomme une classe, une propriété ou un événement Wisej.NET - utilise exactement celui-là. Là où elle laisse le choix, dis-moi quelle API tu as choisie et pourquoi.
5. Garde l'autorité côté serveur : l'état et la validation vivent dans .NET, le navigateur affiche et remonte les informations. Ne déplace jamais une décision dans JavaScript parce que c'est plus simple là-bas.
6. Rends le résultat démontrable : je dois pouvoir l'exécuter et voir le comportement dont traite ce module, y compris ce qui se passe quand quelque chose tourne mal.

Avant d'écrire la moindre ligne de code, reformule avec tes propres mots ce que tu vas construire et où vivra chaque élément, pour que je puisse corriger la conception tant que cela ne coûte pas cher.

Quand le code est terminé, rassemble toute la solution dans un unique fichier .zip téléchargeable et donne-moi les étapes exactes pour la compiler et l'exécuter.

Enfin, réponds à ces questions de la spécification en t'appuyant sur ta propre implémentation :

- Un collègue signale que la sélection fonctionne jusqu'à ce qu'il zoome, après quoi les clics tombent sur la mauvaise pompe ou sur rien. Rien n'a changé dans le code de dessin. Quelle unique conversion manque-t-il, et pourquoi le défaut reste-t-il invisible au zoom 1,0 sans panoramique?
- Deux opérateurs ouvrent la topologie en même temps et chacun voit le nœud de l'autre bouger sous son curseur. Dans quoi la scène a-t-elle été stockée, et que cela vous apprend-il sur la place de l'état interactif dans un framework web côté serveur?
- Votre glisser produit une requête à chaque mouvement du pointeur. À partir de quelle mesure cesseriez-vous d'optimiser la boucle serveur pour déplacer l'interaction vers du JavaScript client, et qu'est-ce qui devrait malgré tout rester faisant autorité sur le serveur?

Pose-moi des questions sur tout ce qui est ambigu avant de commencer.

Puis relisez ce qui vous revient

Un code qui compile n'est pas encore un code juste. Les questions intéressantes sont des questions de conception : est-ce bien le contrôle Wisej.NET que la tâche exige, suit-il le cycle de vie que le framework vous offre, l'état de référence est-il sur le serveur ou a-t-il été laissé dans le navigateur parce que c'était pratique ? Lisez le résultat en relecteur, pas en client.

Ce module vous donne la checklist. Répondez à chaque point en vous appuyant sur le code que vous venez de recevoir :

- ✓ Un collègue signale que la sélection fonctionne jusqu'à ce qu'il zoome, après quoi les clics tombent sur la mauvaise pompe ou sur rien. Rien n'a changé dans le code de dessin. Quelle unique conversion manque-t-il, et pourquoi le défaut reste-t-il invisible au zoom 1,0 sans panoramique?
- ✓ Deux opérateurs ouvrent la topologie en même temps et chacun voit le nœud de l'autre bouger sous son curseur. Dans quoi la scène a-t-elle été stockée, et que cela vous apprend-il sur la place de l'état interactif dans un framework web côté serveur?
- ✓ Votre glisser produit une requête à chaque mouvement du pointeur. À partir de quelle mesure cesseriez-vous d'optimiser la boucle serveur pour déplacer l'interaction vers du JavaScript client, et qu'est-ce qui devrait malgré tout rester faisant autorité sur le serveur?

Tout ce que vous avez dû corriger mérite d'être noté. Une courte note des conventions et des décisions retenues, collée dans le premier message de votre prochaine session, vaut plus que n'importe quel prompt astucieux.



Pas encore d'exemple de référence pour ce cours — le vôtre est le premier

Le [dépôt des exemples LearnWisej](#) contient des solutions complètes pour la plupart des cours, mais pas encore pour celui-ci. Il n'y a donc rien à quoi vous comparer et aucun raccourci : la spécification et votre propre relecture sont votre seul filet de sécurité. Gardez ce que vous construisez — et si le résultat est réussi, nous aimerions le voir.

Trois prompts pour aller plus loin

Orientez votre assistant vers le projet que vous venez de construire et essayez ceux-ci. Chacun est un exercice de relecture et d'extension, pas une réécriture.

- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet VisualOperationsStudio pour repérer les modes de défaillance qu'il ne gère pas encore — une saisie erronée ou manquante, une dépendance lente ou indisponible, un traitement qui aboutit après que l'utilisateur a changé de page, et deux actions qui se chevauchent. Pour chacun, dis-moi d'abord s'il s'agit d'un risque réel dans Wisej.NET 4 et pourquoi, et seulement ensuite écris la correction. Là où tu estimes qu'aucune gestion n'est nécessaire, dis-le et explique ton raisonnement au lieu d'ajouter du code défensif."*
- ✓ **Ajoutez le deuxième cas.** *"Étends ce projet VisualOperationsStudio avec un deuxième cas qui suit le même modèle que le premier — une autre entité, un autre écran ou un autre contrôle de même forme, câblé de la même manière. Extrais dans une classe de base ou un service ce qui est réellement partagé, mais seulement là où cela se justifie ; deux choses similaires n'appellent pas toujours une abstraction. Montre-moi le diff et justifie chaque élément que tu as extrait."*
- ✓ **Prouvez que ça fonctionne vraiment.** *"Passe en revue ce projet livrable par livrable par rapport à la spécification du module sur <https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-05.md> et montre-moi la preuve que chacun est réellement terminé — le chemin dans le code, où cliquer et ce que je dois voir. Là où un livrable n'est que partiellement satisfait, dis-le clairement au lieu de le défendre. Ensuite, ajoute ce qui manque."*



Emportez-le dans le lab

Le lab pratique qui clôt ce module demande les mêmes livrables que ceux que vous venez de faire produire à l'assistant. Servez-vous de ce que vous avez ici comme forme de votre réponse et comme preuve à l'appui — pas comme la réponse elle-même. Le lab est noté sur votre raisonnement, et cette partie doit venir de vous.