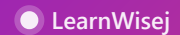


Esercizio di coding con l'IA

Costruisci il progetto VisualOperationsStudio di questo modulo con ChatGPT o Claude partendo da un prompt già pronto che indirizza il modello alla specifica del modulo, poi rivedi, esegui ed estendi quello che ti restituisce.



Lo stesso modulo, costruito come costruirai davvero le cose.

Tutte le altre sezioni di questo modulo spiegano come funziona "Canvas interattivo: stato, input, hit testing e prestazioni". Questa ti chiede di produrlo: dirigendo un assistente con la specifica del modulo stesso, poi verificando che il risultato giri e faccia quello che descrive la traccia del lab. È il flusso di lavoro che userai sul lavoro vero, provato su qualcosa di cui conosci già la risposta giusta.

Che cosa stai chiedendo

Trasforma il playground Canvas in un editor di topologia per VisualOperationsStudio. Crea le classi NodeModel ed EdgeModel con ID stabili e geometria in coordinate mondo, tieni la scena in un oggetto TopologyScene memorizzato per sessione invece che in un campo statico, e fai di RenderScene l'unico metodo che disegna. Aggiungi le conversioni ToWorld e ToScreen per l'offset di pan e il fattore di zoom correnti, poi gestisci MouseDown per fare hit testing sui limiti dei nodi in coordinate mondo e impostare Selected, MouseMove per trascinare il nodo selezionato o spostare il viewport, e MouseUp per chiudere il gesto — ciascuno cambia lo stato e poi chiama RenderScene. Aggiungi lo zoom con la rotella o con pulsanti attorno al puntatore, salta i nodi fuori dal rettangolo visibile durante il rendering, dai a chi usa la tastiera un modo per selezionare e spostare un nodo, e registra il traffico di richieste prodotto da un trascinamento più il punto a partire dal quale sposteresti il loop su JavaScript lato client.

Il modulo chiede questi deliverable, e il prompt qui sotto li chiede all'assistente esattamente così:

- ✓ NodeModel, EdgeModel e un TopologyScene per sessione con ID stabili e nessuno stato statico mutabile
- ✓ Conversioni ToWorld e ToScreen usate per l'hit testing, così che la selezione resti corretta dopo pan e zoom
- ✓ Handler di MouseDown, MouseMove e MouseUp che cambiano lo stato e poi chiamano l'unico metodo RenderScene
- ✓ Culling del viewport che salta i nodi fuori dal rettangolo visibile, più un percorso da tastiera per selezionare e spostare
- ✓ InteractionNotes.md con il traffico di richieste misurato durante un trascinamento e la soglia oltre la quale passeresti al lato client

Perché è fatto così

Il Modulo 4 ha lasciato VisualOperationsStudio con un playground Canvas: un metodo DrawPlayground che l'evento Redraw richiama, ricostruendo da zero ogni path, gradiente e trasformazione. È corretto, ed è completamente inerte. Il personale di operations vuole fare clic su una pompa, trascinarla dove ha senso, spostarsi sull'impianto con il pan e zoomare su un gruppo — e un metodo di disegno non può rispondere a nessuna di queste richieste, perché il canvas del browser contiene pixel e i pixel non sanno quale forma sia una pompa. Nel momento in cui una superficie diventa interattiva smette di essere un renderer e diventa una piccola macchina a stati che, per inciso, rende.

Questi sono i termini che l'assistente deve azzeccare. Se quello che ti restituisce li usa con leggerezza, non ha capito il modulo:

Termine	Che cosa significa qui
Model, Input, State, Render	Quattro livelli con una sola regola fra loro: l'input cambia lo stato, e solo allora un unico metodo di render trasforma lo stato corrente in chiamate di disegno.
Oggetti della scena	Classi C# come NodeModel ed EdgeModel che portano l'ID stabile, i limiti in coordinate mondo, lo stato e il flag Selected che i pixel possono soltanto suggerire.
Stato della scena per sessione	La scena appartiene alla sessione di un singolo utente; una lista static mutabile darebbe a ogni visitatore gli stessi nodi e la stessa selezione.
Eventi di input del Control	Canvas deriva da Control, quindi mouse down/move/up, click, doppio click e gli eventi touch e gesture come TouchStart, Tap, Pinch, Rotate, Swipe e Track sono tutti disponibili.
Hit testing basato sul modello	L'API Canvas documentata di Wisej non espone isPointInPath né isPointInStroke, quindi alla domanda «su che cosa ha fatto clic l'utente?» si risponde confrontando un punto con la geometria che possiedi tu.
Conversione da schermo a mondo	Se il rendering applica schermo = mondo × zoom + pan, allora ogni coordinata del puntatore in arrivo va ripassata attraverso l'inversa prima di essere confrontata con qualsiasi cosa nel modello.
Culling del viewport	Saltare gli oggetti della scena i cui limiti non intersecano il rettangolo di mondo visibile, così che una topologia grande costi solo ciò che è davvero sullo schermo.

Costruiscilo — scegli il tuo assistente

Entrambi i pulsanti aprono una nuova scheda con lo **stesso prompt già pronto**. Il prompt indirizza il modello alla specifica di questo modulo — un unico file Markdown con la lezione, la traccia del lab e i deliverable — così lavora sul materiale vero invece di tirare a indovinare che cosa sia il Modulo 5. Puoi aprire tu stesso [module-05.md](#) per vedere esattamente che cosa leggerà.



Usa ChatGPT



Usa Claude

Apri chatgpt.com con il prompt di questo modulo già caricato. Funziona con GPT-5 o Codex; qualsiasi piano che possa navigare il web recupererà la specifica.

Apri claude.ai con lo stesso prompt già caricato. Chi usa Claude Code può invece incollarlo in un terminale dentro una cartella vuota.



Mandalo così com'è e ottieni un progetto, non un frammento

Il prompt si chiude chiedendo l'intera soluzione in un unico [.zip](#) con i passi per compilarla ed eseguirla. Scaricalo, apri la soluzione in Visual Studio, premi F5 e VisualOperationsStudio si apre su [localhost](#). Ti servono l'SDK .NET e il pacchetto NuGet [Wisej-4](#); tutto il resto il prompt chiede all'assistente di scriverlo da sé.

Sto seguendo il Modulo 5 - "Canvas interattivo: stato, input, hit testing e prestazioni" - del corso LearnWisej "Disegno e grafica in Wisej.NET", e voglio costruire con te il progetto del modulo. Rispondimi in italiano.

Comincia leggendo la specifica del modulo. È un unico file Markdown (in inglese) che contiene l'intera lezione, la traccia del lab e i deliverable:

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

Poi costruisci un progetto Wisej.NET 4 completo che io possa aprire in Visual Studio:

1. Progetto "VisualOperationsStudio", C#, net8.0 o successivo, pacchetto NuGet Wisej-4 4.1.x, host Kestrel che chiama `app.UseWisej()`.
2. Costruisci quello che chiede il lab della specifica: Trasforma il playground Canvas in un editor di topologia per VisualOperationsStudio. Crea le classi `NodeModel` ed `EdgeModel` con ID stabili e geometria in coordinate mondo, tieni la scena in un oggetto `TopologyScene` memorizzato per sessione invece che in un campo statico, e fai di `RenderScene` l'unico metodo che disegna. Aggiungi le conversioni `ToWorld` e `ToScreen` per l'offset di pan e il fattore di zoom correnti, poi gestisci `MouseDown` per fare hit testing sui limiti dei nodi in coordinate mondo e impostare `Selected`, `MouseMove` per trascinare il nodo selezionato o spostare il viewport, e `MouseUp` per chiudere il gesto - ciascuno cambia lo stato e poi chiama `RenderScene`. Aggiungi lo zoom con la rotella o con pulsanti attorno al puntatore, salta i nodi fuori dal rettangolo visibile durante il rendering, dai a chi usa la tastiera un modo per selezionare e spostare un nodo, e registra il traffico di richieste prodotto da un trascinamento più il punto a partire dal quale sposteresti il loop su JavaScript lato client.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - `NodeModel`, `EdgeModel` e un `TopologyScene` per sessione con ID stabili e nessuno stato statico mutabile
 - Conversioni `ToWorld` e `ToScreen` usate per l'hit testing, così che la selezione resti corretta dopo pan e zoom
 - Handler di `MouseDown`, `MouseMove` e `MouseUp` che cambiano lo stato e poi chiamano l'unico metodo `RenderScene`
 - Culling del viewport che salta i nodi fuori dal rettangolo visibile, più un percorso da tastiera per selezionare e spostare
 - `InteractionNotes.md` con il traffico di richieste misurato durante un trascinamento e la soglia oltre la quale passeresti al lato client
4. Segui la specifica ovunque nomini una classe, una proprietà o un evento di Wisej.NET - usa esattamente quello. Dove lascia una scelta aperta, dimmi quale API hai scelto e perché.
5. Tieni l'autorità sul server: stato e validazione vivono in .NET, il browser disegna e riferisce. Non spostare mai una decisione in JavaScript solo perché lì è più comodo.
6. Rendi il risultato dimostrabile: devo poterlo eseguire e vedere il comportamento di cui parla questo modulo, compreso che cosa succede quando qualcosa va storto.

Prima di scrivere codice, riformula con parole tue che cosa costruirai e dove starà ogni pezzo, così posso correggere il progetto finché costa poco.

Quando il codice è pronto, impacchetta l'intera soluzione in un unico file .zip scaricabile e dammi i passi esatti per compilarla ed eseguirla.

Infine rispondi a queste domande della specifica, usando come prova la tua implementazione:

- Un collega segnala che la selezione funziona finché non zooma, dopodiché i clic cadono sulla pompa sbagliata o nel vuoto. Nel codice di disegno non è cambiato nulla. Quale singola

conversione manca, e perché il difetto resta invisibile a zoom 1.0 senza pan?

- Due operatori aprono la topologia nello stesso momento e ciascuno vede muoversi sotto il proprio cursore il nodo dell'altro. In che cosa è stata memorizzata la scena, e che cosa ti dice questo su dove debba stare lo stato interattivo in un framework web lato server?
- Il tuo trascinamento produce una richiesta per ogni movimento del puntatore. A quale punto misurato smetteresti di ottimizzare il loop lato server per spostare invece l'interazione su JavaScript lato client, e che cosa dovrebbe comunque restare autorevole sul server?

Chiedimi qualsiasi cosa non ti sia chiara prima di cominciare.

Poi rivedi quello che ti è tornato indietro

Codice che compila non è ancora codice giusto. Le domande interessanti sono domande di progetto: è questo il controllo Wisej.NET che il compito richiede davvero, segue il ciclo di vita che il framework ti offre, lo stato autorevole sta sul server o è stato parcheggiato nel browser perché era comodo? Leggi il risultato da revisore, non da cliente.

Questo modulo ti dà la checklist. Rispondi a ciascuna domanda guardando il codice che hai appena ricevuto:

- ✓ Un collega segnala che la selezione funziona finché non zooma, dopodiché i clic cadono sulla pompa sbagliata o nel vuoto. Nel codice di disegno non è cambiato nulla. Quale singola conversione manca, e perché il difetto resta invisibile a zoom 1.0 senza pan?
- ✓ Due operatori aprono la topologia nello stesso momento e ciascuno vede muoversi sotto il proprio cursore il nodo dell'altro. In che cosa è stata memorizzata la scena, e che cosa ti dice questo su dove debba stare lo stato interattivo in un framework web lato server?
- ✓ Il tuo trascinamento produce una richiesta per ogni movimento del puntatore. A quale punto misurato smetteresti di ottimizzare il loop lato server per spostare invece l'interazione su JavaScript lato client, e che cosa dovrebbe comunque restare autorevole sul server?

Tutto ciò che hai dovuto correggere vale la pena di scriverlo. Una breve nota con le convenzioni e le decisioni a cui sei arrivato, incollata nel primo messaggio della sessione successiva, vale più di qualsiasi prompt ingegnoso.



Per questo corso non c'è ancora un esempio di riferimento — il tuo è il primo

Il [repository degli esempi LearnWisej](#) contiene build già fatte per quasi tutti i corsi, ma non ancora per questo. Significa che non c'è niente con cui confrontarsi e nessuna scorciatoia: la specifica e la tua revisione sono tutta la rete di sicurezza. Tieni quello che costruisci — e se viene bene, ci farebbe piacere vederlo.

Tre prompt per andare oltre

Punta il tuo assistente sul progetto che hai appena costruito e prova questi. Ognuno è un esercizio di revisione ed estensione, non una riscrittura.

- ✓ **Fallo sopravvivere a una brutta giornata.** *"Esamina questo progetto VisualOperationsStudio cercando le modalità di guasto che non gestisce ancora — input sbagliato o mancante, una dipendenza lenta o non disponibile, lavoro che arriva dopo che l'utente ha cambiato pagina, e due*

azioni che si accavallano. Per ciascuna dimmi prima se è un rischio reale in Wisej.NET 4 e perché, e solo dopo scrivi la correzione. Dove decidi che non serve gestire nulla, dillo e spiega il ragionamento invece di aggiungere codice difensivo."

- ✓ **Aggiungi il secondo caso.** "Estendi questo progetto VisualOperationsStudio con un secondo caso che segue lo stesso schema del primo — un'altra entità, schermata o controllo della stessa forma, collegato allo stesso modo. Estrai in una classe base o in un servizio ciò che è davvero condiviso, ma solo dove se lo merita; due cose simili non chiedono sempre un'astrazione. Mostrami il diff e giustifica ogni cosa che hai estratto."
- ✓ **Dimostra che funziona davvero.** "Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-05.md> e mostrami la prova che ciascuno è davvero fatto — il percorso nel codice, su che cosa cliccare e che cosa dovrei vedere. Dove un deliverable è soddisfatto solo in parte, dillo chiaramente invece di arrampicarti sugli specchi. Poi aggiungi quello che manca."



Portalo dentro il lab

Il lab pratico che chiude questo modulo chiede gli stessi deliverable che hai appena fatto produrre all'assistente. Usa quello che hai qui come forma della tua risposta e come prova a sostegno — non come la risposta in sé. Il lab viene valutato sul tuo ragionamento, e quella parte deve essere tua.