# Engraginy: Simulació de Sistemes de Transmissió Mecànica

Biel Alavedra Busquet

January 17, 2026

**Resum–** Engraginy és un videjoc 3D que busca simular mecaniques de transmisió mecanica, fent ús d'engranatges, eixos, i altes mecanismes amb l'objectiu de construïr cadenes de producció i fabriques. El nucli del projecte és el sistema de potencia i transmisió mecànica, basat en grafs que fa ús de calcul vectorial i senyals per actualitzar el sistema quan és requereix. Ademés d'un sistema de transport de materials basat en simulacions físiques. Tots aquests sistemes disenyats i pensats per oferir una experiencia totalment reactiva.
**Paraules clau–** Godot, Videojoc, Grafs, GDScript, Logística, Simulació 3D.

**Abstract–** Engraginy is a 3D video game that seeks to simulate mechanical transmission mechanics, utilizing gears, shafts, and other mechanisms with the goal of building production chains and factories. The core of the project is the power and mechanical transmission system, based on graphs that use vector calculus and signals to update the system as required. Additionally, it features a material transport system based on physical simulations. All these systems are designed and intended to offer a fully reactive experience.
**Keywords–** Godot, Videogame , Graphs, GDScript, Logistics, 3D Simulation.

## 1 Introducció - Context del treball

El projecte Engraginy neix de la voluntat d'explorar mecàniques de joc més físiques dins del gènere de l'automatització. Mentre que títols referents com *Factorio* i *Satisfactory* utilitzen l'electricitat com un recurs binari o simplificat, aquest treball proposa una xarxa de potència mecànica on cada component (engranatges, eixos, cintes) afecta el rendiment de les cadenes de producció segons les lleis de la cinemàtica bàsica.

L'objectiu principal és la creació d'un sistema modular i extensible en el motor Godot 4 que permeti al jugador construir xarxes de potència complexes. El repte tecnològic resideix en la propagació dinàmica de la rotació en un entorn 3D, requerint l'ús de càlculs vectorials per determinar la viabilitat de les connexions, i les direccions de transmisió.

## 2 Motivació

La meva motivació per desenvolupar aquest projecte és principalment la d'aprendre com funciona el desenvolupament d'un videojoc. Els meus referents a l'hora de desenvolupar el projecte són videojocs com 'Factorio'[5], 'Satisfactory'[6], 'Dyson sphere program'[7], 'Shapez'[8], i 'Minecraft: Create Mod'[9] (una modificació del joc no oficial). D'aquests referents amb els quals tinc més experiència són DSP i Satisfactory, dos videojocs que m'han enganxat molt i m'han mostrat què entretingut i mentalment activador pot arribar a ser el gènere. El DSP és un joc molt més ambiciós, es comença en un petit satèl·lit dins d'un cúmul d'estrelles generat aleatòriament, i acaba quan el jugador construeix una esfera de Dyson i viatja entre sistemes solars. Tot això amb una perspectiva isomètrica que dona molt control i molta visió de tot allò que construeixes. En el Satisfactory l'objectiu és completar un ascensor espacial. Es comença en un planeta (el qual sempre és el mateix, el mapa és prefixat). Aquest joc és en primera persona, i totes les màquines que es construeixen són gegantines. Això impedeix una visió completa d'allò que construeixes. En quantitat de màquines les construccions són sempre més petites i molt més lentes de construir.

Dels dos jocs vull fer una cosa més similar a Satisfactory, ja que aquesta visió en primera persona dona un control més granular de les construcción a petit nivell, i encaixa molt millor amb les mecaniques que vull incorporar de Minecraft: Create mod, en la majoria de jocs de l'estil una part molt important és la generació d'energia, no serveix de res fer fàbriques més grans si no es tenen els recursos per mantenir-les, tant DSP, Factorio i Satisfactory utilitzen energia elèctrica, sigui solar, eòlica, carbó o nuclear. Tots aquests generadors s'acaben connectant a la xarxa elèctrica per donar energia a totes les màquines. En canvi, el Minecraft: Create mod requereix que tot sigui alimentat per energia cinètica, fent ús d'eixos, engranatges grans i petits, cadenes, corretges de transmissió... Com el meu objectiu és fer un joc amb una ambientació prerevolució

• E-mail de contacte: biel.alavedra@gmail.com
• Menció: Computació
• Treball tutoritzat per: Enric Marti Godia
• Curs 2025/26

industrial, i no del tipus fantàstic, crec que aquesta és la millor forma de fer-ho. També això afegeix totes les mecàniques d'haver de connectar les màquines, no només a una velocitat concreta, doncs algunes seran direccionals, com les cintes per transportar material.

D'entre tots els gèneres de videojocs perquè he escollit automatització? Considero que aquest és un gènere que segueix molt la filosofia d'un programador, o en general la de tots els enginyers: dividir i vèncer. L'inici d'aquests jocs és senzill, hi ha unes poques màquines i cal fer processos simples. Per exemple, amb un extractor de recursos automàtic i cal processar els recursos en brut per tal d'obtenir el recurs processat (de mena de ferro lingots a de ferro). Però això ràpidament canvia, agafant d'exemple el joc DSP. Si vols fer una placa de circuits es necessita un forn de fosa que converteixi el ferro en brut a plaques de ferro, un altre forn de fosa que converteixi coure en brut a plaques de coure i un assemblador que agafi aquests dos materials i els converteixi en la placa de circuits. Això és només un dels primers passos, on cada vegada es disposa de més i més materials diferents, processos diferents i els objectes requerits són cada vegada més complexes. Per força no es poden afrontar tots aquests problemes simultàniament, cal dividir els problemes en mòduls que es puguin replicar cada vegada que calgui per a resoldre aquell problema.

Dins de la comunitat de fans d'aquest gènere de videojoc hi ha molts enginyers i gent que li agrada molt optimitzar processos i fer construccions el màxim d'eficients possibles. Per tot això no només considero el gènere com a entretingut, sinó que també en certa manera és educatiu, doncs força al jugador tant sí com no a organitzar-se, pensar formes eficients de construir coses, com connectar les entrades i sortides de totes les fabriques, calcular què tanta entrada es necessita per la sortida objectiva i construir d'acord amb això. Per completar un joc d'aquests cal ser organitzat, metòdic i pensar en solucions segons el problema que es tingui.

## 3 ARQUITECTURA DEL SISTEMA

El programari s'ha estructurat seguint els principis de la programació orientada a objectes (POO) mitjançant el sistema de nodes de Godot. La classe base `Building.gd` defineix els comportaments d'interacció, mentre que `PowerNode.gd` hereta aquesta base per incloure la lògica de ports i connexions. Engranatges, eixos, generadors i maquines hereten d'aquest `PowerNode.gd` per tal de definir més concretament les propietats especifiques de cada cas. Ja que s'ha fet ús de Godot el projecte havia d'estar estructurat internament tal i com ho requereix el motor, és imperatiu fer us del sistema de nodes i escenes per crear tots els objectes i parts del videojoc. Això ens deixa mab dues grans parts, l'estructura del programari, basada en POO i l'estructura interna de tots els objectes i escenes que són un conjunt de nodes predeterminats i nodes amb una implementació propia.

### 3.1 Nodes i escenes

Abans de parlar de la meva implementació tècnica és important explicar com funciona el sistema de nodes i escenes de Godot. Cada element del joc és un node amb unes propietats i funcions diferents. Hi ha tres famílies de nodes principals, nodes 3D, nodes 2D i nodes de GUI. Dins de la família de nodes 3D tenim exemples com:

- Camera3D
- StaticBody3D
- MeshInstance3D
- CollisionShape3D
- RayCast3D
- etc.

Cada un d'aquests nodes és una classe diferent amb les seves propietats, però totes hereten de la classe Node3D. Els nodes de GUI ens permeten crear interfícies d'usuari on utilitzarem l'estructura jeràrquica per encapsular cada element d'interfície com ara:

- Container
- Label
- TextureRect
- Button
- Panel
- ItemList
- etc.

Les escenes seran definides per l'usuari i són una col·lecció de nodes de qualsevol tipus, una escena pot ser el jugador, la interfície d'usuari o un nivell sencer d'un joc. Depèn de cada desenvolupador decidir com organitzar. Aquestes escenes es poden afegir a altres escenes per poder així evitar treballar amb grans problemes a la vegada, i tenir-ho tot en mòduls més petits.

En l'escena que conté el jugador, Fig. 1, tenim com a node pare un PlayerCharacter (node propi, hereta de la clase base CharacterBody3D), amb 5 fills, el punt d'anclatge de la camera (CameraHolder), la caixa de col·lisions del personatge (Hitbox), els RayCast3D (Raycasts), el Model i la màquina d'estats encarregada del control del personatge. Aquesta màquina d'estats està iniciada amb nodes buits, aquests nodes són la classe base per a tots els objectes de Godot, i no tenen propietats especials. Podem veure que en el lateral tenim una icona amb un pergamí, això ens indica que hi ha un script vinculat al node, si està en gris vol dir que aquest script és heretat, compartit amb altres nodes de la mateixa classe. Si és blanc ens indica que el node té un script únic.

## 4 LÒGICA DE TRANSMISSIÓ MECÀNICA

La transmissió de moviment entre components és la part més complexa del sistema. Aquesta es defineix mitjançant la funció `calculate_speed` a `PowerNode.gd`.

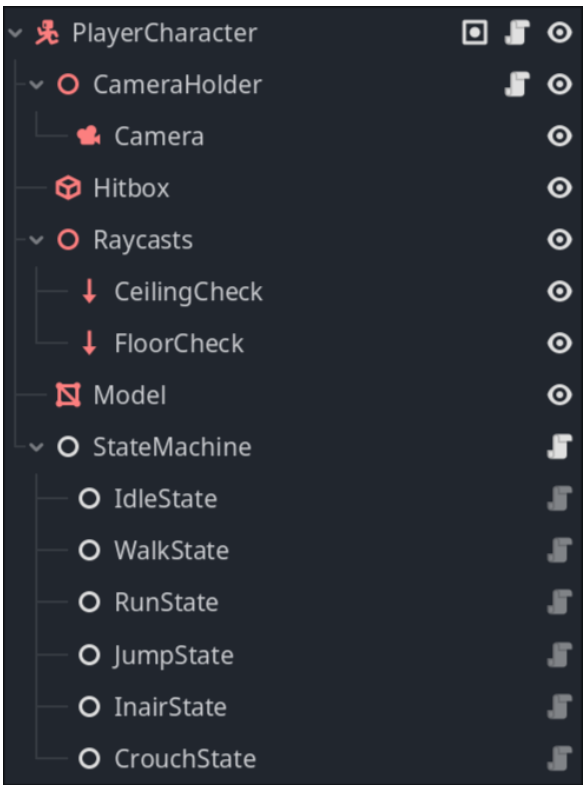


Fig. 1: Escena del jugador

### 4.1 Càlcul Vectorial de Rotació

El sistema utilitza l'orientació dels eixos de gir en l'espai 3D per determinar com es transmet la velocitat:

1. **Alineació Directa:** Si els eixos estan alineats ($|\vec{a} \cdot \vec{b}| > 0.9$), la velocitat es transmet multiplicant-la pel signe de l'alineació.

2. **Canvi de Direcció per Engranatges:** Si la connexió és entre engranatges (*cogs*), el sentit de gir s'inverteix automàticament per simular el contacte físic entre dents.

3. **Connexions Perpendiculars:** S'utilitza el producte vectorial entre l'eix de gir i el vector de distància per calcular la tangent de gir i assegurar que la rotació sigui coherent amb la posició relativa de les màquines.

### 4.2 Ràtios de Velocitat

Cada port de connexió pot aplicar un `ratio_multiplier`. Això permet la creació de reductores i multiplicadores on un engranatge gran (`cog_big.tscn`) pot fer girar un de petit al doble de velocitat, simulant així el comportament real de les transmissions mecàniques.

## 5 Logística i Transport d'Ítems

El sistema de transport s'ha implementat a `belt.gd` utilitzant els nodes `Path3D` de Godot. Aquestes cintes transportadores actuen simultàniament com a consumidors de potència i com a conductes logístics.

### 5.1 Gestió de Cues i Bloquejos

Per oferir un comportament realista, els ítems a la cinta no només es mouen a la velocitat de la xarxa, sinó que interactuen entre ells. Mitjançant `overlapping_areas`, cada ítem detecta si el camí està obstruït per un altre recurs. Si hi ha una obstrucció i el moviment és en la direcció del bloqueig, l'ítem s'atura de forma independent, permetent l'acumulació de recursos (*backstuffing*) típica dels jocs d'automatització.

### 5.2 Transferència entre Segments

La connectivitat entre cintes es gestiona mitjançant ports de proximitat (`FrontPort` i `BackPort`). Quan un ítem arriba al final del seu recorregut paramètric (`progress_ratio` igual a 0 o 1), el sistema de cerca de connexions intenta transferir-lo a la següent cinta disponible en la cadena logística.

## 6 Implementació Visual i Feedback

El feedback visual és crucial perquè el jugador entengui l'estat de la seva fàbrica.

- **Shaders dinàmics:** S'ha desenvolupat un shader per a les cintes (`belt.gdshader`) que anima la textura segons la velocitat de gir calculada pel `PowerGridManager`.
- **Rotació Física:** Els components visuals dels engranatges i eixos giren mitjançant codi en el mètode `_process`, ajustant la seva freqüència de gir a la velocitat cinemàtica real.

## 7 Conclusions

La implementació del sistema de potència mecànica d'Engraginy demostra que és possible crear simulacions industrials complexes i eficients en motors de propòsit general com Godot 4. L'ús de grafs per a la xarxa d'energia permet mantenir un rendiment estable fins i tot amb centenars de nodes actius, ja que els càlculs pesats només es realitzen quan la xarxa canvia estructuralment.

## Agraïments

Vull agrair al meu tutor el seu suport i guiatge durant el desenvolupament d'aquest TFG, així com als desenvolupadors de la comunitat de Godot pels recursos i eines *open-source* que han fet possible aquest projecte.

## References

[1]

## Apèndix

### A.1 Algorisme de Resolució de Velocitats

El següent bloc de codi (simplificat) mostra com el `PowerGridManager` propaga la velocitat en la xarxa

utilitzant una cua BFS per garantir que cada node es calcula en l'ordre correcte de dependències:

- S'inicialitza una cua amb els generadors actius.
- Per a cada node, es visiten els seus veïns a través de `PortConnection`.
- Es crida a `calculate_speed()` per determinar la nova velocitat segons la ràtio del port receptor.