Universitat Autònoma de Barcelona

Informe progrés I

Engraginy

Biel Alavedra

Tutor: Enric Marti Godia

Grau en Enginyeria Informàtica

Escola d'Enginyeria

Curs 2025-2026

Control del document

Informació del document

	Informació
Identificació del document	Informe Progrés I
Autor del document	Biel Alavedra Busquet
Data de creació	11/11/2025
Data del document	11/11/2025
Nom del fitxer	Informe progres I.odt

Historial del document

Versió	**Data versió**	**Canvis**
1.0	11/11/2025	Creació document, canvis en apartats anteriors per reflectir millor els nous objectius
1.1	12/11/2025	Començar a escriure l'apartat d'implementació
1.2	13/11/2025	Continuar amb l'apartat d'implementació
1.3	14/11/2025	Acabar informe
1.4	20/11/2025	Aplicar correccions

Taula de continguts

1. Resum

Aquest projecte té com a objectiu el desenvolupament d'un videojoc del gènere de l'automatització. Aquest gènere es va començar a popularitzar amb la sortida de Factorio, però no va ser el primer del gènere, aquest joc, Factorio, està inspirat en el mod de Minecraft, IndustrialCraft.

En el projecte desenvoluparé un videojoc d'aquest gènere utilitzant el motor de codi obert Godot[1], vull fer un videojoc en primera persona, amb una ambientació prerevolució industrial i sense elements fantàstics, l'objectiu és implementar 4 nivells tecnològics, començar utilitzant fusta i pedra, coure, ferro i acer com a últim. Com a llenguatge de programació utilitzaré GDscript perquè està fortament integrat amb el motor i és més senzill, l'objectiu final serà passar les parts del codi més crítiques i que necessitin una millora de rendiment a C++.

2. Paraules clau

Videojoc, desenvolupament, gènere d'automatització

3. Motivació

La 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).

Abans d'entrar en cap altre tema vull definir una mica que és per mi aquest gènere i que el fa especial. En els jocs d'automatització jo considero que hi ha dos grans apartats, les fàbriques, encarregades de produir tots els materials necessaris per progressar en el joc, i les centrals d'energia, dedicades a subministrar tota l'energia necessària per al funcionament de les fàbriques. No té cap sentit ampliar la fàbrica si no tens energia per suplir-la, i no té sentit dedicar-te només a millorar el subministrament d'energia, primerament perquè a mesura que avances en el joc desbloqueges millors mètodes de generació, i segon perquè perds temps i recursos en ampliar una cosa que no t'ajuda a progressar. Cada joc fa una implementació diferent dels dos sistemes, a excepció del Shapez[8], el qual no té un sistema d'energia.

L'inici de tots aquests jocs és similar, es comença processant uns recursos bàsics, per fer un producte més complex, aquest producte complex l'utilitzes per poder construir noves

màquines i tecnologies que a la vegada et serviran per fer un producte encara més complex. I així es va repetint el cicle fins que el joc acaba. És una idea senzilla i no és un cicle de joc massa complex, la complexitat de tots aquests jocs ve de poder balancejar totes les entrades de recursos i escalar correctament totes les parts de les fàbriques, això mentre fas un disseny no massa rebuscat perquè si es trenca alguna cosa ho hauràs d'arreglar.

D'aquests referents amb els quals tinc més experiència són el DSP[7] i el Satisfactory[6], es tracta dos videojocs que m'han enganxat molt i m'han mostrat que tan entretingut i mentalment activador pot arribar a ser el gènere. El DSP[7] és un joc molt més ambiciós, comences en un petit satèl·lit dins d'un cúmul d'estrelles generat aleatòriament, i acabes fent una esfera de Dyson i viatjant entre sistemes solars, tot això amb una perspectiva isomètrica que et dona molt control i molta visió de tot allò que construeixes. En el Satisfactory[6] 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 gegantesques, això impedeix una visió tan 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 al Satisfactory, amb mecàniques del mod Minecraft: Create[9], 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 tens els recursos per mantenir-les, tant DSP[7], Factorio[5] i Satisfactory[6] 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[9] 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 industrial, i no vull fer un joc fantàstic, aquesta és la millor forma de fer-ho, crec jo, també això afegeix totes les mecàniques d'haver de connectar les màquines no només a una velocitat concreta, també 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 has de processar els recursos en brut per tal d'obtenir el recurs processat (mena de ferro → lingots de ferro). Però això ràpidament canvia, agafant d'exemple el joc DSP[7], si vols fer una placa de circuits necessita un forn de fosa que converteixi el ferro en brut a plaques de ferro, un 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 complexos. Per força no es poden afrontar tots aquests problemes simultàniament, has de dividir els problemes en mòduls que puguis replicar cada vegada que calgui resoldre aquell problema.

Dins de la comunitat de fans d'aquest gènere de videojocs 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, et veus forçat tant sí com no a organitzar-te, pensar formés eficients de construir coses, com connectes les entrades i sortides de totes les fabriques, calcular que tanta entrada necessites per a la sortida i construir d'acord amb això. Per completar un joc d'aquests cal ser organitzat, metòdic i pensar en solucions segons el problema que tinguis,

4. Objectius del projecte.

Aquest projecte té com a objectiu el desenvolupament d'un videojoc en primera persona del genere d'automatització, amb una progressió tecnològica de 4 nivells i un sistema d'energia basat en la rotació. Com a eines principals de desenvolupament s'utilitzarà Godot com a motor i Blender per a la creació de tots els models

Llistat d'objectius del projecte:

- Primera persona
- Un sol mapa, sense generació automàtica
- Quatre nivells de tecnologia: pedra, coure, ferro i acer
- Sistema d'energia basat en rotació

Eines de desenvolupament:

- Godot[1] com a motor del videojoc.
- C++[2] for Godot[3] per a les parts del codi que necessitin un millor rendiment.
- Visual Studio Code[4] com a editor de text
- Blender[11] per a crear tots els models 3D.

5. Planificació

Aquesta taula s'utilitzarà de guia per fer un seguiment del projecte i seu el grau de maduresa d'aquest. Aquest grau es calcula com un promig ponderat.

Tasca	Descripció	Durada (setmanes)	Grau de finitzalització
Estat de l'art	Buscar productes similars, caracteritzar-los i comparar-los	1	100%
GDD	Creació del GDD	3	80%
Implementació	Implementació i creació del videojoc a partir del GDD	10	20%
Test	Test intern i extern	2	0%
Memòria i Presentació	Escriure la memòria i la presentació oral	2	0%
GRAU DE MADURESA			30%

Taula 1. Planificació de tasques del projecte

5.1.Estat de l'art

Aquest apartat està completat en el document Informe Inicial1.5.pdf

5.2.Game design document

En el document adjunt com a Annex 1 s'inclou el Game Design Document (GDD), aquest document inclou tota la informació de com es farà la implementació del projecte. Tenir clar com fer un videojoc des del principi no és tasca senzilla, per tant, aquest document s'anirà actualitzant amb el temps segons els canvis que faci, sigui per dificultats tècniques, o perquè en el moment d'implementar-ho he vist que no encaixa amb la visió inicial.

Aquest document té com a objectiu ajudar-me a seguir una guia, per mantenir la coherència del projecte i no perdre de vista els objectius del joc, no pretén ser un document formal, sinó una eina de treball per al desenvolupament.

5.3. Implementació

En aquesta secció explicaré com s'estan implementant les diferents mecàniques del joc i quines dificultats i solucions estic trobant. Parlaré de quines mecàniques s'implementen, com s'implementen tècnicament i quin és el futur del projecte.

5.3.1. Implementació de les mecàniques

Originalment, s'havia pensat aquest projecte en tercera persona i amb càmera isomètrica, però això no encaixa bé amb els objectius del joc, ja que si tot és vist en tercera persona no hi ha tanta percepció de la profunditat dels edificis, i si el joc funciona amb aquest sistema d'eixos i engranatges és molt important veure on està connectada cada cosa o amb quin sentit gira cada engranatge i eix. Per descriure millor el meu objectiu actual, vull fer un videojoc de construcció en primera persona, que permeti fer ús de la verticalitat i desplegar grans sistemes de producció, tot utilitzant un sistema d'energia rotacional per forçar al jugador a pensar en cadenes d'engranatges i relacions de velocitat. És a dir un Satisfactory amb la mecànica de sistemes de rotació de Minecraft: Create mod.

5.3.2. Implementació tècnica

Abans de parlar de la meva implementació tècnica és important explicar com funciona Godot i quines són les eines més importants.

a) Sistema de nodes i escenes:

Godot funciona amb un sistema de nodes i escenes. Els nodes ho són tot, cada objecte que afegim a l'escena, cada interfície, partícula, font de llum o font d'àudio. 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:

- *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 i poder afegir text i botons allà on necessitem. Les escenes seran definides per l'usuari i són una col·lecció de nodes, una escena pot ser el jugador, la interfície d'usuari o un nivell sencer d'un joc, depèn de cada desenvolupador decidir com són organitzades, 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.

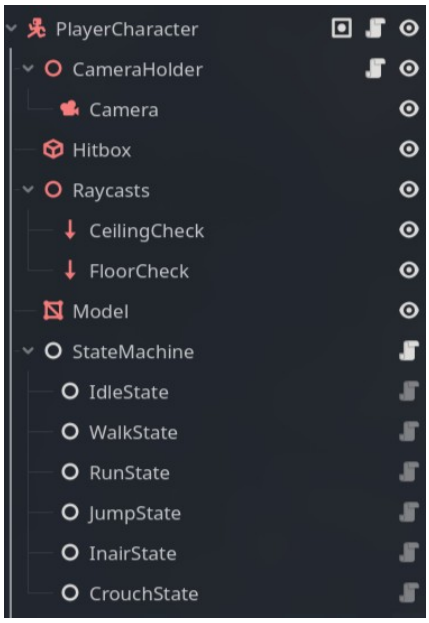

Figura 6. Escena del jugador

En l'escena que conté el jugador, vista en la figura 6, tenim com a node pare un *CharacterBody3D*, amb 5 fills, la càmera, la caixa de col·lisions del personatge, els *RayCast3D*, 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 a aquell node, si està en gris vol dir que aquest script és heretat, si és blanc ens indica que aquell node té un script únic.

En l'escena d'un objecte construïble (figura 7), en aquest cas un generador que s'utilitza com a test i que no estarà inclòs en el producte final, tenim un *Node3D* com a pare de la classe *Generator*, com a fills tenim dues *MeshInstance3D*, per al cos i per a l'eix, dos *StaticBody3D* amb les seves respectives col·lisions per interactuar amb el jugador, un *RayCast3D* que s'utilitza per comprovar si connectat a l'eix hi ha un objecte vàlid per a ser

mogut, i 4 RayCast3D que s'utilitzen per comprovar que en el moment de construir l'edifici les 4 cantonades interactuen amb el terra.

Figura 7. Escena d'un edifici construïble

b) Sistema de senyals:

Godot té un sistema per poder connectar diferents nodes i escenes sense necessitat de saber exactament l'estructura de l'escena ni haver de referenciar el node directament. Aquests senyals són missatges que els nodes poden emetre, els altres nodes es poden connectar a aquest senyal i executar una funció quan reben el missatge. Aquests missatges es poden emetre directament per codi o poden respondre a inputs del jugador o a canvis en l'estructura de l'escena. Aquest sistema redueix molt les dependències i ho fan tot més modular, ja que evites referències directes entre escenes i nodes. Per explicar com utilitzo aquest sistema posaré com a exemple el sistema que he desenvolupat per interactuar amb objectes.

En aquest fragment de codi (figura8) es defineixen dues funcions utilitzades per emetre els tres senyals utilitzats per interactuar amb l'entorn, *unfocused*, *focused* i *interacted*, la funció *interact_cast()* és executada cada fotograma del joc per comprovar si estem interactuant o no amb objectes, el primer que fem es obtindre l'objecte que el jugador està mirant directament, emetem un *RayCast3D* de l'origen de la càmera a una certa distància i retornem el primer objecte amb el qual interactua, això ho fa la funció *make_cast_query()*.

Si l'objecte retornat és el mateix que en el fotograma anterior llavors no fem res, en cas contrari mirem si en l'objecte anterior hem definit el senyal *unfocused*, si no estigués definida vol dir que amb aquest objecte no es pot interactuar, i llavors emetem el senyal, si el nou objecte que estem mirant té el senyal *focused* llavors llancem aquest senyal.

```gdscript
func interact_cast() -> void:
    var current_cast_result : StaticBody3D = make_cast_query()
    if current_cast_result != interact_cast_result:
        if interact_cast_result and interact_cast_result.has_user_signal("unfocused"):
            interact_cast_result.emit_signal("unfocused")
        interact_cast_result = current_cast_result
        if interact_cast_result and interact_cast_result.has_user_signal("focused"):
            interact_cast_result.emit_signal("focused")
    pass

func interact() -> void:
    if interact_cast_result and interact_cast_result.has_user_signal("interacted"):
        interact_cast_result.emit_signal("interacted")

    pass
```

Figura 8. Fragment de codi de l'objecte PlayerCharacter que conté les funcions per llençar senyals

La funció *interact()* només s'executa quan el jugador executa l'acció d'interactuar, i fem el mateix, comprovem que l'objecte tingui el senyal *interacted* i en cas afirmatiu la llancem.

En els objectes tenim un node base, anomenat *InteractionComponent* amb el script encarregat de processar aquests senyals i cridar a les funcions de l'objecte adients. Aquest node ha de ser fill d'un node *StaticBody3D*, ja que quan fem la comprovació amb objectes aquesta han de tenir un *StaticBody3D* per interactuar amb el *make_cast_query()*. Per tant, processem els senyals des del fill i cridem les funcions del pare.

```gdscript
func _ready() -> void:
    parent = get_parent()
    main_object = parent.get_parent()
    connect_parent()

func connect_parent() -> void:
    parent.add_user_signal("focused")
    parent.add_user_signal("unfocused")
    parent.add_user_signal("interacted")
    parent.connect("focused", Callable(self, "in_range"))
    parent.connect("unfocused", Callable(self, "not_in_range"))
    parent.connect("interacted", Callable(self, "on_interact"))
```

Figura 9. Fragment de codi de l'objecte InteractionComponent

En l'objecte *InteractionComponent* (figura 9) senzillament busquem el pare i connectem totes aquests senyals amb les funcions *in_range*, *not_in_range* i *on_interact*.

5.3.3. Eines de desenvolupament

El programa utilitzat principalment per a desenvolupar el joc és Godot, i tots els models estan fets utilitzant Blender[11]. Per mantenir un historial del projecte i un control de versions s'utilitza Git.

5.3.4. Dificultats i solucions

Només començar a desenvolupar aquest projecte ja m'he trobat amb moltes dificultats, programar un videojoc és molt diferent de qualsevol programació feta fins ara i a més mai havia utilitzat Godot en profunditat. Entendre com construir interfícies d'usuari, utilitzar l'editor correctament, aprendre els diferents nodes i els seus usos, etc.

Encara queden moltes coses a començar que estic segur que seran igualment difícils o més, però amb el sistema que tinc més dificultats actualment és amb la connexió d'eixos i engranatges entre si, amb generadors i poder connectar-ho als edificis corresponents. No només els eixos s'han de connectar i tenir la mateixa propietat de velocitat, sinó que els canvis s'han de propagar per a tot el sistema, si connecto un nou generador tot el sistema ho ha de saber.

No ho he pogut implementar encarà, però la meva idea actualment és representar cada xarxa d'eixos i engranatges com un graf, i utilitzar tècniques per recórrer el graf sencer a cada fotograma per propagar canvis i mantenir tot el sistema actualitzat. Hi haurà nodes sense cost (eixos i engranatges), nodes que sumaran (generadors) i nodes que restaran (edificis).

5.3.5. Estat actual i passos futurs

Actualment, el projecte està en una fase poc desenvolupada, són moltes eines completament noves que he d'utilitzar per dur a terme el projecte i no s'ha pogut avançar massa.

Acabat del tot fer no hi ha res, tot està en les fases inicials: el sistema de construcció, poder connectar eixos entre ells i a un generador, interactuar amb els objectes construïts i tenir el menú de cada construcció (ex. en el menú del constructor només han d'aparèixer les receptes de constructor que estan desbloquejades).

Sistemes que he de començar completament i no hi ha res fet són: El transport de materials utilitzant cintes i que aquests materials entrin als edificis corresponents, sistema d'engranatges, inventari, la base del jugador, i el sistema d'objectius.

6. Bibliografia i referències

[1] https://godotengine.org/ Pàgina oficial del motor Godot (últim accés setembre 2025)

[2] https://isocpp.org/ Pàgina sobre el llenguatge C++ (últim accés setembre 2025)

[3] https://docs.godotengine.org/en/4.4/tutorials/scripting/gdextension/gdextension_cpp_example.html Documentació oficial Godot per crear plugins utilitzant C++ (últim accés octubre 2025)

[4] https://code.visualstudio.com/ Pàgina oficial Visual Studio Code (últim accés s etembre 2025)

[5] https://www.factorio.com/ Pàgina oficial de Factorio (últim accés octubre 2025)

[6] https://www.satisfactorygame.com/ Pàgina oficial de Satisfactory(últim accés setembre 2025)

[7] https://store.steampowered.com/app/1366540/Dyson_Sphere_Program/ Pàgina d'Steam de Dyson sphere program (últim accés setembre 2025)

[8] https://github.com/tobspr-games/shapez.io Pàgina de Github de Shapez (últim accés setembre 2025)

[9] https://github.com/Creators-of-Create/Create Pàgina de Github de Create (últim accés setembre 2025)

[10] https://www.curseforge.com/minecraft/mc-mods/industrial-craft Pàgina oficial del mod (últim accés octubre 2025)

[11] https://www.blender.org/ Pàgina oficial de Blender (últim accés novembre 2025)

Annexos i GDD

- El Game design document (GDD) complet s'adjunta com a Annex 1 en un document separat, titulat «Engraginy_GDD.pdf».