UAB
Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Fraile Marin, Adrià. *Anàlisi de jugabilitat a videojocs randomitzats i desenvolupament d'ajuda*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Grau en Enginyeria de Dades)

This version is available at https://ddd.uab.cat/record/337350

under the terms of the CC BY-SA license.

Anàlisi de jugabilitat a videojocs randomitzats i desenvolupament d'ajuda

Adrià Fraile Marín

Resum— Aquest treball busca utilitzar processos de gestió i visualització de dades per desenvolupar una eina de suport per a jugadors de videojocs modificats per la comunitat, concretament del videojoc "Hollow Knight". El procés agafa una sèrie d'arxius interns produïts per l'arxiu de guardat o els reorganitza a un format unificat que permet tractar-los tots amb la mateixa eina, oferint després visualitzacions de la partida útils pel jugador, ja sigui per analitzar el seu recorregut o buscar el següent pas.

Paraules clau— Processament de dades, graf, visualització de dades, arxius de registre, modding, aleatoritzador.

Abstract— This project intends to use data management and visualization processes to develop a helping tool for players of videogames modified by the community, in this case "Hollow Knight". The process accesses internal files created by the safe file and reorganizes them in a single format that facilitates working on them by the program, generating a visualization of the game that can be used by the player, either to analyse their playthrough or just look for their next step.

Index Terms— Data processing, graph, data visualization, log files, modding, randomizer.

1 Introducció - Context del treball

Al món dels videojocs, un dels criteris principals que molts jugadors utilitzen al moment d'avaluar una compra és la quantitat de temps de vida que es pot treure del joc, ja que fa sentir que són diners més ben invertits: si pel mateix preu pots comprar un joc que dona per 20 hores o un que dona 50, és normal voler el de 50. Aquest fenomen arriba a un punt on la gent busca maneres d'estendre la vida útil del joc tot el possible.

La primera manera (i la més popular) és la dels reptes autoimposats. El concepte és fàcil: jugues al joc des de l'inici fins al final seguint una sèrie de normes que no formen part del joc com a tal, per exemple començar des de l'inici si mors una sola vegada o prohibir l'ús de certes eines clau, afegint més dificultat a la partida.

La segona és bastant més moderna: *modding*. La paraula *modding* prové de la paraula *modify* (modificar) en anglès, referint-se a modificacions fetes a un videojoc per part de la comunitat. Quant més temps passa, més popular es torna aquesta opció i més es facilita per part dels desenvolupadors, ja que permet a la comunitat estendre les possibilitats d'un videojoc més enllà de les capacitats inicials d'aquest.

Aquest treball explora una combinació de les dues opcions, el concepte dels *randomizers* (aleatoritzadors). La paraula prové de l'anglès *random* (aleatori), i consisteix en jugar un videojoc d'inici a fi on els seus continguts han estat aleatoritzats, o sigui, canviats de lloc entre ells de forma aleatòria. L'objectiu d'aquest repte és donar una experiència de joc il·limitada al jugador, ja que cada vegada que es vol fer-ne un la situació serà completament diferent de qualsevol altre. Aquest tipus de repte requereix un coneixement avançat del videojoc degut a que el jugador necessita saber com creuar zones sense els objectes necessaris, com explotar els que no hauria de tenir i saber les localitzacions dels objectes per saber on buscar les coses.

2 Estat de l'art - Objectius

Una vegada aleatoritzes una partida, es generen arxius de registre d'aquesta, on s'indica informació com: objectes aleatoritzats, configuració de l'aleatorització, noves localitzacions dels objectes, distribució del mapa i progrés del jugador.

L'objectiu és desenvolupar un programa capaç de llegir els arxius de registre i crear una visualització de la informació que sigui fàcil de llegir pel jugador, de manera que pugui utilitzar aquesta informació per analitzar coses que podria haver fet diferent o buscar com avançar a la partida actual. Aquesta informació serà representada en forma d'un graf de punts fàcilment editable (en cas de requerir canvis de generació del mapa), i per fer-ho és necessari primer gestionar totes les dades dels arxius de registre per obtenir la informació important.

3 CONTACTE AMB LA COMUNITAT

Aquest projecte requereix bastant investigació i dades que requeririen massa hores d'esforç individual. Llavors, el primer pas del treball és contactar amb la comunitat de jugadors de *Hollow Knight*, concretament les relacionades amb el *modding*. Després d'explorar diferents opcions, tres servidors de *Discord* (una aplicació de comunitats online on la gent pot crear servidors per reunir membres i parlar d'interessos comuns) han estat seleccionats:

- **The Blue lake[1]:** servidor de "BlueSR"[2], un creador de contingut a Youtube, el qual és un dels més grans dins del món d'aquest videojoc particular, creant contingut relacionat amb *speedruns* (acabar un joc en la menor quantitat de temps possible) i partides amb *mods*. Va ser bàsicament la meva primera toma de contacte, on em van dirigir a la resta de servidors per fer una investigació més profunda, a més de permetre'm fer preguntes a la comunitat relacionades amb l'experiència de jugador
- **Hollow Knight Modding[3]:** servidor on es troba tota la informació relacionada amb el desenvolupament, cerca, instal·lació i suport de *mods* de *Hollow Knight*. Aquí es on vaig poder entrar en contacte amb el creador del *mod* d'aleatorització, el qual em va facilitar accés a la pàgina de GitHub del sistema d'aleatorització[4], a més d'oferir-se a respondre preguntes sobre el funcionament del programa.
- **HK Speedrun and Racing Community[5]:** Veterans i mestres de jugar a aquest videojoc, fent competicions amb tot tipus de configuracions contínuament. Gràcies a aquesta comunitat, obtenir les dades per provar correctament les visualitzacions ha estat difícil.

4 FUNCIONAMENT DE L'ALEATORITZADOR

L'estructura dels arxius de registre està feta en base al funcionament de l'aleatoritzador, per tant és important entendre com funciona en general per poder analitzar correctament aquests arxius. Primer, totes les instàncies del joc estan representades com a *items* que pertanyen a categories relacionades amb el que són, per exemple: transició, història/text, objecte clau, diners, etc...

Segon, en base a la configuració de l'aleatoritzador, aquest separa els *items* en "aleatoritzar" i "no aleatoritzar", seguit de utilitzar les categories internes dels *items* per barrejar-los entre categories o només dins d'elles mateixes. Les configuracions més comunes d'aquest pas són:

- ***Standard*:** només els objectes "normals" estan barrejats entre ells. El repte d'aquesta configuració és conèixer el mapa per saber quins llocs tenen la major quantitat d'objectes a recol·lectar i a on pots accedir amb quines habilitats.
- ***Room Rando*:** les transicions també estan barrejades, però només entre elles mateixes, al mateix temps es divideix en configuracions aparellades (Si A -> B, llavors. B -> A) o caòtiques (A -> B no significa que B -> A). Aquesta configuració demana bona memòria i capacitat d'orientació, ja que estàs explorant un mapa completament nou, a més d'haver-te d'adaptar a les habitacions que et van sortir.
- ***TRTR / True Random*:** totes els objectes estan barrejats entre ells, sense importar la seva categoria interna, això significa que potser reculls un objecte del terra i ets transportat a una habitació ja que ha resultat ser una transició. Aquesta categoria busca desafiar-te amb complert caos, ja que mai sabràs què t'espera, també té un sistema d'arxius de registre completament diferent degut al seu comportament

És important destacar que no tots els objectes tenen la mateixa prioritat al moment de ser barrejats: els objectes necessaris per a la progressió y finalització del joc estan duplicats per més facilitat per trobar-los, a més d'utilitzar un sistema de lògica encadenada per assegurar-se de que l'arxiu aleatoritzat es pot passar sense fer trampes

5 ARXIUS DE REGISTRE

Aquest projecte tracta amb dues configuracions diferents degut als seus objectius: *Room Rando* i *TRTR*. Degut a que tenen diferents estructures d'arxius de registre, primer s'han analitzar les dues per trobar els arxius importants i quina informació es pot extreure d'aquests.

5.1 Room Rando

Degut a la separació natural que fa l'aleatoritzador dels *items* normals i les transicions, fa una separació clara entre els dos dins dels seus registres ho aprofiten per mostrar la informació dividida.

5.1.1 RawSpoiler.json

Com es pot entendre del nom, és la informació crua de tot l'aleatoritzador: quin objecte canvia amb quin i la ubicació exacta del mapa, configuració, costos d'objectes a els tendes, etc...

Ja que tenim la resta d'arxius, en aquest cas RawSpoiler.json només serveix per poder fer el graf del mapa sense modificar del joc, el qual es pot utilitzar com a referència.

5.1.2 TransitionSpoilerLog.json

L'arxiu més important per a la visualització, conté la informació de les connexions entre les habitacions de la partida, d'aquí s'extreu tot el necessari per fer el graf del mapa aleatoritzat, fent de base per la resta de processos.

5.1.3 TrackerData.json

Conté tota la informació relacionada al progrés del jugador: objectes agafats, transicions recorregudes, objectes claus trobats, etc... Tot en ordre cronològic. El més important és la informació de les transicions visitades, ja que permet observar l'evolució de l'exploració del mapa i el seu resultat final.

5.2 TRTR / True Random

Degut al tractament equitatiu de tots els *items* de la partida, la quantitat d'arxius és més petita, però tenen informació molt més densa.

5.2.1 _TrackerLogItems.json

Conté la informació de tots els objectes: identificador, nom, categoria, localització habitual, localització a la partida, entre altres coses. La informació de les transicions del mapa estan aquí, encara que una mica amagades.

5.2.2 TrackerLog.txt

Similar a l'arxiu "TrackerData.json" però incloent encara més informació que no és necessària. Degut a la caòtica naturalesa de la distribució dels *items* només es pot treure la informació del progrés del jugador del mapa d'aquí.

5.2.3 CorpseDreamRawData.json

Resulta que aquesta partida utilitza un altre *mod* que afegeix nous *items* a la partida. La informació necessària d'aquest items no està correctament indicada al Tracker i, per tant, s'ha d'aconseguir l'arxiu d'aquesta informació de la pàgina del *mod*[6].

6 TRACTAMENT DE DADES DE REGISTRE

Ara amb la informació localitzada, és necessari tractar-la per estandarditzar-la i utilitzar-la correctament. Diferents programes han estat desenvolupats per cadascuna de les transformacions.

6.1 Room Rando

Els formats d'aquesta configuració serà utilitzada com a referències per les transformacions, fent que no sigui necessari quasi cap programa

6.1.1 Mapa_vanilla.py

Extreu de l'arxiu "RawSpoiler.json" la informació de les localitzacions originals de totes les transicions, posant-les en el format que serà referit com a "TransitionLog":

```
[ {
        "source": sala1[porta entrada],
        "target": sala2[porta sortida]
} ]
```

El resultat és guardat a "VanillaSpoiler.json", el qual serà utilitzat posteriorment per crear una visualització del mapa per *default* del joc, permetent fer comparacions.

6.1.2 Lector_player_path.py

Agafa un arxiu en format "TrackerLog":

```
{
visitedTransitions: {
        "sala1[porta entrada]" : "sala2[porta sortida]"
        }
}
```

I el transforma a un json en format TransitionLog, per posterior representació com a mapa

6.2 TRTR / True Random

Els arxius al ser més densos requereixen més tractament, tant per obtenir la informació volguda com per posar-la en els formats mencionats anteriorment

6.2.1 CorpseDream_Filter.py

Extrau la localització dels "corpses" de CorpseDream-RawData.json per crear un diccionari "Filtered_CorpseDream_Data.json" on la clau és l'identificador del "corpse" i el valor és la sala on es trova habitualment.

6.2.2 OrganizadorTRTR.py

Localitza les transicions dins de l'arxiu "_TrackerLogItems.json" i, amb ajuda del diccionari "Filtered_CorpseDream_Data.json", escriu dos arxius:

- TRTR_Objects.json: recol·lecta la informació de les localitzacions dels objectes relacionats amb alguna transició i guarda la seva posició. Serà combinat amb "Filtered_CorpseDream_Data.json" per crear "TRTR_Objects_Complete.json" per a futur ús.

- TRTR_Map.json: Aprofita la recol·lecció per recopilar les connexions de transicions i guardar-les en format "TransitionLog".

6.2.3 Filtro_player_path_TRTR.py

El problema amb l'arxiu de registre "TrackerLog.txt" és que cataloga totes les obtencions de *items* amb els seus noms identificadors, sense localitzador. Fins ara era fàcil saber a quina sala pertany cada transició ja que sempre estaven a portes, les quals tenen la sala de la seva localització al nom (sala1[porta]), però aquest no és el cas per *items* més habituals.

El diccionari "TRTR_Objects_Complete.json" és d'on s'extreuen les ubicacions dels objectes per fer el mapa en format "TrackerLog", omplint els vuits on falta informació, finalment sent passat pel "Lector_player_path.py" per canviar novament el format a "TransitionLog".

7 PREPARACIÓ DE LA VISUALITZACIÓ

Per a fer una visualització adequada de la informació és necessari escollir les eines correctes

7.1 Llibreries de Python

- ***Networkx*:** Centrada en la creació i gestió de grafs de punts. Utilitzada per la facilitar crear els mapes i calcular models de forces per facilitar la visualització d'aquests.
- ***Pyvis.network*:** Una altra eina per grafs de punts, però més difícil d'utilitzar en comparació a *Networkx*. Utilitzada per la creació de les visualitzacions en format html

7.2 Obtenció d'imatges de les sales

Totes les habitacions de l'extens mapa de *Hollow Knight* són fàcilment accessibles, però amb el problema de que estan en blanc i negre. Per poder tenir una visualització més clara de les distincions de les zones a les que pertanyen cada sala (el seu color), ha estat necessari fer captures de pantalles de cadascuna de les sales del joc mitjançant una web desenvolupada per la comunitat[7].

7.3 Versions alternatives de les imatges

Hi han dos arxius dits "transparentes.py" i "border.py" encarregats de fer dues versions extra de cada imatge: una amb vores blanques i una altra amb la opacitat al 50%. Aquestes versions alternatives seran utilitzades per representar zones on ha passat i no ha passat el jugador més endavant.

7.4 Guardar el resultat

Per evitar haver d'executar el codi cada vegada que es vol visualitzar el graf, és important guardar el resultat en un arxiu que es pugui obrir i permeti complerta observació en qualsevol moment. Després de considerar diverses opcions, he arribat a la conclusió que un arxiu html que obrir al navegador seria la millor opció.

8 FUNCIONAMENT GENERAL DE LA VISUALITZACIÓ

Buscant complir diferents possibles necessitats de l'usuari, hi han 4 codis diferents, els quals segueixen un procés similar

8.1 Requisits

Cada programa té requisits de funcionament lleugerament diferents, però la llista total és:

- Llibreries de Python: json, networkx, matplotlib.pyplot i pyvis.
- Arxius en format TransitionLog: mapa de la partida i recorregut del jugador.
- Carpeta amb les imatges de les sales i les seves versions modificades.

8.2 Grafo_mapa.py

- **Objectiu:** creació del mapa aleatoritzat de la partida.
- **Input:** arxiu json en format TransitionLog de les connexions del mapa, imatges normals de les sales.
- **Output:** arxiu html.

Llegeix el json amb les dades del mapa i les implementa en un graf G mitjançant networkx, on els nodes són les habitacions i les arestes els camins entre aquestes, tenint com a títol la descripció del camí exacta que representa (exemple: "Cliffs_03[right1] - Fungus1_07[left1]").

Networkx aplica un model de forces amb la funció "forceatlas2_layout()" amb una *seed* definida per a resultats consistents on es pot veure sense problema el mapa.

L'usuari pot utilitzar aquest html com un mapa per guiarse durant la partida, ja que és molt fàcil oblidar-se de certes coses i no tot el món vol apuntar tots els descobriments que fa.

8.3 Grafo_descubrimiento.py

- **Objectiu:** visualitzar i comparar el mapa aleatoritzat de la partida amb tot l'explorat pel jugador.
- **Input:** arxiu json en format TransitionLog de les connexions del mapa, arxiu en format TransitionLog del recorregut del mapa del jugador (no necessàriament en ordre correcte), imatges de les sales amb bora blanca y imatges de les sales transparents.
- **Output:** arxiu html.

Segueix el mateix procés per crear el graf G i el model de forces, però també utilitza el json de l'exploració del jugador per crear un graf Gp.

Al moment de representar les parts de G que coincideixen amb Gp són ressaltades (arestes vermelles i imatges amb vores blanques), mentre que la resta són semi transparents per un major contrast.

Una vegada complerta una partida, aquest html pot donar informació de què tan "òptima" ha estat aquesta, observant les parts del mapa que han evitat recórrer.

8.4 Grafo_player_timeline.py

- **Objectiu:** visualització del recorregut del jugador del mapa en forma de línia de temps.
- **Input:** arxiu json en format TransitionLog de les connexions del mapa, arxiu en format TransitionLog del recorregut del mapa del jugador, imatges de les sales amb bora blanca y imatges de les sales transparents.
- **Output:** arxiu html.

Similar a "Grafo_descubrimiento", utilitza G y Gp per fer una comparació de l'exploració del jugador amb el mapa complert, amb la diferència de que el recorregut del jugador és mostrat pas a pas amb una línia de temps,

és una eina que busca facilitar a un jugador que ha acabat una partida poder analitzar el recorregut d'aquesta, observant possibles rutes alternatives que potser haurien ajudat a l'evolució de l'exploració.

8.5 Grafo_siguiente_paso.py

- **Objectiu:** visualització limitada del mapa respecte.
- **Input:** arxiu json en format TransitionLog de les connexions del mapa, arxiu en format TransitionLog del recorregut del mapa del jugador, imatges de les sales amb bora blanca y imatges de les sales transparents.
- **Output:** arxiu html.

És bàsicament el mateix procés que "Grafo_descubrimiento", amb la diferència de que oculta tots els nodes i arestes que no són accessibles des de les seccions del mapa explorades del jugador. Bàsicament, és una eina per al jugador per consultar on poden anar sense veure tot el mapa.

9 POSSIBLES MILLORES

A causa de la sortida de la seqüela de *Hollow Knight* (*Hollow Knight: Silksong)* el meu accés a la informació tècnica més detallada de l'aleatoritzador ha estat limitada, ja que els creadors dels *mods* estan ocupats creant les versions pel nou joc, i no he pogut entrevistar tot el que hagués volgut. Junt amb la falta de temps, hi van haver diverses característiques que no he pogut implementar, que podrien ser implementades en un futur:

9.1 Visualització Camí Òptim

Desconec el funcionament exacte de l'aleatoritzador al moment de calcular el camí fins el final al generar la partida modificada (per assegurar que la partida és passable), i utilitzar algoritmes de grafs convencionals no seria correcte ja que certs factors no podrien ser tinguts en compte.

9.2 implementació lògica de l'aleatoritzador

Al generar la distribució del mapa, l'aleatoritzador utilitza una lògica assignada a cada transició del mapa, indicant quins són els requisits exactes per poder accedir a aquesta transició. Aquesta lògica és bastant complexa i no la vaig poder trobar als arxius, per tant no han estat implementades.

L'ús d'aquesta lògica permetria una millor representació dels recorreguts pels mapes i quines transicions són realment accessibles a cada situació.

9.3 Visualització Items Clau

Per a ser una bona eina de visualització per el públic més avançat d'aquest entorn, seria necessari crear una forma de poder observar quins items clau estan a quina sala, per entendre per què el jugador ha fet certes decisions

9.4 Generació conjunta de grafs

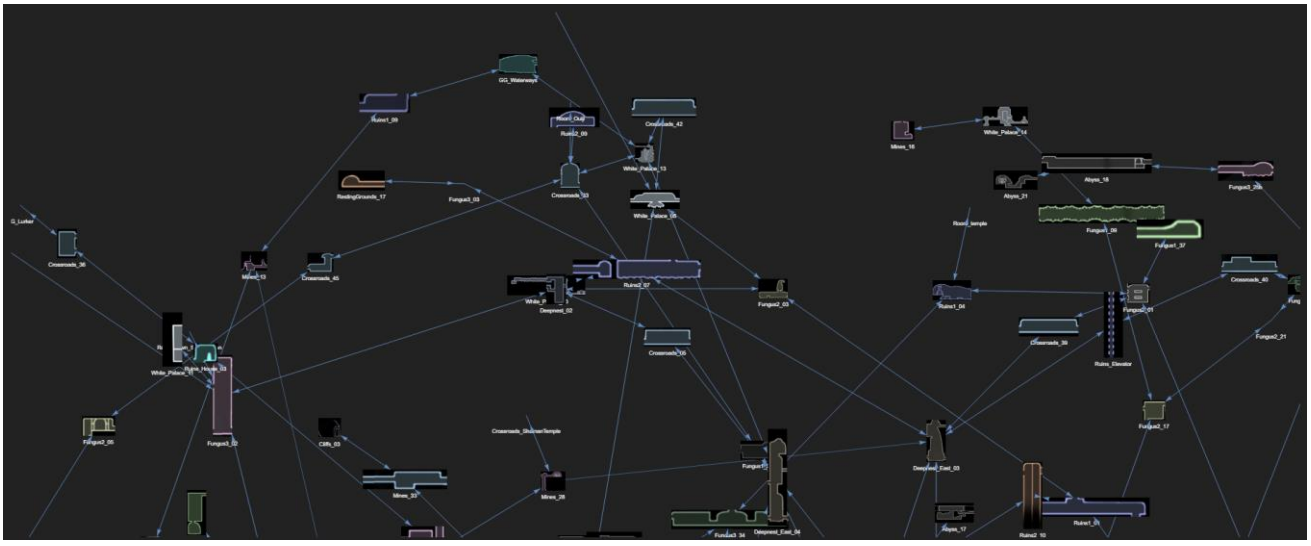
Es podria fer un script capaç de fer tots els processos i posar-los a un sol html, però per simplicitat he escollit no fer-ho.

Bibliografia

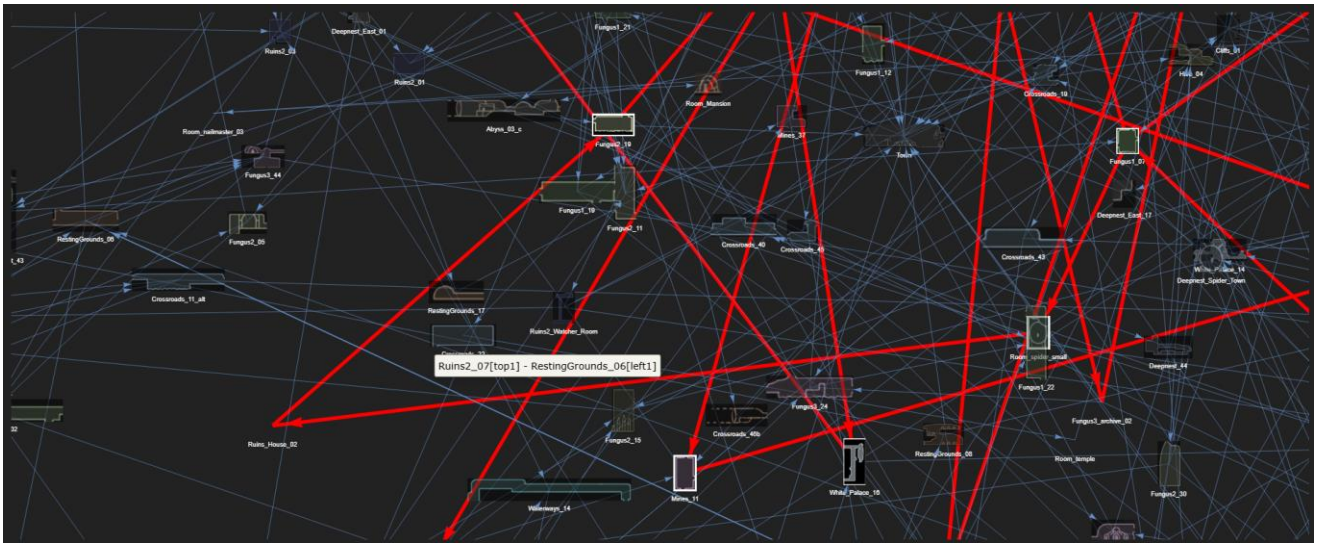
[1] https://discord.com/invite/eAbmKXw4nP
[2] https://www.youtube.com/@BlueSR
[3] https://discord.gg/33eGu7EPU
[4] https://homothetyhk.github.io/RandomizerCore/index.html#randomizer-taxonomy
[5] https://discord.gg/ZHvngqmp
[6] https://github.com/TheMathGeek314/CorpseDreamRando
[7] https://mapgenie.io/hollow-knight/maps/hallownest

APÈNDIX

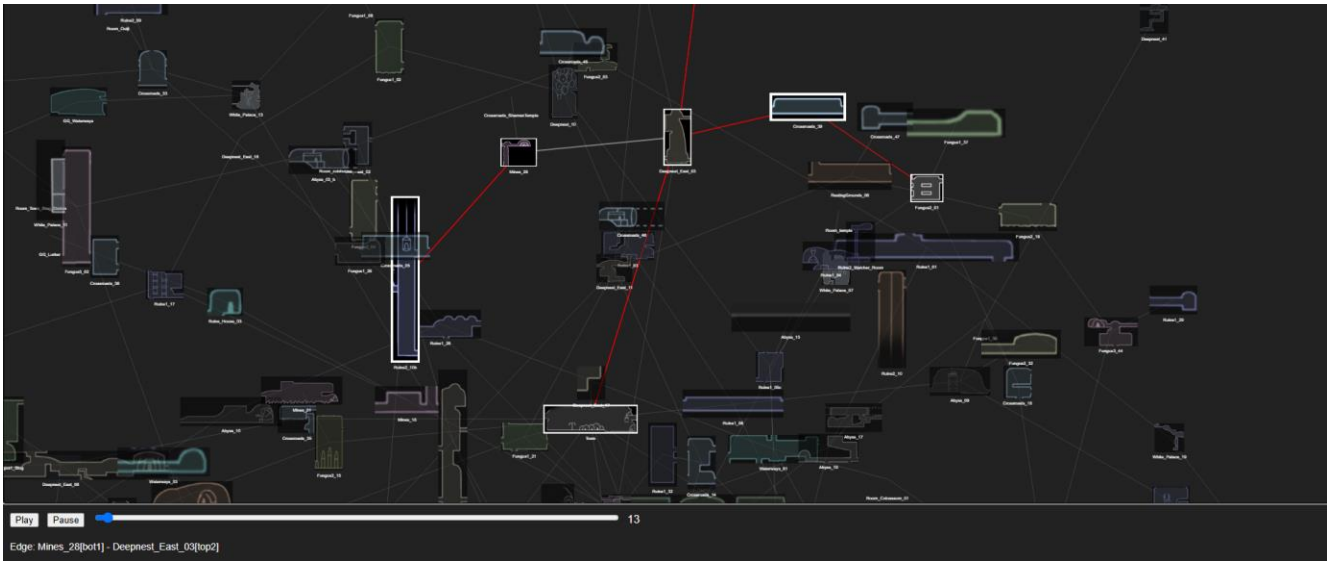
A1. SECCIÓ VISUALITZACIÓ GRAFO_MAPA



A2. SECCIÓ VISUALITZACIÓ GRAFO_DESCUBRIMIENTO



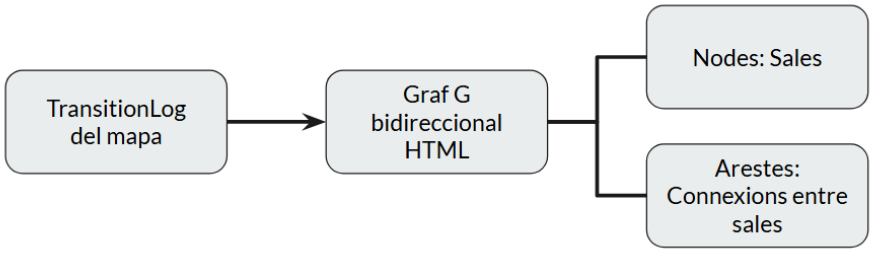
A3. SECCIÓ VISUALITZACIÓ GRAFO_PLAYER_TIMELINE



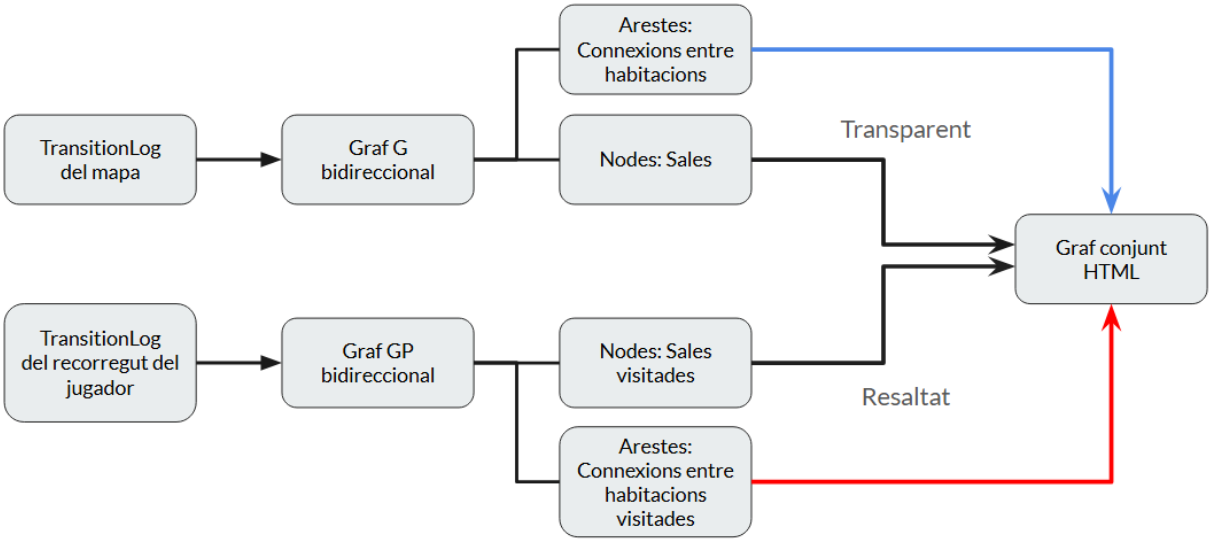
A4. SECCIÓ VISUALITZACIÓ GRAFO_SIGUIENTE_PASO



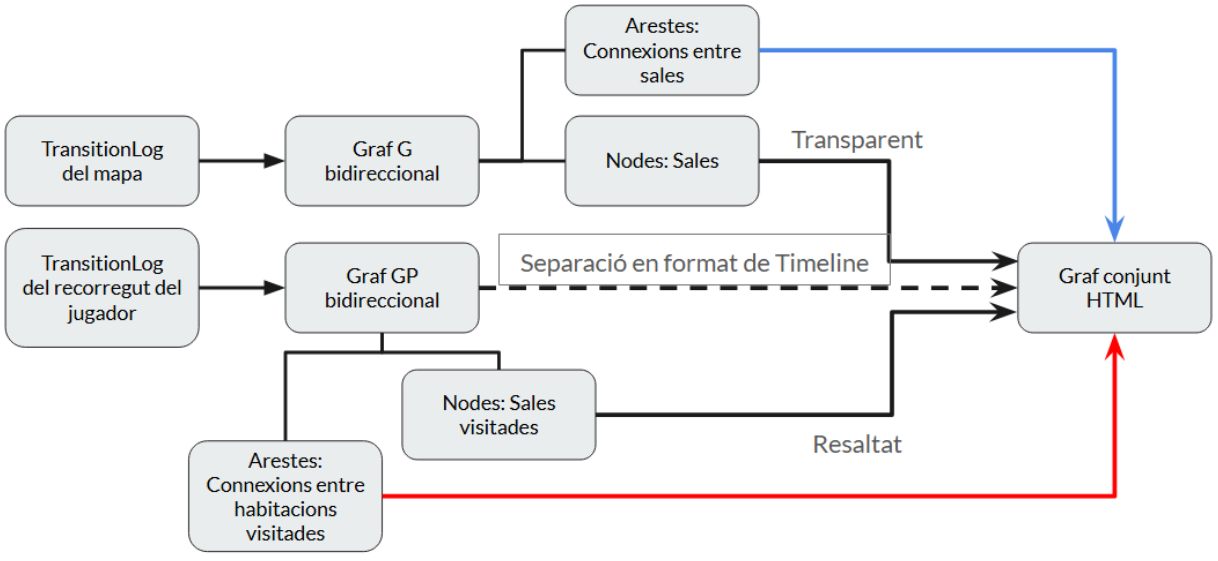
A5. FUNCIONAMENT GRAFO_MAPA



A6. FUNCIONAMENT GRAFO_DESCUBRIMIENTO



A7. FUNCIONAMENT GRAFO_PLAYER_TIMELINE



A8. FUNCIONAMENT GRAFO_MAPA

