

Mario PRO2 (2025 Primavera)

Mario PRO2 és una pràctica a on l'objectiu és programar noves funcionalitats en un joc ja començat.

La preparació de la pràctica (li direm la Part 0) és estudiar amb detall el codi donat, i experimentar amb ell per familiaritzar-s'hi al màxim. L'examen final de PRO2 tindrà un problema sobre la pràctica i és basat en l'estructura del codi inicial donat i en classes amb especificació que es desenvolupin posteriorment.

Després de la "Part 0", hi ha les 3 parts pròpiament dites a desenvolupar:

- Part 1:** Una classe (un objecte del joc) a partir d'uns requisits.
- Part 2:** La classe Finder, amb una especificació donada i que resol un problema d'eficiència en el joc.
- Part 3:** Almenys 2 classes més que implementin noves idees en el joc, o bé eines relacionades amb el joc.

El projecte tindrà 3 entregues, una per a cada part. Cada entrega és incremental i s'acumula per a la següent. La última entrega és el joc acabat. Després de la segona entrega, els professors de laboratori revisaran amb cada alumne l'estat del projecte a la classe de laboratori.

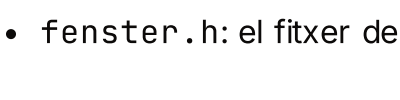
La "Part 0" té un doble propòsit:

- Ser una explicació del codi inicial de la pràctica; i
- Servir com a guió de la **sessió L9** de laboratori, d'introducció a la pràctica.

Per a aquest segon propòsit, en el text hi ha **exercicis intercalats** que pretenen augmentar la comprensió sobre el projecte i fer les primeres modificacions per agafar confiança amb el codi mentre teniu el professor de laboratori a mà.

Part 0: Estudiar el projecte donat (= Sessió L9)

L'arxiu del projecte inicial del Mario PRO2 es pot descarregar en el botó següent:



Per poder entendre el codi que es proporciona, cal llegir el programa i esbrinar com funciona tot. Com més clara estigui l'estructura del projecte i a on està implementada cada funcionalitat, més fàcil serà el desenvolupament.

Excepte la classe Window (fixers window.cc i window.hh), i el fitxer fenster.h que l'acompanya (que tenen codi complex de gestió de finestres en Windows, Linux i MacOS), la resta del codi no té gairebé res que no s'hagi estudiat abans, ja sigui a PRO1 o a PRO2. Una petita part del que cal saber es farà durant la segona part de PRO2, com és el treball amb punters.

Compilació

Descomprimeix el projecte en una carpeta apart i obre VSCode en la carpeta. Tot hauria d'estar preparat perquè al premer **F5** el projecte es compili i s'executi. Ha de sortir la finestra del joc sola.

Per altra banda, des d'un terminal, tens aquestes altres possibilitats:

- make clean, que esborra tots els fitxers intermitjos (*.o i l'executable).
- make tgz, que crea un tgz amb el projecte, per si vols guardar la versió actual. El fitxer creat per tgz es numera amb l'epoch del moment (un número de segons molt gran), de tal manera que no pots sobrescriure un altre tgz que tinguis i que sigui anterior, perquè tindrà un epoch diferent. Recomanem fer make tgz amb certa freqüència i guardar els fitxer en un lloc segur per no perdre versions de la pràctica i poder recuperar versions anteriors en cas d'emergència.
- make MODE=release, aquesta comanda construirà una versió optimitzada del joc, que és possible que tingui més velocitat. Abans de cridar make amb el mode "release" cal fer make clean primer, per tal d'esborrar els fitxers objecte (.o) sense optimitzar.

Arquitectura del projecte

Per una banda hi ha el codi que no cal llegir ni estudiar:

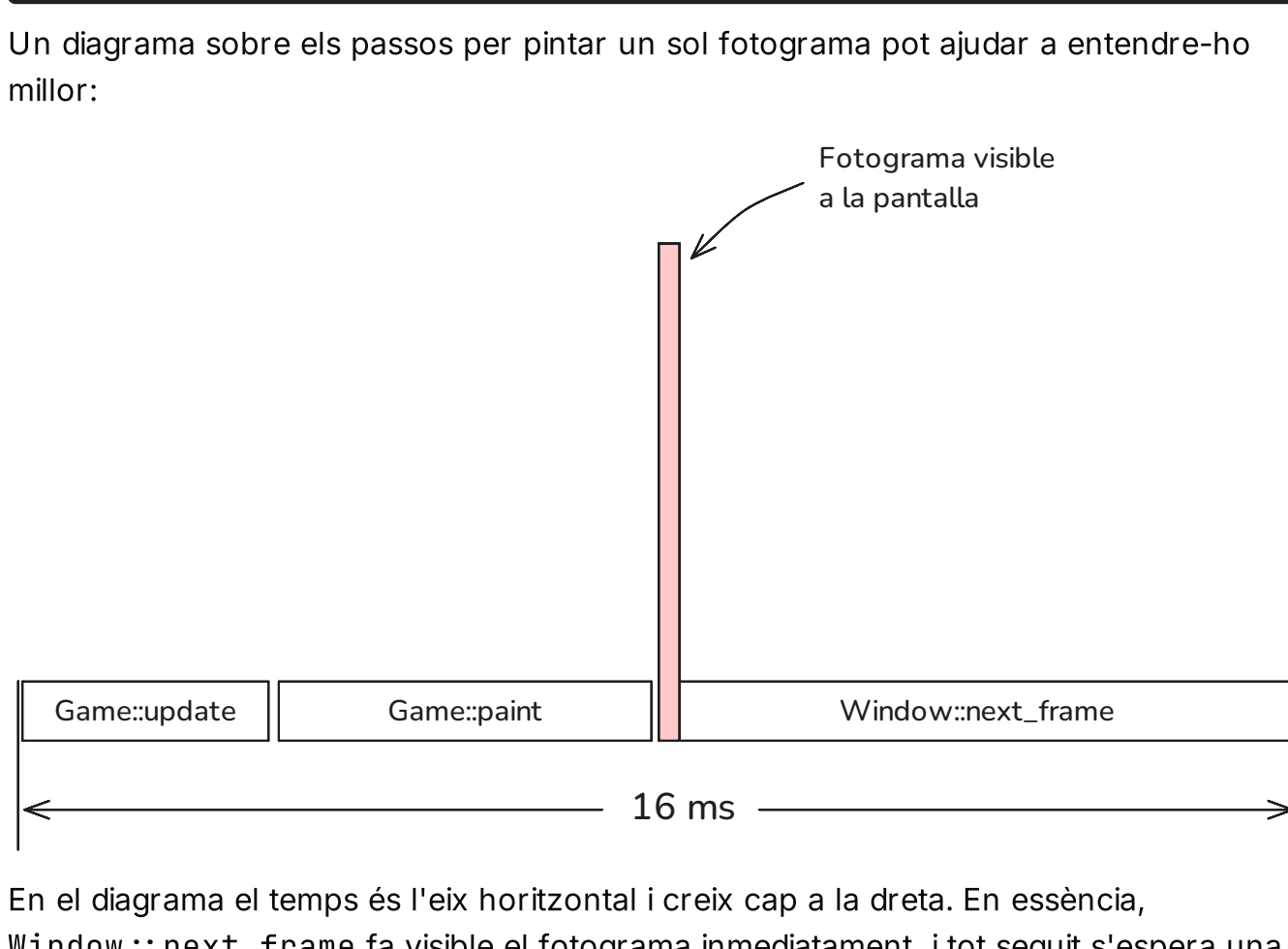
- Window: la classe vista a les pràctiques de principi de curs.
- fenster.h: el fitxer de capçalera que Window utilitza internament.

Després tenim el codi que cal llegir i entendre i que conté els mòduls següents:

- geometry.hh: que només declara les dues tuples Pt (un punt 2D) i Rect (un rectangle 2D).
- utils.{hh,cc}: funcions útils ja implementades.
- Platform: l'objecte plataforma que apareix al joc.
- Mario: el personatge del Mario que apareix al joc.
- Game: la classe que gestiona el joc sencer.
- main.cc: el programa principal, amb el bucle principal del joc.

Els 3 primers mòduls estan a l'espai de noms (o namespace) anomenat pro2. Per tant, per fer servir les classes i mòduls d'aquests mòduls cal: o bé prefixar el nom de la classe amb pro2:: (en fitxers .hh); o bé posar using namespace pro2; (en fitxers .cc).

El següent diagrama mostra les relacions de dependència entre mòduls. Una fletxa des d'un mòdul A a un mòdul B indica que A utilitza B.



El diagrama fa palès que els 3 mòduls de baix de tot són la base sobre la que es construeix el joc (i alhora són els que pertanyen a l'espai de noms pro2). Després hi ha Game, que implementa la funcionalitat del joc. Game utilitza Mario i Platform perquè té membres privats que són el personatge principal i un vector de plataformes que són les plataformes del joc. El diagrama mostra, també, quina posició ocuparia una classe nova (de nom ????) que representi algun objecte del joc (o bé la classe Finder que caldrà fer a la segona part).

Bucle principal

El programa sencer es basa en un bucle principal, molt típic dels jocs, en què es va creant una "pel·lícula interactiva" a base d'anar pintant fotografies i simulant què passa entremig de dos fotogrames consecutius en funció de les tecles premudes, el ratolí, i les pròpies regles del joc.

El bucle és el següent:

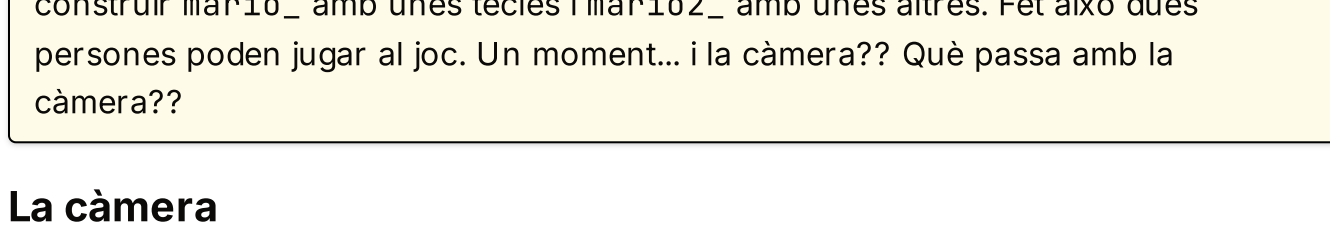
- S'espera fins el proper fotograma window.next_frame().
- Si el joc s'ha acabat o bé s'ha tancat la finestra, es surt del bucle.
- update: S'actualitza l'estat del joc, és a dir, es fa la simulació de moviments i tots aquells canvis que hi hagi hagut durant el temps entre l'últim fotograma i l'actual.
- paint: Es pinta l'estat del joc a la pantalla.
- Es torna al pas 1.

La classe Game té, precisament, dos mètodes anomenats update i paint que es corresponen amb els passos 3 i 4, i també is_finished que decideix parcialment si el joc ha finalitzat.

El codi del bucle és, exactament:

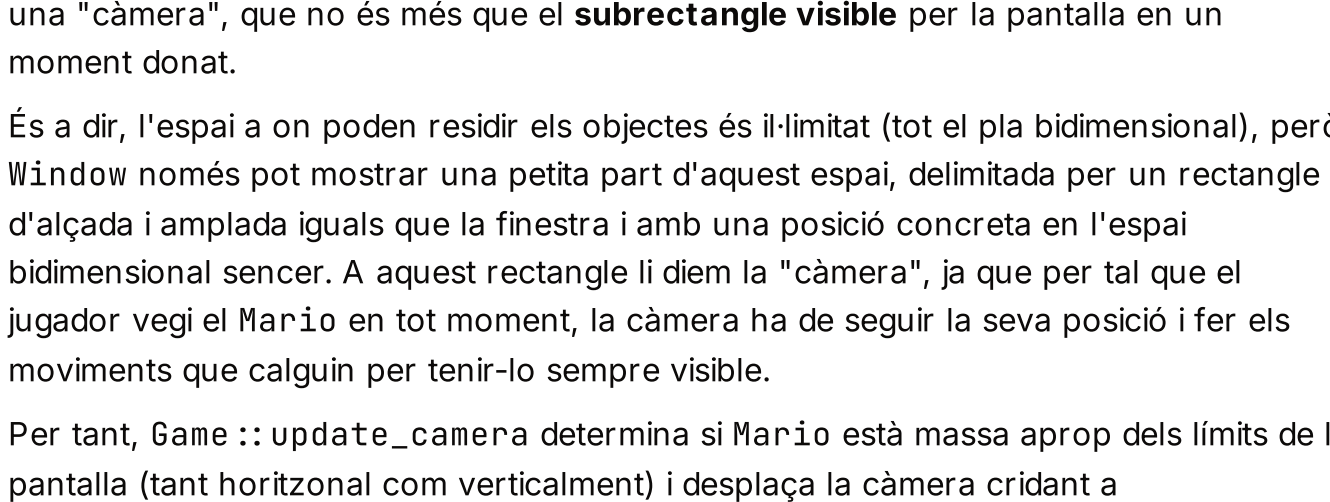
```
while (window.next_frame() && !game.is_finished()) {
    game.update(window);
    game.paint(window);
}
```

Un diagrama sobre els passos per pintar un sol fotograma pot ajudar a entendre-ho millor:



En el diagrama el temps és l'eix horitzontal i creix cap a la dreta. En essència, Window::next_frame fa visible el fotograma immediatament, i tot seguit s'espera una quantitat de temps que es calcula amb respecte a la última crida a Window::next_frame, per tal de poder ajustar el temps entre fotogrames, que sol ser de 16ms. Aquests 16 mil·lisegons donen lloc a la freqüència necessària per produir 60 fotogrames per segon (1000ms / 60 fotogrames). 60 FPS es considera l'estàndard perquè hi hagi sensació de fluïdesa suficient en un joc.

La seqüència de fotogrames, doncs, seria una repetició d'aquest patró:



Game::update

El mètode Game::update té una estructura que ha de resultar entenedora:

```
void Game::update(Window& window) {
    process_keys(window);
    update_objects(window);
    update_camera(window);
}
```

Primer es processen les tecles (Game::process_keys) que s'han premut entre el fotograma anterior i l'actual. Window té mètodes per determinar quines són. La implementació de Game::process_keys només detecta la tecla "Escape" i això és el que Game fa servir per indicar al programa principal que el joc s'ha acabat.

Exercici 0.1: Fes els canvis necessaris al codi del projecte que permeti pausar el joc amb la tecla 'P', i tornar-lo a engegar tornant a prémer 'P'. Pausar el joc vol dir que l'acció queda congelada en el mateix fotograma permanentment, fins que no es torni a posar en marxa. Això implica que no s'actualitzen els objectes ni la càmera, però sí que se segueix pintant el joc (i es processen les tecles). Els canvis en aquest exercici són: a) afegir un camp nou a la classe Game, i modificar un o més mètodes existents.

Després del mètode Game::update_objects crida el mètode update de cada objecte que hi ha al joc. Això és una manera de delegar la feina a cada objecte particular. Els objectes del joc que necessiten update són aquells que no són estàtics i canvien de posició o tenen animacions, com ara el Mario. En canvi, com que les plataformes són fixes, no tenen mètode update i Game no necessita cridar-los.

Exercici 0.2: Fes els canvis necessaris (de nou afegir membres i canviar mètodes) per tenir un altre Mario al joc. Abans de fer-ho, pensa bé si es tracta d'una nova classe o d'un nou objecte! Posa'l com a membre de la classe Game amb nom mario2_. Inicialitza'l amb una coordenada x uns 30 píxels a l'esquerra del mario_ principal. Busca els llocs a on apareix el primer Mario i fes les mateixes crides als mateixos mètodes que els que es fan amb mario_. Abans de provar-ho, què penses que succeirà quan executis? Els dos Marios són independents? Ja tenim un joc per a dos jugadors?

En general, per afegir qualsevol objecte al joc, doncs, el mínim que cal fer és posar-lo com a membre de la classe Game i afegir les crides a les operacions update i paint corresponents dins dels mètodes Game::update i Game::paint.

Exercici 0.3 (difícilment mitjana): Fes que mario2_ es pugui controlar amb tecles diferents que mario_. Caldria primer modificar la classe Mario de la següent manera. Primer cal que Mario pugui guardar les 3 tecles que el controlen en camps privats, que es poden dir jump_key_, left_key_, i right_key_. En el projecte inicial, les tecles són fixes i són l'espai i les fletxes d'esquerra i dreta. Pel mario2_ cal que siguin "W" per saltar, "A" per l'esquerra i "D" per la dreta (això permet jugar a dues persones). Fixa't que pots passar un caràcter als mètodes Windows::is_key_pressed i Windows::was_key_pressed perquè es fan servir els codis ASCII.

Els valors d'aquestes 3 tecles s'haurien de passar al constructor de Mario i allà guardar-les en els 3 camps privats. Després cal anar al codi que comprova les tecles que s'han premut i treure els valors fixos i posar-hi els camps privats, que tenen les tecles passades en el constructor. Finalment, en el constructor de Game, caldria construir mario_ amb unes tecles i mario2_ amb unes altres. Fet això dues persones poden jugar al joc. Un moment... i la càmera?? Què passa amb la càmera??

La càmera

Seguint amb Game::update, un cop actualitzats tots els objectes al mètode Game::update_objects, hi ha una crida a Game::update_camera. Aquí és important explicar amb més detall què és la càmera i com funcionen les coordenades en el joc.

El joc fa servir un sistema de coordenades absolut, amb coordenades enteres i amb l'eix x creixent cap a dreta (com en matemàtiques), però l'eix y **creixent cap a baix** (al revés que en matemàtiques). Malgrat aquest sistema de coordenades és força confús per algú que ha estudiat matemàtiques primer, el món dels videojocs i els sistemes de finestres han fet servir aquesta convenció des del principi i s'ha anat consolidant amb el temps.

Així doncs, els objectes del joc poden tenir coordenades enteres arbitràries, tant negatives com positives, i el que determina si són visibles o no és el fet que Window té una "càmera", que no és més que el **subrectangle visible** per la pantalla en un moment donat.

És a dir, l'espai a on poden residir els objectes és il·limitat (tot el pla bidimensional), però Window només pot mostrar una petita part d'aquest espai, delimitada per un rectangle d'alçada i amplada iguals que la finestra i amb una posició concreta en l'espai bidimensional sencer. A aquest rectangle li diem la "càmera", ja que per tal que el jugador vegi el Mario en tot moment, la càmera ha de seguir la seva posició i fer els moviments que calguin per tenir-lo sempre visible.

Per tant, Game::update_camera determina si Mario està massa aprop dels límits de la pantalla (tant horitzontal com verticalment) i desplaça la càmera cridant a Window::move_camera, que no mou la "càmera" instantàniament, sinó que distribueix el moviment en uns quants fotogrames, per tal que sigui més suau.

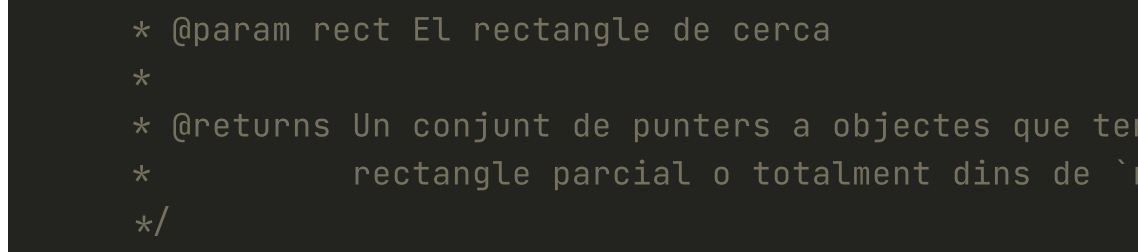
Exercici 0.4: Simplifica el codi de Game::update_camera per tal que la càmera sempre tingui al Mario al centre de la pantalla. És perquè, independentment del tipus de Mario, la posició central de la càmera, i cridar Window::move_camera indicant la distància entre la càmera i el Mario. Executa el joc i prova a saltar a les plataformes. Et sembla que la càmera té un moviment bo per jugar?

Game::paint

El mètode Game::paint és el que fa servir les funcions de pintat de Window per pintar tots els píxels del joc. L'**ordre** en què es pinta cada objecte és molt important, perquè si dos objectes A i B se solapen en l'espai (com ara el Mario atravesant una plataforma de baix a dalt), pintar A i després B implica que B es veurà en primer pla, i A es veurà al seu darrere.

Exercici 0.5: En aquest exercici farem servir les utilitats del mòdul utils.hh anomenades paint_hline i paint_vline que pinten una línia horitzontal i una línia vertical, respectivament. Es tracta de d'afegir codi a Game::paint per dibuixar un requadre al marc de la pantalla, però en comptes de posar coordenades fixes i que el requadre sembli que està en el món del Mario, cal que el requadre es mogui amb la càmera i sembli que realment és un requadre que es mou amb la pantalla. Pinta el requadre l'últim de tots o bé just després de window.clear(...) per veure la diferència. En aquest exercici hauràs de buscar-te tu mateix els mètodes que et permetin fer el que es demana!

Per exemple, en la següent imatge, en què Mario està atravesant una plataforma desde sota, es veu com cal pintar el Mario després de la plataforma perquè surti en primer pla.



El codi de Game::paint efectivament pinta el Mario (cridant a Mario::paint) en l'últim lloc.

Exercici 0.6: Afegeix una funció a util.cc que es digui paint_rect, que rebi un rectangle (de tipus Rect) i un color i pinti tots els píxels del rectangle del color (mira primer paint_hline o paint_vline com a inspiració). Recorda declarar la funció a utils.hh!

Exercici 0.7: Utilitzant paint_rect, pinta un rectangle de color groc al mig de la pantalla a sobre de tots els altres objectes. Prova també de posar el rectangle al fons de tot, per sota inclús de les plataformes.

Exercici 0.8: Com faries perquè el rectangle groc fes pampallugues, és a dir, que aparegués i desaparegués ràpidament en seqüència?

Ho has entès tot?

Arribats aquí, cal que revisis la **lista de preguntes d'aquí sota**, i t'autoavaluis, per veure si les pots contestar. Això et demostrarà si has entès l'estructura del joc i veus com poder-hi fer modificacions pel teu compte, un pre-requisit per poder començar la **Part 1**.

- Què hi ha a cada fitxer del projecte.
- On està el bucle principal del joc i com funciona.
- Com són les tuples Pt i Rect i què representen.
- On es creen els objectes del joc.
- On està el codi que pinta cada objecte del joc.
- Perquè serveix el mètode Game::update.
- Quina tecla acaba el joc.
- Com se sap si el joc ha acabat.
- On està el codi que mou la càmera.
- Com es canvia: el tamany de la finestra, el número de píxels reals per cada píxel del joc (el "zoom"), o els fotogrames per segon.
- Com afegir plataformes al joc.
- Es pot canviar el tamany d'una plataforma un cop creada?
- On s'han d'afegir objectes nous al joc.
- On posar detecció de tecles.
- Com canviar la textura de les plataformes.
- Com canviar la força de la gravetat amb què el Mario cau.
- Com canviar la direcció en la que mira el Mario quan prems les fletxes horitzontals.
- Com s'aconsegueix que el Mario no pugui cap a una banda o cap a l'altra.
- Quines variables i mètodes estan involucrats en el fet que el Mario pugui saltar i com s'aconsegueix.

Cada una d'aquestes preguntes no és només una qüestió conceptual, sinó que la millor resposta és saber fer modificacions al codi que demostren que teniu controlat aquell aspecte del joc.

Part 1: Un nou tipus d'objecte

Es tracta de fer un nou tipus d'objecte que el Mario pugui recollir. El joc original tenia monedes, però pot ser qualsevol altra cosa.

Els requeriments d'aquest objecte són:

- Que n'hi pugui haver molts en el joc, en llocs accessibles (o no tantl).
- Que quan el Mario passa pel damunt els agafi implícitament i per tant, desapareguin de la pantalla.
- Que es tingui un comptatge de quants objectes s'han agafat. Donat que el codi inicial no proporciona utilitats per mostrar text, no cal mostrar el comptatge per pantalla textualment.
- Que tinguin una animació senzilla. Poden "flotar" en l'aire amb moviments verticals de tipus sinusoidal, o bé rotar (cosa que implica fer "sprites" diferents i anar-los mostrant consecutivament), o qualsevol altra cosa que atregui l'atenció del jugador.

ENTREGA 1: Cal fer una classe nova pel nou tipus d'objecte, tal com s'ha explicat, i estructurar-la de tal manera que encaixi perfectament amb la resta del joc. Com ha passat amb els exercicis de la Part 0, es pot modificar la classe Game i afegir-hi dades noves o mètodes nous (tot i respectant la lògica actual), apart de fer la nova classe.

La primera entrega es farà al Racó i la data límit serà l'**11 de Maig de 2025**.

Part 2: Eficiència

Aquesta part comença amb una constatació molt trista, i és que el joc **no és prou ràpid**: si posem uns pocs centenars de plataformes (cosa totalment normal en un joc com Déu mana), els FPS cauen en picat, perquè, mantenint la forma actual del joc, resulta que es pinten tots els objectes inclús si no estan visibles a la pantalla.

Exercici 2.1: Afegeix milers de plataformes al joc i comprova com el joc es torna tant lent que no es pot jugar.

Per tant, o bé fem alguna cosa o el projecte s'acabaria aquí... nooooo!

Que no cundeixi el pànic... el que farem és introduir una nova classe que ens permetrà només pintar a la pantalla aquells objectes que són visibles, ja sigui total o parcialment. D'aquesta manera, encara que el "mapa" del nivell sigui enorme (amb milers de plataformes), com que en un moment donat només n'hi ha visibles uns quantes (màxim 50, com a molt estirari), llavors el joc funcionarà perfectament.

La classe que volem hauria de tenir la següent funcionalitat:

- Serà un **contenedor**, és a dir, contindrà objectes (del joc). Però només contindrà els punters als objectes, no els objectes en si. Això és perquè, independentment del tipus d'objecte que tinguem, si aquest objecte té el mètode get_rect, el nostre contenedor ja es podrà espabilar i classificar els rectangles d'alguna manera, sense haver de conèixer els altres detalls.
- Farà la relació que els objectes que es posin al contenedor **tenen un mètode get_rect()**, que retorna un Rect. Això és perquè, independentment del tipus d'objecte que tinguem, si aquest objecte té el mètode get_rect, el nostre contenedor ja es podrà espabilar i classificar els rectangles d'alguna manera, sense haver de conèixer els altres detalls.
- Tindrà 4 mètodes: afegir, esborrar, actualitzar i consultar. Aquests mètodes permetran:
 - Afegir un objecte al contenedor.
 - Esborrar un objecte del contenedor.
 - Actualitzar un objecte que ja estava al contenedor, que vol dir actualitzar el seu rectangle, realment.
 - I finalment, el mètode més important: **consultar els objectes de dins el contenedor que estiguin dins d'un cert rectangle**. Aquest mètode el cridarem amb les coordenades de la càmera per saber quins objectes són visibles.

A aquesta classe li direm Finder, i tindrà la següent declaració:

```
template <typename T>
class Finder {
    // ...
public:
    Finder() {}

    void add(const T &t);
    void update(const T &t);
    void remove(const T &t);

    /**
     * @brief Retorna el conjunt d'objectes amb rectangles
     *         total o parcialment dins de 'rect'.
     *
     * Si el nombre de rectangles del contenedor és 'n', el
     * cost de l'algorisme ha de ser O(log n).
     *
     * @param rect El rectangle de cerca
     *
     * @returns Un conjunt de punters a objectes que tenen un
     *         rectangle parcial o totalment dins de 'rect'
     */
    std::set<const T &> query(pro2::Rect rect) const;
};
```

Serà un template ja que ens agradaria tenir un Finder per a cada tipus d'objecte (caldrà un Finder diferent per a cada cadascun, de fet). La única cosa que cal que tingui el tipus T, que són els elements de dins del Finder és el mètode get_rect esmentat (que Platform ja té, si mirem).

ENTREGA 2A: Als voltants del dia 8 de Maig sortirà un entregable al Jutge per poder enviar la classe Finder i que aquesta sigui comprovada pel Jutge, de forma que es tingui una certesa major sobre la seva correctesa i sobretot eficiència. Aquesta entrega es correspon al D2 es farà al Jutge mateix, tal com amb el D1.

ENTREGA 2B: Caldrà modificar el projecte per fer servir el Finder amb els objectes que hi ha al joc (tots?) i comprovar que amb milers d'objectes tot funciona sense problemes i de forma fluïda. Pensa bé a on convé que estigui i si cal un o més Finders. Aquesta entrega es farà al Racó, tal com la primera.

La data límit d'aquesta segona entrega serà el dia **25 de Maig de 2025**.

Part 3: Acaba el joc com més t'agradi

La tercera entrega, la final, serà una entrega que "acabi" el joc de la manera que més gràcia us faci. Aquesta part és lliure intencionadament, perquè es tracta d'implementar funcionalitat nova a sobre de l'estructura establerta, i que cadascú sigui lliure d'avançar en la direcció que més li interessa. Es poden copiar característiques de jocs coneguts (del propi Mario, es clar), inventar noves idees que siguin molt boques, i en general experimentar i treballar els temes estudiats a PRO2 en un entorn en el qual la mandra no ens desconcentri. També és important que cadascú faci una cosa personal i que els projectes siguin diferents entre si.

Les úniques restriccions a la tercera entrega són:

- Que es facin almenys dues classes noves (se'n poden fer més).
- Fer servir algun contenedor dels estudiats a la segona part de PRO2, potser fent-hi modificacions per adaptar-lo als requeriments del que es vol implementar en el joc. (Aquest contenedor no compta dins de les dues classes noves a no ser que sigui molt diferent dels explicats.)
- Enviar un petit vídeo, de 10 o 20 segons, amb un moment del joc en funcionament.

Una cosa força necessària a la Part 3 és consensuar amb el vostre professor de laboratori que el plantejament que esteu fent és factible i té sentit, i ja tinguem una garantia d'èxit mínima. Això és important per evitar fer propostes massa optimistes que després costen molt més d'implementar del que semblava en un principi (encara que no ho sembli, és el més freqüent!). A més, saber que ja teniu l'aprovació del professor és ideal perquè ell mateix us corregirà la pràctica.

ENTREGA 3: L'entrega 3 es farà pel Racó.

La data límit d'entrega del projecte acabat serà el **8 de Juny de 2025**.