Memòria del projecte de fi de carrera VirtualBot

Xavier Moreu Reñé

Índex:

El que volia que fes el programa i el que fa:

En la PAC 1 vaig especificar el que volia implementar sense saber molt bé com ho anava a fer i quines complicacions tindria.

La meva intenció era fer un programa fet en entorn web que servis per aprendre a programar una sèrie de robots virtuals que es mouen en un entorn 2D a base de quadrants. La idea es que l'usuari posi robots en un pla 2D que son dibuixos o sprites i que mitjançant una Caixa de text al costat, es programi el codi de cada robot i posteriorment es pugui compilar i executar.

En aquest entorn també vaig especificar que es podrien posar murs i que el robot detectaria la col·lisió dels murs i no podria continuar avançant. També vaig dir que es podrien posar balises que el robot podria agafar i traslladar. El programa resultant ha estat una mica més ampli perquè efectivament compleix amb tot el descrit i a més a més es poden posar com una mena d'interruptors de terra que el robot pot activar deixant-hi una balisa a sobre. El robot té una instrucció mitjançant la qual pot valorar l'estat d'un interruptor; si està activat o no. La idea d'aquesta nova funcionalitat era la de que el usuari poses en pràctica la idea dels esdeveniments i dinamitzar més l'entorn i la cooperació entre robots.

En la PAC 1 i dos, vaig especificar que volia que un usuari pogués entrar en entorns creats per altres usuaris i veure com robots que hauria programat un usuari determinat, es mourien sense poder accedir al codi. D'aquesta manera el usuari podria programar els seus robots i fer cooperacions o lluites o el que sigui amb els robots d'un altra usuari. La idea era bona crec però no l'he posada en pràctica per varis motius. El motiu més gran és per falta de temps, tot això requereix una sèrie de operacions sobre la base de dades, sobre el codi de javascript (més endavant en parlaré) etc que no eren ni trivials, i portaven un cert temps. El segon motiu és perquè em vaig centrar més en les funcionalitats dels robots i en el codi que s'havia de compilar que crec que és el més important i el fet de que es puguin gravar els entorns en una BD em va portar temps. En la PAC 1 a més vaig especificar com serien les instruccions una vegada compilades per el compilador virtual dient que es basarien en nombres enters que especificaven el tipus d'instrucció (mitjançant un codi) i els operadors que també serien valors enters. En la PAC 2 ja vaig dir com seria perquè ja havia

programat part del que és el compilador virtual. Al ser un projecte força complex, em veig amb la necessitat de explicar què fa el programa i després ja entraré en el com ho fa.

Hi ha una plana web on posa una mena de glossari resum de totes les instruccions que el compilador interpreta, això es podria ampliar molt més de cara a una publicació del programa real ja que l'usuari necesita saber com programar en aquest llenguatge fictici i com actua el compilador. D'entrada les instruccions ja compilades es poden veure en un listbox a sota del editor de codi i podem veure que no son tot nombres sinó paraules curtes i nombres.

Tenia molt clar que volia fer un codi virtual que tingues plena funcionalitat i que admetés l'ús de funcions, variables, matrius etc. Per fer això vaig pensar en crear una memòria RAM virtual que no seria més que un array de nombres amb coma flotant. Vaig decidir que fossin en coma flotant perquè es poden expressar nombres molt grans o petits amb una gran eficiència en quan al processador (real) i a l'espai que ocupen. Vaig pensar que si fossin nombres 'double' ocuparien més memòria i es processarien molt lent. La velocitat de processament de les instruccions virtuals és crucial perquè influeix molt en el càlcul de operacions i pot fer que es vegi el robot anant més lent.

La declaració de variables volia que fos el més senzilla possible. El codi virtual ha acabat sent una mescla de codi dalt nivell i de assemblador. Les variables es declaren amb el terme var seguit del nom de la variable i el que fa el compilador es afegir-la a un array que relaciona el nom de la variable amb la posició de memòria de la RAM virtual. Vaig fer que es poguessin definir variables (var nomVar=3,7,4,97;) D'aquesta manera es podien crear arrays amb l'apuntador de memòria 'nomVar' que apunta a la posició 1 del array de valors. Recordem que no es poden posar caràcters, només nombres en coma flotant. També vaig fer que es poguessin implementar de la manera (var nomVar=(6)) on el compilador el que fa és afegir a la memòria RAM una sèrie de valors 0 en n posicions, en aquest cas 6.

Després vaig començar a crear les instruccions aritmètiques com la suma, resta, multiplicació i divisió i més endavant hi vaig afegir lògiques com la 'and' i l''or' que després explicaré perquè son útils.

No m'estendré gaire perquè amb el glossari que inclou el projecte es poden veure exemples etc, el que vull deixar clar és la estructura de una instrucció compilada. Vaig meditar-ho molt en les anades i tornades de la feina ☺ i vaig arribar a la conclusió que tenien que tenir un tipus i un subtipus que el processador virtual llegís i interpretés. Seguidament, la instrucció tenia uns operands, en concret tres. Tres perquè el màxim de operands que requereix una operació com suma son tres variables o nombres. Per exemple, add(R0,R1,R2) suma el valor que hi ha a les posicions RAM R1 i R2 i el resultat les posa a la posició destí R0. Observem que per tant un operand pot ser una posició de memòria o un nombre. Podria ser una suma add(R0,4,6) on posa 10 en la posició R0. També pot ser que la posició de memòria fos especificada per el operador R1[3]. Aquest operador especifica la posició de memòria que està en la posició R3 + 3 de la memòria RAM virtual. Per tant cada operand té fonamentalment tres tipus de tractament. Aquest tractament o tipus el vaig especificar en l'objecte Operand amb el paràmetre tipus. Una instrucció per tant, add(R0,R1[1],5) tindria el format de instrucció compilada seguent:

"op" , "add" Tipus i subtipus

Op1: Tipus :1 ,adressa :3; Tipus de operador 1 que es refereix a una posició de memòria que en aquest cas és la 3 perque suposem que R0 = 3.

Op2: Tipus : 2 , adressa:4,i:1; On tipus d'operador és el 2 que especifica que s'ha de sumar el valor de l'adreça amb el i que especifica el valor 5 de la memòria RAM.

Op3: Tipus: 0 , n:5; On especifica que no és una posició de memòria sinó un nombre, en concret el 5.

Suposem que volem fer un salt en el codi, la instrucció per fer-ho seria:

Tipus:jump

Subtipus: >= “Descriu el tipus de comparació que es farà entre el segon i tercer operador

Op1: tipus:1,adressa=4 “Refereix a la posició que saltarà si es produeix la condició”

Op2 i op3 contindrien els valors a comparar. Poden ser nombres o posicions a la memòria com ja hem vist.

A la PAC 1 i 2 tenia en ment que el codi virtual només fos de tipus assemblador, en el que el compilador només fa que traduir una cadena String a la instrucció de codi màquina virtual. Per exemple, add (R0,R0,1) la traduiria a un objecte de la classe Instrucció amb els paràmetres i subobjectes Operand que pertoqui. La compilació llavors és ràpida i instantània. A la hora de fer els meus primers programes vaig detectar que era molt farregos programar posant etiquetes de salt de codi per tot, per fer un bucle, per fer un condicional etc. Es enutjós perquè has de definir una etiqueta diferent per cada salt que es produeix. Llavors vaig tenir la idea de posar una sentencia anomenada lblini i lblend. En mig d’aquestes dues podia repetir el nom d’etiquetes anteriors i el programa ho interpretaria i ho compilaria interpretant les etiquetes com a úniques. Aquesta solució no hem va semblar satisfactòria, es pot fer, l’he deixat programada, però volia anar més enllà i fer que l’usuari pogués programar en un llenguatge més proper al natural amb sentències while, if...

El problema era com fer el algoritme que transformés unes sentències:

While(a<4)

Forward

Add(a,a,1)

Endwhile

A les instruccions de màquina pertinents. El primer que vaig comprendre és que la condició while en aquest cas és una condició de continuïtat del flux nor-

mal del programa(anar cap endavant executant la instrucció ‘n + 1’ a continuació) Per tant s’ha de produir un salt si NO es compleix la instrucció cap a la posició de memòria del ‘endwhile’.

La solució era traduir el while per la instrucció de màquina virtual jump a>=4 [endwhile]. Observem que faig un salt si la condició és la inversa. I en la posició on està el endwhile faig un salt incondicional a la posició del while. Per tant era més senzill del que pensava i vaig afegir les instruccions “While....endwhile” i les “if....else....endif”.

L’altra repte amb el que m’afrontava era com gestionar les disjuntives i conjuntives de les condicions de salt del while o del if etc. Vaig començar a reflexionar sobre el tema i la solució estava per construir els salts valorant les disjuntives i conjuntives de una expressió dins de un while o if. Però era molt complicat i no disposava de molt temps per generar un algoritme per fer-ho, es pot fer més endavant en la versió 2.0 però ara per ara ho vaig descartar. La manera de com gestionar i compilar les disjuntives i conjuntives de una expressió condicional la vaig deixar en mans del usuari que té que usar les sentències COMP, AND i OR manualment.

Un exemple de codi seria:

Comp(R0,R1<5) .....Aquesta instrucció compara el valor que hi ha a R1 amb 5 i escriu a R0 si es produeix verdader o fals (un 1 o un 0).

Comp(R2,6=9)

And(R0,R0,R2).... Fa una operació and del operadors R0 i R2 on s’han gravat els valors true o fals de les comparacions.

If (R0=1);.....;endif

Com veiem el que fan les sentències és if (R1<5 and 6=9); .....;endif;

Podem fer d’aquesta manera operacions lògiques el complexes que vulguem. Pot resultar enutjós però és la manera que vaig trobar més senzilla de implementar en el compilador.

Finalment i no m'estendré més, el tema de les funcions era tot un repte posar a la pràctica la seva implementació. Al fer exemples de codi em vaig veure amb la necessitat de que es poguessin elaborar funcions ja que moltes vegades els robots han de fer operacions repetitives com ara anar endavant un cert nombre de quadrants, cercar una balisa en un espai etc. A més a més, la finalitat del programa és que serveixi d'eina d'aprenentatge de programació i l'ús i declaració de rutines es fa necessari.

Un ordinador té el que s'anomena la pila, que és un tipus de metodologia per guardar valors en memòria en el que mitjançant dues instruccions (PUSH i POP), es guarden en un espai de la RAM els valors com si es tractes de una pila de monedes. No m'estendré més perquè sé que estic sent massa pedagògic. El cas és que aquesta metodologia serveix per la crida de subrutines perquè podem anar guardant la posició de la qual es crida a la pila i recuperar-la després quan la funció acaba i tornar a la posició de codi on es va quedar. A més la pila serveix per passar-li còmodament paràmetres a les funcions o perquè retornin valors.

Així que per declarar una funció i cridar-la passant-li paràmetres es farà així:

```
while(R0<4)
push(5)
call(endavant)
right
add(R0,R0,1)
endwhile
end

function endavant
pop(R1)
mov(R0,0)
while(R0< R1)
forward
add(R0,R0,1)
```

```
endwhile
return
```

Com veiem, el programa te una funció que fa anar el robot tantes passes endavant com li passa la sentència push abans d cridar la funció.

No m'estendré més en la programació virtual però creia necessari més que res explicar els problemes i idees que m'han sorgit pel que fa a la programació dels robots.

Un altra aspecte era la part de edició gràfica de l'entorn del robot, posicionament dels robots dins del entorn etc. Com deia en la PAC 2, els casos d'ús incloïen el fet de poder inserta murs, robots i balises i poder-los suprimir o modificar. Pel que fa a la inserció de qualsevol element, vaig necessitar dissenyar uns botons de tipus toggle els quals es tenien que quedar polsats com qualsevol editor de dibuix. La manera com ho vaig dissenyar ho descriuré més endavant en l'apartat "Com es va fer". El cas és que volia que hi haguessin pocs botons perquè fos el més senzill possible i es va quedar en "Seleccionar", "Robot" , "Mur", "Balisa", "Interruptor" i "Borrar".

Els botons "Robot, Mur, Balisa,Interruptor" afegeixen un objecte dels tipus que descriuen al mapa però vaig veure que la manera d'afegir dels abjectes "mur" tenia que ser més àgil i vaig haver de fer-ho de manera que l'usuari polsés el mouse i al moure 'l per la pantalla s'incloguessin tants elements murs a la posició del mouse tenint en compte que no podia incloure's cap objecte nou si ja hi havia un objecte (sobre escriptura). En canvi la inclusió de objectes "Robot", "Balisa" o "Interruptor" es fa polsant el mouse un cop en un quadrant determinat. El cas d'ús de esborrat es fa un per un sobre els objectes en que es posiciona el mouse.

El cas d'ús modificar objecte, només el vaig aplicar al objecte "Robot" i no als dames. Per modificar la posició original d'una balisa per exemple, cal eliminar la balisa i tornar a crear una de nova. El objecte "Robot" en canvi, es pot

seleccionar i editar el seu codi, canviar-li la posició inicial (que la té) el seu nom identificatiu etc. En aquest aspecte no es desdiu molt dels casos d'ús que es podien fer. També es poden eliminar els entorns o mapes en la plana de' Zona d'Usuari'.

En la PAC 2 vaig dir que es podrien carregar els codis dels robots i gravar-los independentment dels robots als quals s'adjudica. Però al final ho he fet de manera que es puguin copiar codis d'un mateix usuari que pertanyen a un robot determinat, a el robot que s'està editant. D'aquesta manera no s'havia de crear una taula nova a la BD amb els codis com a camp que seria redundant. El que es fa és fer una consulta SQL que dóna els codis de tots els robots d'un mateix usuari, i aquest selecciona el que vol i el còpia al codi que està editant,.

En quant a els casos dus de registre d'usuaris etc, el programa permet registrar-se a la web posant unes dades mínimes i identificar-se. En la BD del projecte també es podria gravar la data de registre i estadístiques d'ús com les vegades que l'usuari entra, etc. Tot és millorable. A la hora de estar editant i programant robots, vaig voler fer que les opcions de anar a la zona d'usuaris i des identificar-se o crear un nou mapa estiguessin en un menú superior visible en tot moment. Em sembla molt clar i pràctic per l'usuari.

A continuació resumiré el que el projecte fa:

- Es pot dibuixar un entorn, un mapa, a on els robots poden explorar sent programats per l'usuari.
- Aquests entorns poden ser gravats en una base de dades en el servidor.
- Els tipus de objectes que es poden posar en un mapa son: Robots, balises, interruptors de balises i fragments de mur.
- El programa ha de permetre el disseny de l'entorn en 2D i per quadrants, restringint-se el moviment dels objectes pels quadrants com un tauler d'escacs
- Es poden afegir més d'un robot i es podran programar independentment en un codi fictici. Aquest codi es compilarà podrà ser executat independentment o tots a la vegada.

- Cada Robot té una sèrie de funcionalitats. També és poden moure manualment per la pantalla.
- Es poden visualitzar en tot moment l'estat de la memòria RAM, la instrucció que s'executa en aquell moment i estadístiques com nombre de col·lisions etc

Funcionalitats d'un robot:

- Direccionalitat: Endavant, esquerra, dreta, enrere.
- Acció:
    - Incorporen una pinça per agafar i desar les balises. Només poden agafar una balisa per vegada.
    - Són capaços de pintar en el terra amb un pinzell de color blanc.
- Detecció:
    - Incorpora uns infrarojos virtuals que detecten murs, balises i altres robots al davant a una distància de uns quadrants determinats a més de retornar el tipus d'objecte.
    - Té uns sensors infrarojos de reflexió que detecten el terra blanc a davant seu.
    - Sensors que detecten si estan a sobre de un interruptor de terra.
    - GPS: Coordenada a on es troba
    - Compas: Direcció en la que està encarat el robot.
- Comunicació:
    - Són capaços de comunicar-se amb robots que estiguin a uns certs quadrants a la rodona. Envien i reben paquets de informació

Funcionalitats del processador:

- Es capaç de a partir d'instruccions ja compilades, fer les funcionalitats del robot descrites anteriorment.
- Té una memòria RAM que seran una sèrie de valors tipus float.
- Potser executat pas per pas per facilitar el que fa cada instrucció.
- Es poden detectar els errors de compilació

- És capaç de detenir la execució si s'accedeix a una posició de memòria errònia o un nombre és massa alt etc etc.

Com es va fer

Al ser un programa fonamentalment gràfic, necessitava conèixer com representar gràfics 2D en una plana web i a part de fer-ho en Flash, existia la possi-

bilitat molt més elegant i extensible de fer-ho amb l'objecte Canvas de HTML 5. Aquest objecte es manegava amb JavaScript que és el llenguatge per excel·lència de les planes web en el costat del client (browser) Tot això ho vaig especificar en la PAC 2 molt bé crec però faré menció en aquest apartat de nou de tota la arquitectura general de la aplicació començant pel registre de usuaris, la zona d'usuaris i l'entorn d'edició i explicant també la base de dades i de com es graven i recuperen dades d'aquesta i com aquestes dades arriben a veure's en la plana web. Tota aquesta mecànica no va ser trivial perquè requereix l'ús de diversos objectes tant a la banda del client com del servidor. Vaig necessitar veure molts exemples en internet.

Primer de tot vaig veure'm amb el repte de tornar a saber gestionar una base de dades utilitzant el Workbench de MySql que va ser el motor de base de dades que vaig decidir usar ja que molts servidors de planes web l'incorporen en els seus serveis. Però va ser un aprenentatge que va anar unit a l'aprenentatge de l'ús de la base de dades des de els arxius JSP o servlets. Una de les decisions que vaig fer és usar Hibernate per gestionar totes les insercions, consultes i esborrats etc de la BD. Si el projecte l'hagués fet per a mi i no l'hagués tingut que presentar ni res, òbviament no hagués usat aquest framework ja que m`ha portat més mal de caps del que estava previst. És un framework molt complexa que crec que s'ha d'especialitzar en ell i no es pot usar fàcilment. Si ho hagués fet amb connexions manuals a la BD i consultes SQL, operacions que ofereix el llenguatge Java en la programació de servlets, hagués sigut molt més ràpid i no me les hagués tingut amb la multitud d'errors que Hibernate em donava. Hibernate és una eina molt útil per algú que fa un projecte molt ampli en quan a tenir una base de dades molt complexa etc com pot ser un comerç virtual etc. Però en el meu cas no feia falta. De tota manera m'ha servit d'aprenentatge de les quatre pinzellades necessàries per fer servir aquesta eina.

No començaré a fer un tutorial d'Hibernate ni de bases de dades però sí el com vaig anar aprenent conforme les necessitats que requeria el projecte.

Primer vaig pensar en el que volia que fes el projecte, a més del registre d'usuaris etc que son coses més trivials, vaig pensar en com tractaria el pro-

blema de la compilació del codi virtual i de l'execució i representació gràfica dels robots.

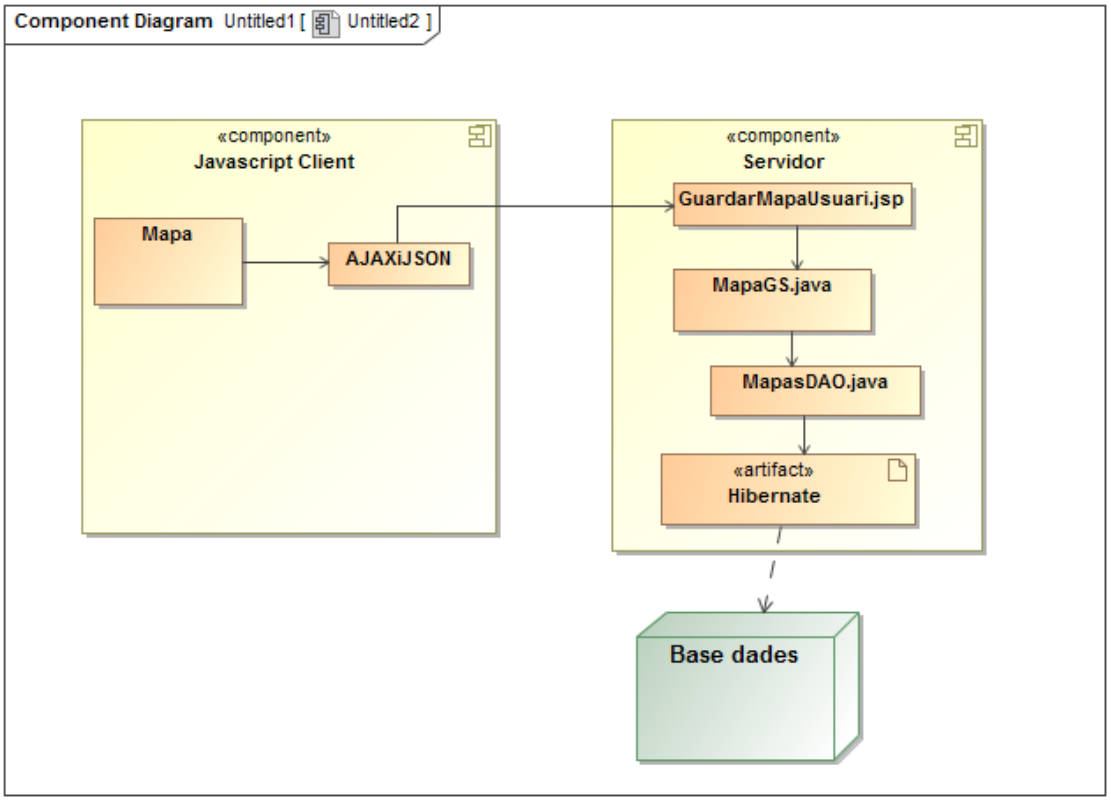
Tenia dos entitats, el servidor i el client, i es poden fer servir per implementar codi un o l'altra. El client fa servir HTML i javascript fonamentalment i el servidor Java. Al ser la compilació i execució del codi virtual, dos processos francament complexos, sobre tot el primer, vaig pensar que era millor programar-lo en Java mitjançant objectes de Java perquè m'ofereix molta més claredat al meu gust que no pas Javascript, que no és un llenguatge purament orientat a l'objecte. En principi volia fer servir Javascript només per la representació gràfica en el Canvas de la plana web i que el servidor atengués les peticions de compilar i executar el codi virtual dels robots. Compilar un codi és interpretar una cadena String i retornar un array de instruccions virtuals. Per tant es podia passar al servidor un String, que aquest el processes amb codi Java i l'enviés al client ja compilat. El procés d'execució del codi és diferent, es produeix a temps real i la execució de les instruccions seqüencial ha de produir una sensació de moviment del robot continu. Per tant s'havia de fer usant una comunicació TCP entre el servidor i l'usuari tal com fan els jocs a temps real i no pas un protocol HTTP de peticions /respostes. L'execució dels robots la manaria per tant el servidor i no el client. Aquesta opció era més complexa i no valia la pena. És útil quan es vol generar un món virtual en el que un avatar es mogui per un entorn i aquest moviment sigui capturat per un altra usuari a temps real. Però en el meu cas vaig pensar que si l'execució del codi virtual el fa el usuari en javascript, tot seria més senzill. Per la meva conte m'agradaria aprendre a fer servir comunicacions TCP i m'hi posaré una vegada entregui el projecte.

Vaig començar a crear una base de dades que tingués totes les taules i relacions que necessitava i el problema que se'm plantejava era que primer era millor saber quins objectes faria servir en el programa dels quals volgués persistència. En el cas del registre d'usuari tenia ben clar que necessitava una classe Persona del qual Usuari heretés propietats però això significava una relació d'herència que sempre complica una mica més tot. La vaig eliminar perquè no necessitava més subclasses que heretessin de Persona. Directament vaig usar la classe Usuari la qual tindria totes les propietats pertinents que expressessin el que volia que quedés gravat. La idea era que Usuari contingués

objectes Mapa que son els que representen els diferents entorns gràfics i que aquests a la vegada continguessin els Robots, elements mur, balises etc. El objecte Mapa i Robot serien els més complexes de programar i com veurem tindrien altres subobjectes que usen o contenen que no cal que tinguin persistència com per exemple Processador.

En javascript vaig començar a dissenyar els objectes que necessitaria per elaborar el programa i conforme veia el que necessitava persistència, ho anava implementant. El objecte Robot, té una propietat anomenada 'codi' que és de tipus String i conté tot el codi que hem dissenyat per aquell Robot en particular. Aquest codi com hem dit es compila al servidor i ens ve tornat en forma de un array d'instruccions interpretable per el processador virtual implementat en javascript. També se'ns retorna la memòria RAM virtual del robot en qüestió. Vaig pensar que seria interessant de persistir les instruccions i la RAM en la base de dades perquè era com més "real" ja que un processador, una vegada li carreguem instruccions i memòria, els conté per sempre fins que es posa informació nova etc. Pensava més aviat en l'ús extensible de la aplicació com ara poder fer un joc en el que hi haguessin robots pre programats. En aquest cas, voldríem que només entrar en una pantalla creada, s'executessin tots els codis de cop sense necessitat de compilar-los etc. Però vaig pensar que era més senzill no crear noves taules i ocupar memòria innecessari a quan es podien compilar tots els codis de cop i després executar-los de cop. Per això vaig incorporar la funcionalitat aquesta.

En la PAC 2 ja vaig explicar a fons el disseny de classes i arquitectura del projecte general, però diré quatre pinzellades de la problemàtica que va tenir passar els objectes creats amb javascript al servidor per posteriorment fer-los persistents en la base de dades mitjançant Hibernate.



Com veiem, en l'esquema que vaig fer es pot apreciar com un objecte Mapa que conté tots els subobjectes com Robots, Balises, Murs etc, ha de transferir-se al servidor sent aquest el que s'encarrega de gravar-lo a la base de dades.

El problema és que el objecte Mapa de javascript conté moltes propietats que no son necessàries transferir al servidor i que a més aquesta transferència es fa amb la llibreria JSON de javascript que vaig incloure i amb JQUERY. Recordem que vaig decidir enviar cadenes JSON entre Servidor i Client per ser molt més compactes i fàcils de gestionar tant en java com en javascript mitjançant les llibreries i frameworks que contenen. En la banda del client (javascript) vaig decidir de crear uns objectes paral·lels als objectes que volia transferir i persistir. Els objectes son MapaJSON, ObjecteMapaJSON i RobotJSON. Llavors vaig crear una classe en javascript que es diu GestorJSONiAJAX que s'encarrega de transformar els objectes Mapa, Balisa etc en objectes MapaJSON i ObjecteMapaJSON i RobotJSON. A l'hora de transferir al servidor, uso aquests objectes creant una cadena JSON amb el mètode de la llibreria JSON.stringify(MapaJSON). Aquest mètode simplement llegeix totes les propietats del objecte va creant la cadena JSON . Aquesta cadena es pasarà al

servidor mitjançant AJAX amb Jquery. Ja vaig posar com en la PAC 2. Llavors quan arriba al servidor la cadena, arriba a un servlet que en aquest cas s'anomena GuardarMapaUsuari.jsp que usa la llibreria GSON de Google per interpretar la cadena JSON i llegir les seves dades creant objectes Mapa en la banda del servidor que posteriorment Hibernate usarà per gravar les dades del Mapa a la base de dades.

Aquests objectes en la banda del servidor, son els Getters/Setters que el mateix Hibernate haurà creat a partir del disseny de la base de dades amb un automatisme que té. L'arxiu GuardaMapaUsuari.jsp a més ha de adjudicar al nou mapa creat, el usuari al qual pertany perquè es gravi bé en Hibernate. Aquest usuari l'obté de una variable de sessió que prèviament s'ha carregat al registrar-se dit usuari. Vaig fer una classe GestorBDMapas que el que fa és usar Hibernate per gravar els mapes d'usuari, fer consultes d'aquests mapes, obtenir llistats de mapes d'usuari, eliminar un mapa etc. En el cas de gravar s'usa el mètode guardarMapa2. Aquest mètode el que fa és primer esborrar tots els objectes del mapa, robots, balises tot de la base de dades. I després grava a aquesta els nous objectes del mapa. Ho vaig fer així perquè al gravar un mapa en concret, donava ordre de gravar tots els seus subobjectes i no gravo un objecte en particular un per un com si es tractés de un objecte Estudiant o Assignatura, aquí he de gravar cada vegada tot de subobjectes i per tant he de esborrar els existents relacionats amb el mapa pertinent. Aquí em vaig trobar amb un problema que vaig solvatar més tard. Al crear la taula en la base de dades que identificava als Robots per exemple, com a clau identificativa de cada element de la taula vaig crear un id auto incremental que el motor de la base de dades crea. Això comportava un problema, quan esborrava els objectes del mapa, al guardar-ne de nous, es creaven identificadors incrementals amb un nombre cada vegada més gran. Per eliminar aquest problema, la solució va ser usar com a clau identificadora, alguna mena de codi únic creat per mi i la solució era posar com a clau identificativa una combinació de tres valors: El codi del mapa al qual pertanyen, la posició X dins d'aquest i la Y. Hibernate el que fa a l'hora de l'automatització dels Getters/Setters i troba una combinació de camps com a clau identificadora, és crear per exemple en el cas de la taula Robot, una classe Robot i RobotId. RobotId contindrà els atributs idMapa, posicioX i posi-

cioY que identifiquen a cada Robot de manera única. Per tant, a l'hora de construir els objectes Robot a mans de la llibreria Gson que a la seva vegada llegeix les dades de la cadena JSON, aquesta llibreria ha de trobar els atributs amb el mateix nom dins la classe per copiar-los. Per tant, com que els tres atributs que identifiquen la classe Robot no estan en la mateixa classe sinó en la subclasse RobotId, vaig crear aquests atributs i a l'hora de guardar els robots, s'ha d'invocar per cada robot el mètode crearIdRobot que crea l'objecte RobotId. D'aquesta manera Hibernate el guardarà a la base de dades correctament. El mateix amb la classe ObjecteMapa. Quan en canvi, recupero un 'aquests objectes de les taules de la base de dades, Hibernate ja crea aquestes classes identificadores i el que faig és fer una còpia dels valors de la clau id en la classe Robot o Objectemapa i així és pot mitjançant la llibreria Gson, formar la cadena JSON per transferir-la al client.

Això em porta a explicar que el procés invers, el pas de informació del servidor al client, usa com he dit també la classe Gson, i s'envia amb un out.print al client.

El client rep la cadena JSON del servidor i la transforma en objectes de javascript mitjançant el mètode de Jquery, parseJSON. Això també ho gestiona la classe GestorJSONiAJAX en javascript.

En un altre ordre de coses parlaré una mica de com vaig dissenyar els efectes gràfics que fan que es vegi el robot movent-se, agafant les balises, detectant els murs etc:

En la PAC 2 ja vaig explicar com funcionava el canvas i com podia moure objectes però el que intentaré explicar son les reflexions que vaig tenir a l'hora de fer-ho i com ho vaig fer finalment. Tenia clar que el dibuix tenia que tenir una vista zenital de l'entorn perquè era la més simple i la que representava millor les coordenades cartesianes. Més tard he descobert que és possible fer una vista isomètrica en Canvas però es força complicat. També em vaig informar de que és possible dibuixar en un context en tres dimensions però pel programa que volia fer és més que suficient la vista 2D.

En la PAC vaig dir que el dibuix s'ha de fer per capes. L'objecte Canvas com vaig dir, té un subobjecte Context en el que es dibuixen tot el que volem. Especialment important és el mètode drawImage perquè funciona com si poséssim transparències una a sobre de l'altra. La primera transparència contindria el fons del dibuix, en aquest cas el terra de la vista zenital. La transparència de sobre contindria els murs o objectes del mapa inamovibles i la tercera els mòbils com son balises o Robots. Però per fer l'efecte de moviment dels objectes movibles tenia que cridar reiteradament en el temps un mètode que redibuixes totes les capes. Això ho fa el mètode setInterval de Window . El problema que jo li veig a aquesta manera de fer-ho és que només puc programar un interval i si jo volgués per exemple programar un objecte que modifiqui el seu estat en un valor de milisegons transcorreguts diferent, no puc programar un altra esdeveniment amb un interval diferent. Hi ha altres maneres que sí ho permeten però no les he estudiat. Un altra problema és que setInterval no espera a que les instruccions que s'executen en un pols del rellotge s'hagin executat del tot. Això produeix que si l'interval és petit, es puguin produir col.lapsaments del flux d'execució. Jo vaig triar en un inici un interval de 20 milisegons. Cada 20 milisegons es redibuixen les capes gràfiques i s'actualitzen les posicions dels Robots i s'executa la següent instrucció de tots els Robots que estan actius. Per tant la rapides del flux del programa de un robot en particular no és molt ràpida i no m'acaba de agradar l'efecte que dona. Per exemple, vaig programar un Robot que podia anar de un punt X,Y a un altra fent càlculs trigonomètrics i d'aquesta manera podia triar en cada cicle si anar endavant, dreta o esquerra. Però anava molt lent amb l'interval, el únic que es pot fer és escorçar aquest interval en temps d'execució del programa. Aquesta era una de les raons del perquè hagués estat bé que el servidor s'hagués encarregat del processament de les instruccions i hagués transmès al client la posició X,Y de cada Robot. El client hagués seguit un mer representador gràfic que amb l'interval de 20 milisegons en té prou per donar sensació de moviment cinematogràfic. En java es podria haver usat els Threads que permeten crear fils d'execució independents que el processador de l'ordinador gestiona perfectament.

Aquest defecte dóna com a resultat que si posem molts robots fent tasques anirà molt lent. Aquest problema té que ser solventable, però no m'ha donat temps ha informar-me.

Centrant-nos en l'objecte Robot, aquest té tots els mètodes perquè aquest es mogui per l'entorn a més de totes les propietats necessàries per fer-ho. També conté les restriccions de moviment com la detecció de col·lisions amb l'entorn que fonamentalment son: col·lisions amb elements mur, altres robots i balises. Els robots son els únics elements que es mouen de manera activa per la pantalla i els moviments que tenen son la de anar endavant i girar a la dreta o a la esquerra. Podria haver programat el moviment de anar enrere però vaig pensar que podria ser programable simplement donant la volta. Recordo que els robots es mouen per els quadrants com si un joc d'escacs es tractes. Això simplifica molt la programació i la celeritat de l'execució de les instruccions virtuals. En quant al tractament dels sprites del robot i el canvi d'sprite en un moment donat vaig elaborar una classe que gestiona com agafar cada imatge del sprite de un mapa d'imatges concatenades i conté el mètode next que dona pas al pròxim sprite de manera circular. Per dissenyar el mapa de sprites concatenats vaig usar el programa Piskel. Un programa online molt senzill que genera sprites. El que vaig fer és dissenyar una sèrie d'imatges pel que fa al cos del robot en les que s'aprecia el moviment de una cadena com de tanqueta que va cap endavant. Per un altra part vaig fer un mapa pel moviment del cap cap a dreta o esquerra per fer que el cap del robot anés independent del cos per donar un efecte quan el robot detecta a la dreta, esquerra o endavant quelcom. Un altra moviment gràfic que vaig programar és quan el robot agafa una balisa. L'objecte Balisa té una sèrie d'sprites en els que es veu una pinça agafant-la. De manera que quan un robot invoca el mètode de agafar la balisa comença l'animació i quan el robot deixa la balisa l'animació es torna a produir però a la inversa. Tot això sembla senzill però em va costar, no sé si es perquè de vegades et lleves més obtús que altres dies.

La classe Robot de javascript, conté una instància de la classe Processador. Aquesta classe té dos propietats, un array d'instruccions i un de la memòria RAM virtual i un mètode que executa instrucció per instrucció, les instruccions s'executen quan el temporitzador creat per setInterval crida la fun-

ció de actualitzar però el programa dóna la opció de fer una pausa i es pot executar instrucció per instrucció manualment. Això va molt bé per saber com s'està executant un programa i com veurem a continuació es pot veure visualment com la memòria RAM va fent escriptures i lectures i com es van succeint les instruccions.

Per fer això vaig dissenyar una classe que gestiona la visualització de l'estat del processador i del Robot. La classe Robot té una instància d'aquesta classe i li passa en un moment donat un valor determinat i és aquesta classe la que s'encarrega de representar-lo en el html de la web.

Per exemple, els valors de la RAM es veurem en una taula de una manera global i cada operació que es fa d'escriptura o lectura es veurà reflectit, posant-se de color verd el fons de la cel.la si és una lectura o vermell si és una escriptura. També es mostra informació d'interès en una taula com son el nombre de col·lisions que té un robot, d'iteracions, nombre de quadrants recorreguts o de girs efectuats. Amb aquestes estadístiques pretenc que l'usuari pugui saber si el codi programat per un robot conté errors perquè col.lisiona o bé pugui veure si hi ha masses iteracions i pugui fer un algoritme que funcioni amb menys iteracions etc.

Aquesta manera de mostrar dades no estava explicada en la PAC 2 i he cregut oportú explicar-la.

Una de les parts més complexa de programar va ser el compilador que com he dit està programat en Java i s'executa a la banda del servidor.

El compilador com he dit rep un String i el converteix en un seguit d'instruccions interpretables pel processador fet en javascript que estaran dins d'un array d'objectes Instrucció. També entrega un array de valors declarats que és la memòria RAM. El que fa el compilador és primerament dividir el String d'entrada en una sèrie de Strings que contindrà un array. Darrerament es procedirà a fer una passada per tot el array a la cerca de variables declarades i construir un array que relaciona el nom de les variables amb la posició dins la memòria RAM que paral·lelament, es va creant. Hi ha 10 variables amb el nom R0 a R9 que son creades per defecte. Aquestes variables son molt útils a l'hora

de escriure el programa perquè escurça la feina ja que no s'han d'anar creant variables. També vaig pensar que era didàctic perquè fa que tingui semblança la programació amb aquest llenguatge inventat, a la programació amb assemblador que fa servir registres per poder executar càlculs mes ràpids sense necessitat de accedir a la memòria RAM sempre. La primera passada per l'array d'strings també serveix per extreure les etiquetes que apunten a una posició dins del programa creant un array que correlaciona el nom de les etiquetes amb la posició en l'array. Dins les funcions es podran declarar variables que poden tenir el mateix nom que el programa principal o que les incloses en altres funcions (abast de la funció) Això es fa en aquesta primera passada del array d'strings posant en el array d'etiquetes/posició RAM, un nou valor que descriu en quina funció es troben declarades.

Després es fa una segona passada en la que ja estem disposats a interpretar cadascuna de les ordres i anar creant objectes Instrucció. En aquesta etapa es va llegint cada String (ordre escrita) i es veu si conté un patró que representi un accés a la memòria RAM com pot ser el nom d'una variable o el nom d'una variable seguit de un altra entre claudàtors o simplement un nombre. Tot això fa que es transformin els noms de variables que son referències a posicions de la RAM, en nombres que indiquen a on es troben els valors. Es van creant d'aquesta manera els objectes instruccions. En resum, en aquesta etapa el compilador construeix tot l'array d'objectes Instrucció amb els seus operands pertinents. La gestió dels errors que hom pot fer no es molt específica però descriu quina instrucció ha donat problemes que es retorna al Usuari per informar-lo. Això seria molt millorable sobre tot perquè seria útil indicar en quina posició del codi ha donat error cosa que no es fa i en una versió millorada es podria incloure aquesta funcionalitat.

Canviant de tema, parlaré a continuació de la manera com he editat l'HTML de les diferents pàgines i com s'intercanvien dades unes amb les altres etc.

Bàsicament hi ha tres planes web, La plana de registre/logging que és a on un usuari es registre o bé si ja ho ha estat s'identifica, la plana de Zona d'usuari

que és on l'usuari gestiona els escenaris que ha creat o d'on pot començar a crear un nou escenari(mapa) i finalment està la plana d'edició de l'entorn dels robots que engloba la edició del codi La plana de Registre/Logging està programada exclusivament en Java i Html. Les entrades de formulari que es produeixen es gestionen a la mateixa plana quan es polsa el botó que és de tipus Submit. Quan un usuari s'identifica o es registra de nou, el botó submit envia la petició a la mateixa plana i aquesta després redirecciona amb el mètode 'sendRedirect' de l'objecte 'response' de la plana JSP a la plana 'ZonaUsuari.jsp'.

En la creació dels estils de les planes web, he usat uns arxius css que contenen els estils però moltes vegades, i sobre tot al editor, degut a la complexió he creat els estils en cada tag div o input etc.

En la plana de zona d'usuari ja apareix javascript . Apareixen els mapes del usuari en un quadre de llista seleccionable. Per fer això faig servir el mètode de JQuery 'replaceWith' que permet bescanviar l'html de un control com ara una llista o un div el que sigui. El nou html conté tot el codi per generar la llista amb el nom dels mapes i faig això amb javascript fent primer una consulta a la base de dades mitjançant jquery i la tecnologia ajax i la consulta em retorna la llista de mapes en un JSON juntament amb l'id de cada mapa. Llavors itero en la llista de mapes i vaig creant els diferents tags 'option' que contindrà el tag 'select' i com he dit el bescanvio per l'html que ja tenia el 'select'. També faig que cada vegada que es selecciona un mapa de la llista aparegui un croquis dibuixat usant canvas del mapa en qüestió, aquest mapa també el podria haver fet creant una còpia de la imatge del mapa creat en l'editor i guardar-la en un arxiu png o jpg. Després caldria afegir-li la ubicació dins del servidor en un camp de la taula Mapa de la base de dades però a més de més complexa, aquesta manera és ràpida i molt més senzilla. Per crear el botó que obre el mapa i que quan es polsi obri el mapa seleccionat en la llista, vaig programar primerament un esdeveniment en Jquery per la pulsació de botó en el que s'usava el mètode open de l'objecte Window de javascript. El mètode open feia una petició GET a la plana 'EntornEditor.jsp' mitjançant la instrucció : window.open('EntornEditor.jsp?idMapa=45') Veiem que llavors en la plana EntornEditor.jsp té que haver programat en java una ordre per recuperar l'idMapa coma variable GET però aquesta manera de fer-ho no està bé perquè llavors

qualsevol usuari si posa el nom de la plana en un brower seguit de un id pot accedir al mapa directament. Per tant hi havia dos maneres de fer-ho, o bé pel mètode POST que oculta les variables de la petició o bé mitjançant variables de sessió. Pel id del usuari tant en la Zona d'usuari com en la zona de edició de mapes s'usa variables de sessió però en el cas del id del mapa vaig voler usar el mètode POST. Per programar-ho vaig crear en javascript el codi html que contindria el formulari i el botó submit que obre el mapa juntament amb un tag de tipus 'hidden' amb el valor del id del mapa. Després novament amb el mètode replaceWith de Jquery, incrustava el codi nou al formulari del botó. Per tant en el projecte hi ha múltiples maneres de passar variables al servidor. Les variables de sessió però son de les més efectives perquè es guarden durant tota la sessió de l'usuari i és extremadament útil quan hi ha multitud de planes. Tot això s'entén i s'explica millor, incrustant codi però en la presentació ho explicaré de manera més visible.

Finalment tenim la plana de Edició dels robots i l'entorn. La complexió d'aquesta es molt més gran i em va costar veure la millor divisió de les diferents parts que havia de contenir. Al final vaig pensar en fer-ho com el 3DStudio o altres programes en el que es treballa amb un editor gràfic i ha de tenir moltes opcions per tal de fer-lo servir. Aquests programes fan servir una divisió fonamentalment en tres parts. Una part superior on hi ha el menú d'opcions i certes eines de dibuix o de visió del dibuix, una part més gran on realment es veia l'entorn 3D i que responia a esdeveniments del ratolí i un altra a una banda, o esquerra o dreta on es podien manipular els objectes 3D, es veien les seves propietats etc etc. També podia haver una part inferior.

Aquesta part lateral solia ser scrollable de manera que podia contenir molta informació i ràpidament amb el mouse fer un scroll i accedir a tota. Per tant vaig posar l'editor de codi dels robots, el control manual dels robots, la visualització de les dades com la RAM etc tot a un cantó de la pantalla i amb un div scrollable s'accedeix a tot. La barra d'eines de dibuix del entorn en la part superior. La millor forma de visualització de la plana és amb resolucions de 1360 per 768 pixels perquè així es poden veure totes les parts del editor. Totes les accions del mouse sobre la barra d'eines o sobre l'editor de codi etc està programat en diferents arxius javascript inclosos en la plana d'edició.

Conclusions i reflexions:

Amb tot el dit ja he fet un repàs bastant exhaustiu del que és el projecte, el que fa i el com ho fa, i també problemes que he tingut al fer-lo i com els he solvatat.

Mai havia fet un projecte web de tanta envergadura i m'ha resultat força complicat i llarg. Soc conscient de que no ha quedat del tot polit i que no funciona a la perfecció però no puc disposar de més temps per millorar-lo. Al principi del curs tenia més temps però cap el final, per causes familiars i laborals no he pogut acabar-lo com m'hagués agradat. M'he adonat que soc molt impetuós i que de vegades no vigilo els detalls que després passen factura. En un projecte complex com aquest s'ha de anar lent en fer cada funcionalitat i preveure totes les possibles accions del usuari. M'hagués agradat deixar un producte comerciable i sense bugs, però he anat mancat de temps.

Malgrat això crec haver demostrat els meus coneixements informàtics i que soc capaç de aprendre llenguatges nous, frameworks etc.

Crec que aquest projecte pot ser molt útil si es poleix i m'he quedat força satisfet per les coses que és capaç de fer i portar-ho a la pràctica. Com vaig dir a la PAC 1, sempre m'ha agradat acostar el mon de la programació a la gent profana i crec haver-ho aconseguit. A més es pot profunditzar més o menys en les seves funcionalitats de manera que si el fan servir nens d'un institut per exemple, es pot usar la eina de dibuix que incorpora amb les instruccions painton/paintoff i anar incorporant més instruccions. La última funcionalitat incorporada és la de comunicació entre robots que no està provada encara ni polida. En ella mitjançant les ordres send/receive em puc comunicar amb robots que estiguin en uns certs quadrants a la rodona i enviar-los paquets de dades. Incorporant aquesta funcionalitat puc estudiar sistemes de col·laboració entre robots o sistemes biològics.

Uns exemples o exercicis concrets a fer serien:

- Un programa que dibuixi un quadrant usant funcions com es pugui
- Un robot que reculli unes balises no cercant-les sinó que vagi directament a trobar-les i dipositar-les en interruptors. Un segon robot que quan estiguin posades totes les balises es mogui en cercles indefinidament.
- Un robot seguidor de línies. Un ha de dibuixar-les i un segon robot la segueix.
- Un programa que navegui per una habitació i no colis ioni amb cap objecte
- Un programa que cerqui balises en un planell i cerqui interruptors de terra per dipositar-les.
- Un programa que donat un punt X, Y vagi cap a ell en línia recta usant el GPS, el compàs i funcions trigonomètriques.
- Dos robots, un d'ells cerca balises i quan les troba va cap un segon robot, li comunica la posició de les balises i vagi directe a recollir-les
- Dos robots, un d'ells cerca la sortida de un laberint o plànol i a continuació dibuixa una línia fins a un altra robot que aquest ha de recórrer. (Simularia com les formigues exploradores deixen un rastre de feromones que és seguible per una formiga obrera)
- 3 o 4 robots que persegueixin un robot determinat programat per fugir. Es pot estudiar la IA i veure com es pot programar arbres de decisions per tal de cooperar en grup en un atac.
- Una bandada de robots que es mogui unificadament com ocells en l'aire. Un d'ells pot portar la iniciativa o bé repartir la decisió entre el grup.

BIBLIOGRAFIA DEL PROJECTE

Introduccion a las aplicacions Web con Java de Marcario Polo y Daniel Villafranca

Introducción a servlets con NetBeans

Tutorial básico de JavaEE por Abraham Otero

Using Hibernate in web application

Ejemplo de aplicación web con Hibernate que funciona usando Netbeans IDE y JSTL (JSP(X))

Comunicar JSP y JQuery

Cómo dibuixar con el ratón en un Canvas HTML5

Usar JSON para enviar datos entre el servidor y el cliente

Introducción a Javascript orientado a objetos

Forum stackoverflow: How do you create a toggle button?

Tutorial básico de Hibernate

Ejemplos de JSON y Java con Gson

Ejemplos de expresiones regulares en Java

HTML5 Canvas

HTML5 Canvas drawImage() method

Animaciones 2D utilizando el Canvas de HTML5

Piskel (Creació de Sprites)

TAG Select

Tablas básicas

Ajax con JSP y servlets

Apunts de Estructura y tecnologia de computadors.

Coneixements previs de controladora Arduino