Universitat
Autònoma
de Barcelona

DESENVOLUPAMENT D'UNA PLATAFORMA DE SIMULACIÓ DE RUTES CICLISTES (SIMULBIKE)

Memòria del Projecte Fi de Carrera

d'Enginyeria en Informàtica

realitzat per

Alan Soldado Campos

i dirigit per

Jordi Carrabina Bordoll

Bellaterra, 01 de Febrer de 2011

Universitat
Autònoma
de Barcelona

Al a baix firmant, Jordi Carrabina Bordoll

Professor de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a qui correspon la present memòria ha estat realitzat sota la meva direcció per Alan Soldado Campos.

I perquè així consti firmo la present.

Firmat: Jordi Carrabina Bordoll

Bellaterra, 01 de Febrer de 2011

Agraïments

Un cop s'ha finalitzat el projecte, és el moment d'agrair a totes aquelles persones que, conscientment o no, han estat al meu costat, donant-me els ànims que tanta falta m'han fet.

Arribat a aquest punt, recordo els ànims que m'han donat els companys de la Universitat, ajudant-me amb la seva experiència personal i oferint sempre els seus consells. Tampoc em puc oblidar de la meva família que, amb la seva insistència, m'han empès perquè miri sempre endavant i no em rendís al llarg de tota la carrera.

Finalment, vull dedicar un reconeixement al meu tutor del projecte, Jordi Carrabina, el qual m'ha acompanyat en un tram del camí, representat en el present projecte, donant els seus consells i recomanacions.

Entre tots heu fet possible aquesta realitat. Moltes gràcies a tots.

Índex

1. Introducció

En aquest apartat principalment tractaré sobre quines han estat les motivacions per tal de portar a terme el present projecte, així com els diferents objectius que m 'he marcat en la elaboració del mateix.

1.1. Origen del projecte

El projecte que es pretén realitzar sorgeix de l'estudi de les necessitats d'un sector de la població que practica el ciclisme de muntanya, que es tracta d'un esport que actualment es troba a l'alça. Aquests aficionats dediquen molts recursos a la pràctica del seu esport preferit que gira entorn un element: la bicicleta. Després d'analitzar els diferents aspectes que es podrien millorar dins el sector mitjançant el desenvolupament i implementació de solucions TIC[1], finalment va sorgir la idea d'un simulador, que servís com a plataforma d'oci o d'entrenament, que permetés emular rutes de muntanya usant la pròpia bicicleta dels ciclistes.

La idea inicialment plantejada consisteix en que els usuaris de la plataforma utilitzin la seva pròpia bicicleta per interactuar amb el simulador. Concretament, el funcionament de l'aplicació consisteix en instal·lar la bicicleta sobre un rodet que permetrà transmetre el moviment del pedaleig a l'aplicació que reproduirà la ruta d'acord a la velocitat de pedaleig.

Un altre aspecte que es pretén desenvolupar en la plataforma consisteix en la possibilitat d'usar-la en mode en línia, és a dir, a través d'Internet. La idea és que es creï una xarxa social entorn de la plataforma i que els ciclistes puguin interactuar entre ells: realitzar rutes conjuntament, realitzar competicions, ...

1.2. Motivació

El principal motiu per el qual m'he decantat per la realització d'aquest projecte és la gran varietat d'àmbits que toca la seva implementació. A llarg del desenvolupament, apareixen conceptes com: programació a baix nivell, programar un reproductor multimèdia, selecció d'un protocol de comunicació o programació en temps real. A més, hi ha d'altres aspectes com ara base de dades, tractament de dades (XML) o conceptes de geoposicionament (GIS). Per tant, com a programador, m'ofereix un gran repte.

D'altra banda, un projecte final de carrera d'una Enginyeria en Informàtica ha d'implicar quelcom més que una simple tasca de programació. Darrera el projecte presentat hi ha la concepció global del projecte així com les tasques relacionades amb la enginyeria del software, és a dir, l'anàlisi i disseny de l'aplicació. Aquestes també recaurien sobre mi.

Si unim els dos aspectes principals comentats prèviament, es pot veure que el desenvolupament d'aquesta plataforma em presenta la oportunitat de realitzar un projecte real al llarg de totes les fases. A més, com que es tracta d'un projecte d'empresa, em dóna la oportunitat de realitzar-ho en el món real, adquirint així

[1]. TIC: Tecnologies de la Informació i la Comunicació.

una visió real de l'entorn en el que, com a enginyer, em mouré en el dia de demà.

1.3. Objectiu general

L'aplicació que es pretén realitzar en aquest projecte consisteix en desenvolupar un sistema de simulació de rutes de muntanya en bicicleta. Com que el desenvolupament de la plataforma completa requereix un període de temps molt ampli i un equip de desenvolupadors, en el present projecte final de carrera em limitaré el desenvolupament d'un prototip del sistema. A mode de resum, destacar que el prototip inclourà els següents aspectes:

- Disseny i implementació del codi que requereix el hardware.
- Disseny i implementació d'un algorisme per dur a terme la simulació.
- Disseny i implementació del reproductor per visualitzar la ruta.

Una de les claus del projecte és la seva especialització per l'exercici dels aficionats al ciclisme de muntanya. Per tant, com a criteris d'èxit s'avalua la major aproximació a la realitat de l'esport, és a dir, que els ciclistes tinguin una sensació de realisme al realitzar la ruta. La clau de l'èxit del projecte es troba en tres factors:

- L'usuari utilitzarà la seva pròpia bicicleta. Els aficionats al citat esport son usuaris que inverteixen molts recursos en la seva bicicleta; per tant, si la plataforma els permet usar-la, serà un atractiu afegit a la hora de l'adquisició del producte.
- Les rutes seran vídeos que reproduiran rutes reals, per la qual cosa, s'aconseguirà una visió realista de la ruta (aquest aspecte representa una novetat respecte a les aplicacions que existeixen en el mercat).
- S'aplicarà una resistència a les rodes de la bicicleta per simular l'esforç que cal aplicar en les pendents (reduir la resistència en les baixades i augmentar la resistència en les pujades).

1.4. Organització de la memòria

La present memòria del projecte s'organitza de la següent manera:

- **Capítol 2:** En aquest capítol es realitzarà un estudi de la viabilitat del projecte, indicant aspectes com ara l'estat de l'art, les novetats que aporta la plataforma que es pretén desenvolupar i la planificació de les diferents fases de desenvolupament.
- **Capítol 3:** Anàlisi de requeriments. En aquest capítol es descriuen quines seran les funcionalitats que s'espera que implementi l'aplicació, aquells requeriments que els usuaris finals de la plataforma esperen obtenir del producte.
- **Capítol 4:** Recerca tecnològica i disseny. En aquest capítol es descriuran les tecnologies que s'han analitzat i les tecnologies finalment aplicades per el desenvolupament, indicant els motius de la seva acceptació o del seu refús. A més, es definirà com el prototip realitzarà les diferents tasques i

com estarà implementat.

- **Capítol 5:** Resultats. Aquest capítol està destinat a oferir els resultats que s'obtenen a partir del prototip desenvolupat prèviament.
- **Capítol 6:** Conclusions. Finalment, en aquest darrer capítol es presentaran les conclusions obtingudes en el desenvolupament de la plataforma, la valoració personal de la realització del projecte i el treball futur que derivi a un producte final.
- **Annex:** En la memòria també figura un annex que conté la documentació generada al llarg del desenvolupament del projecte.

2. Estudi de viabilitat

En aquest apartat s'afegirà l'informe previ que s'ha realitzat sobre el projecte. En concret es parlarà sobre els objectius principals que es persegueixen. També s'explicarà l'estat de l'art i s'aportarà un estudi de la viabilitat del projecte i la planificació temporal que es pretén seguir per dur a bon terme el desenvolupament.

2.1. Estat de l'art

Actualment, en el mercat, es poden trobar diferents solucions que simulen l'activitat del ciclisme mitjançant aplicacions que aconsegueixen un menor o major grau de realisme. Alguns exemples d'aquests productes son:

- **Fitness Solutions (http://www.fitnesssolutions.com.gt):** aquesta empresa ofereix un complet simulador de ciclisme de ruta, és a dir, de carretera. A més ofereixen altres característiques com reproducció de rutes "reals" en 3D, possibilitat de crear noves rutes, possibilitat de realitzar les rutes acompanyat,... Tot i que moltes de les funcionalitats es pretenen afegir en el nostre prototip, hi ha una diferència important: la modalitat, nosaltres volem simular el ciclisme de muntanya, on les pendents tenen un paper molt important. Una altre diferència és que les nostres rutes son reals.
- **Honda Bicycle Simulator:** es tracta d'un simulador que està orientat a educar al ciclistes Japonesos en el codi de circulació. Disposa d'un seguit de recorreguts urbans preestablerts que els usuaris realitzen per adquirir pràctica en l'ús de la bicicleta en entorns urbans. Es caracteritza per posar especial atenció en el compliment del codi de circulació i disposa de sensors en el terra per detectar si l'usuari s'aguanta amb els peus. Aquest producte es troba força allunyat dels propòsits del prototip que volem desenvolupar.
- **TACX (http://www.tacx.com):** l'empresa TACX ofereix una plataforma d'entrenament per a ciclistes de carretera. El seu producte es basa en l'ús d'un rodet, per tant, el ciclista utilitza la seva pròpia bicicleta. Les rutes estan definides prèviament i s'adquireixen per separat. Es tracta de vídeos reals dels recorreguts a realitzar. Novament, la principal diferència amb el nostra producte es troba en la modalitat de ciclisme i la opció de multi corredors.
- **I-Magic (http://simulation.mirage3d.nl):** es tracta d'un simulador de ciclisme que utilitza el Hardware de la empresa TACX. El seu funcionament és molt similar a un videojoc: els recorreguts son dissenys en 3D. Permet al usuaris seleccionar terrenys, curses, oponents, carreres individuals o col·lectives, ... A més, controla la resistència aplicada al pneumàtic segons característiques com el pes, la velocitat, la pendent i la superfície. Aquest projecte s'allunya molt del nostre prototip en el realisme de la visualització, però s'apropa molt en el realisme del tacte amb la bicicleta.
- **Cyberbike de WII:** es tracta d'un projecte en el que treballa Nintendo per oferir un nou accessori a

la seva popular consola WII. Es tracta d'una bicicleta estàtica que és usada per jugar a un vídeo joc de curses en bicicleta. Les principals diferències respecte el nostre projecte es troben en que no s'utilitza la pròpia bicicleta de l'usuari i en que Nintendo simplement busca la creació d'un videojoc que no té res a veure amb la plataforma d'entrenament que es pretén desenvolupar en el present projecte.

Les característiques que es pretenen incloure en el prototip que es desenvoluparà son: un simulador de ciclisme de muntanya, reproduint rutes reals usant la pròpia bicicleta de l'usuari. Com s'ha vist, alguns dels projectes esmentats prèviament satisfan algunes de les característiques, però en cap cas es satisfan totes les condicions en un únic producte. Per tant, la principal novetat es troba en la integració de totes elles en un únic producte. A més, en el producte final (no inclòs en el prototip), es pretén potenciar el caràcter social de la plataforma.

2.2. Estudi de la viabilitat del projecte

L'estratègia a seguir en el projecte és la de crear dos components principals:

- **Hardware i la unitat de control:** aquest component és el que utilitza l'usuari per interactuar amb la plataforma. Es composa de diferents parts: el rodet (sobre el que s'instal·la el pneumàtic de la bicicleta), la unitat de control (responsables de la comunicació entre el rodet , l'usuari i el software local) i les comunicacions (es realitza una comunicació del hardware amb el PC mitjançant un port COM sèrie virtual).

Imatge 1: Exemple de rodet

Imatge 2: Exemple d'unitat de control

- **Reproductor local:** es tracta del software principal de l'aplicació. La seva funció principal és oferir una interfície a l'usuari per interactuar amb l'aplicació. La interacció es realitza a través de la unitat de control que dóna les ordres, indicades per l'usuari, o intercanvia dades com la velocitat de pedaleig o la resistència que s'ha d'aplicar al rodet. Finalment, s'ha de reproduir la ruta tenint en compte la velocitat de pedaleig i anar ajustant la resistència del rodet segons les pendents del tram de ruta que s'està realitzant.

El funcionament de l'aplicació consisteix en:

L'Usuari instal·la el rodet davant la pantalla. A continuació ha de col·locar la seva bicicleta sobre el rodet. Finalment, connecta el rodet a la unitat de control i a una font d'alimentació. El darrer pas per completar la instal·lació és connectar la unitat de control a l'ordinador que ha d'executar l'aplicació mitjançant un port USB. Finalment, ja es pot executar el software.

Imatge 3: Esquema de la instal·lació de la plataforma

Un cop s'ha iniciat l'aplicació, l'usuari podrà accedir a la seva sessió, de forma que la plataforma es personalitza segons l'usuari. Dins la sessió, l'usuari podrà controlar les seves dades personals, les estadístiques, els resultats, ... A més, podrà iniciar rutes, continuar rutes, descarregar-se noves rutes, ... Tot i això, la tasca principal és la reproducció de la ruta; aquest procés consisteix en:

La realització de la ruta consisteix en la reproducció del vídeo (conté la ruta filmada) ajustant la velocitat al pedaleig de l'usuari. De forma que com més ràpid pedalegi l'usuari, més ràpid avança la reproducció de la ruta. L'aplicació controla la distància recorreguda, la velocitat, ... per tal d'ajustar-se al màxim a la realitat. Un aspecte molt important és que, juntament amb la ruta, tenim l'alçada del punt on ens trobem[2]; d'aquesta forma podem saber si l'usuari està en un pla, en una pujada o en una baixada i la pendent de la mateixa; així podem ajustar si el rodet aplica més o menys resistència al pneumàtic de la bicicleta.

Les rutes es composen de diferents components:

- El vídeo on es mostra la ruta (recorregut filmat a una velocitat constant).
- Mapa d'alçades: mentre es filma la ruta, es van captant mostres periòdiques de l'alçada en que ens trobem, de forma que hi hagi una relació entre instants de temps i alçades. Amb aquesta informació es podrà determinar si estem en una pujada o una baixada.

[2]. La ruta està geoposicionada ja que en el moment de la seva filmació, es va registrant les coordenades (latitud, longitud i alçada) a intervals regulars.

El projecte és viable ja que, després de traçar les línies generals del desenvolupament, s'ha obtingut una visió de com s'haurien de realitzar les diferents tasques. El projecte consisteix en el desenvolupament d'un prototip, així que tot i els objectius plantejats, no es pot garantir l'èxit final a causa d'alguns interrogants com el correcte ajustament de la velocitat de reproducció o la correcte interpretació dels valors del rodet. Tot i els dubtes esmentats, es decideix procedir al desenvolupament d'un prototip.

2.3. Planificació temporal del treball

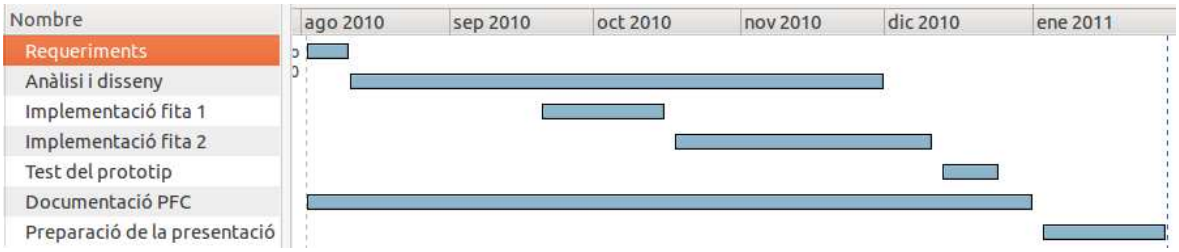
El desenvolupament de la plataforma implica donar resposta a una sèrie de reptes per tal d'aconseguir una aplicació final que satisfaci totes les necessitats funcionals que s'esperen en el simulador. Per tal d'assolir aquest èxit final, es marquen una sèrie de tasques i fites al llarg del desenvolupament que permetran una realització escalonada de l'aplicació:

- **Requeriments:** en aquesta fase realitzaré reunions amb el client per tal d'obtenir els requeriments funcionals que espera de la plataforma que es pretén desenvolupar. Els requeriments parlaran de la plataforma completa, però cal destacar que el present projecte només implica la creació d'un prototip. La resta de funcionalitats queden com a treball de futur si es demostra que el sistema és viable.
- **Anàlisi i disseny:** en aquesta fase realitzaré un estudi dels diferents reptes que requereix la implementació del prototip. A continuació faré una cerca de les diferents tecnologies que poden donar una resposta als reptes plantejats. Finalment triaré les tecnologies més adequades. Per altra banda, dissenyaré els algorismes i especificaré els components que s'hauran de programar.
- **Fita 1:** en aquesta fase espero obtenir una solució que permeti la obtenció de la velocitat de pedaleig de la bicicleta. El desenvolupament d'aquesta fase implica la programació a baix nivell del microcontrolador que rep la senyal del rodet i la comunica a l'ordinador. El criteri d'èxit que cal tenir present és la obtenció correcte de les dades i la fluïdesa de les comunicacions.
- **Fita 2:** en aquesta fase espero obtenir una solució que permeti la reproducció d'una ruta (vídeo) segons la velocitat de pedaleig del ciclista. El desenvolupament d'aquesta fase implica la programació d'un reproductor multimèdia amb el control de velocitat. El criteri d'èxit que cal tenir en compte és que la resposta ha de ser pràcticament immediata; per la qual cosa, s'han d'optimitzar al màxim les operacions per evitar retards.
- **Test del prototip:** en aquesta fase realitzaré una sèrie de proves sobre la plataforma per tal d'obtenir els resultats amb el simuladors. A continuació, realitzaré una comparació entre els obtinguts a nivell teòric i els obtinguts amb el simulador.
- **Documentació PFC:** en aquesta tasca englobo les tasques relacionades amb la redacció de la present memòria.
- **Preparació de la presentació:** en aquesta tasca englobo les tasques que fan referència a la presentació, és a dir: les transparències, preparació del discurs, ...

En la següent taula, es pot observar les dades que defineixen les diferents tasques a realitzar:

Fase	Nom	Comença	Acaba	Feina
1	Requeriments	02/08/10	10/08/10	7d
2	Anàlisi i disseny	11/08/10	30/11/10	80d
3	Implementació fita 1	20/09/10	15/10/10	20d
4	Implementació fita 2	18/10/10	10/12/10	40d
5	Test del prototip	13/12/10	24/12/10	10d
6	Documentació PFC	02/08/10	31/12/10	110d
7	Preparació de la presentació	03/01/11	28/01/11	20d

Finalment, en el següent diagrama de Gantt es pot veure la distribució temporal gràficament:



3. Anàlisi funcional de la plataforma

En el present apartat es podrà llegir un resum dels requeriments que ha de satisfer la plataforma *Simulbike*. Seguidament es mostraran els diagrames de casos d'ús que descriuen gràficament les funcionalitats de l'aplicació. En el darrer punt del capítol, el lector podrà veure les funcionalitats explicades de forma detallada.

3.1. Resum dels requeriments

Tal i com s'ha comentat prèviament, es pretén desenvolupar una plataforma de simulació de rutes de bicicleta per muntanya.

A nivell de components físics, la plataforma disposa d'un rodet (sobre del qual s'instal·la la bicicleta), la unitat de control (que permet la interacció amb l'usuari) i l'ordinador (que gestiona l'execució de l'aplicació). Un aspecte molt important que s'ha de tenir present és que la interacció de l'usuari amb la plataforma es realitza exclusivament mitjançant la unitat de control. Aquesta última disposa de quatre fletxes de desplaçament (amunt, avall, dreta i esquerra) i un botó de confirmació per executar l'acció. Per tant, la solució ha de tenir present l'ús d'aquest dispositiu d'entrada per la navegació a través de la interfície.

L'aplicació disposa de dos modes de funcionament on cada una té les seves pròpies particularitats. Aquests son:

- **Mode *off-line*:** en aquest cas, l'usuari utilitza l'aplicació sense usar una connexió a Internet, per tant, ha decidit no accedir als serveis *on-line*. La aplicació només podrà ser usada a partir de les dades locals (rutes, descripcions, estadístiques, ...). A més, no podrà mantenir les dades sincronitzades amb les de la web social. Una altra limitació que presenta aquest mode és la de no poder realitzar rutes amb altres usuaris de la plataforma.
- **Mode *on-line*:** es tracta d'un mode de joc en que l'usuari té accés a Internet i vol utilitzar els serveis en línia que la plataforma ofereix. D'aquesta forma pot mantenir-se al dia: amb les seves estadístiques a la web social, les darreres notícies, adquisició de noves rutes, saber si els seus amics estan connectats, rebre reptes, ...

A la aplicació s'accedeix mitjançant un usuari i una contrasenya. Però hi ha una particularitat:

Sigui quin sigui el mode de l'usuari, hi ha una validació d'usuari local (exclusiva en el sistema de l'usuari) per tal de poder entrar dins l'aplicació. Un cop s'ha accedir localment a l'aplicació, es pot introduir (o donar-se d'alta) en el mode *on-line* mitjançant el nom d'usuari i contrasenya que s'utilitza en la web social. D'aquesta forma s'habilita el mode en línia.

Un cop l'usuari es troba dins de l'aplicació, pot realitzar les rutes que desitgi mitjançant alguna de les modalitats disponibles segons el mode habilitat i la disponibilitat d'una connexió a Internet o no. Les modalitats de simulació disponibles son:

- **Modalitat passejar:** aquesta modalitat no presenta cap exigència a l'usuari, s'utilitza per conèixer el recorregut.
- **Modalitat d'entrenament:** és la modalitat que utilitzarà l'usuari per tal d'entrenar-se en la ruta seleccionada..
- **Modalitat competició:** quan l'usuari seleccioni aquesta modalitat, podrà realitzar el recorregut seleccionat competint amb els amics que té assignats.
- **Modalitat desafiament:** aquesta modalitat oferirà a l'usuari la possibilitat de competir amb qualsevol usuari de la plataforma que disposi del recorregut seleccionat.

Per tal de donar un aspecte social a l'aplicació, s'ha de disposar d'un gestor d'amistats i permetre la comunicació entre ells.

Cal recordar que per obtenir informació més detallada sobre el funcionament de l'aplicació cal consultar el document de "Document d'anàlisi de requeriments" que es troba en el punt 3.3 del present document.

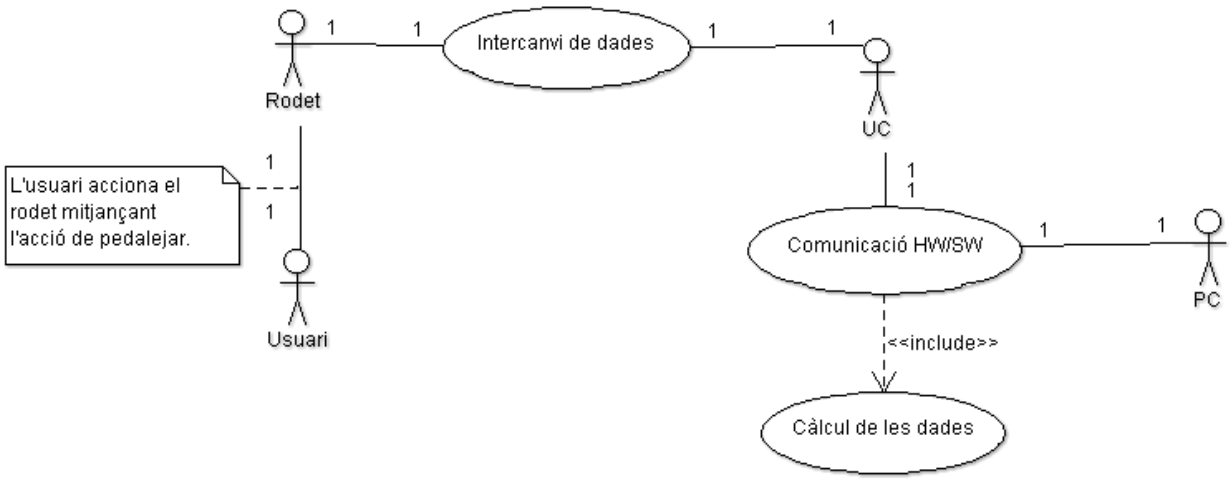
3.2. Diagrames de casos d'ús

En aquest apartat es presentaran els diagrames de casos d'ús que mostren les funcionalitats de l'aplicació d'un mode més gràfic.

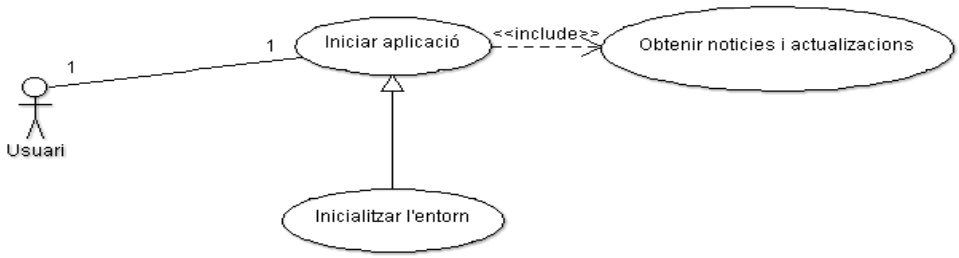
Primerament cal presentar els actors que intervenen en els diagrames posteriors:

- **Usuari:** fa referència a l'usuari principal de l'aplicació. És tracta de la persona que ha adquirit la plataforma i es disposa a utilitzar-la.
- **Servidor:** fa referència al servidor central de la plataforma que ofereix els serveis en línia als usuaris. Aquest component és extern al present projecte.
- **Rodet:** fa referència a l'element sobre el qual s'instal·la la bicicleta de l'usuari. Es tracta d'un component que per si sol realitza accions ja que constantment està enviant les dades de la velocitat i aplica la resistència apropiada al pneumàtic de la bicicleta.
- **UC:** fa referència a la Unitat de Control. Es tracta d'un component que per si sol realitza accions ja que constantment està intercanviant informació entre el rodet i l'ordinador local de l'usuari.
- **PC:** fa referència al ordinador local de l'usuari. Independentment de les accions que executi l'usuari, l'aplicació també executa accions de forma autònoma com ara la sincronització amb el servidor, la validació de la disponibilitat d'una sortida a Internet o la connexió de la unitat de control amb el sistema.

El diagrama de casos d'ús ha estat fraccionat de forma que resulti més comprensible que veure'l en la seva totalitat.

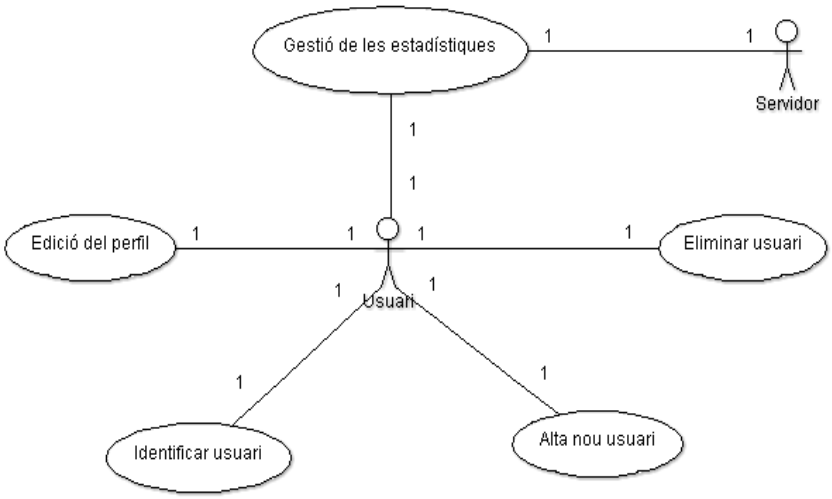


- **Intercanvi de dades:** aquesta funcionalitat fa referència a la comunicació que hi ha entre la unitat de control i el rodet. Concretament hi ha el intercanvi de la velocitat del rodet que s'envia a la unitat de control i el valor de la resistència que la unitat de control envia al rodet.
- **Comunicació HW/SW (Hardware/Software):** en aquesta funcionalitat es descriu la comunicació que hi ha entre la unitat de control i l'ordinador de l'usuari. Les accions que es realitzen son: conversió de la velocitat del rodet a un valor comprensible, control del valor de la resistència segons les indicacions de l'ordinador, enviament de les tecles que l'usuari pitja a l'ordinador, ...
- **Càlcul de les dades:** en aquest cas d'ús es contemplen tots aquells càlculs previs que requereix la obtenció dels valors de la velocitat i la resistència abans de ser enviats al seu destinatari.

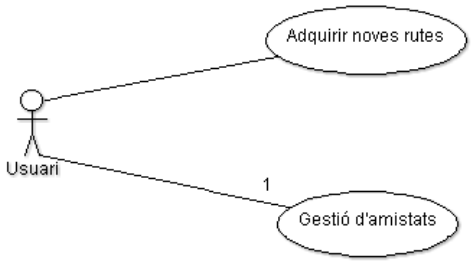


- **Iniciar aplicació:** aquesta funcionalitat contempla totes aquelles accions que s'han d'executar quan s'inicia l'aplicació. Algunes d'aquestes son: comprovació del requisits, comprovació de l'entorn, iniciar la plataforma, controlar el nombre d'instàncies, comprovació de l'existència d'una unitat de control connectada.
- **Inicialitzar l'entorn:** es tracta de la segona fase d'inicialització. Un cop s'inicia una instància de la plataforma, cal adaptar-la al context. És a dir, cal comprovar si hi ha una connexió a Internet, validar la llicència del software o configurar la interfície de l'usuari segons el idioma d'aquest. A més també es gestiona la instal·lació de les actualitzacions.

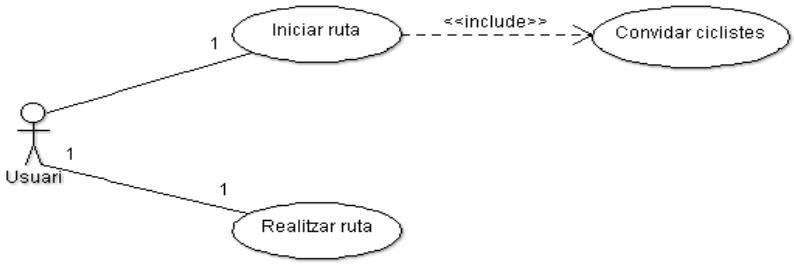
- **Obtenir notícies i actualitzacions:** aquesta funcionalitat gestiona la comprovació de noves notícies i la disponibilitat de noves versions en el servidor.



- **Alta nou usuari:** es tracta de les funcionalitats requerides per donar d'alta un usuari a l'aplicació, des de les validacions que es realitzen fins a les accions per registrar-lo. A més, cal comprovar el nombre d'usuaris que es poden registrar en el sistema local segons la llicència disponible.
- **Identificar usuari:** aquesta funcionalitat contempla totes les accions que la plataforma ha de realitzar per tal de poder decidir si un usuari és vàlid o no (consulta a la base de dades, xifratge de les contrasenyes, ...).
- **Eliminar usuari:** en aquest cas d'ús es gestionen totes les accions que cal realitzar per donar de baixa un usuari (tractament de la base de dades, gestió de les seves dades i alliberament de la llicència).
- **Edició del perfil:** en aquest bloc es contemplen totes les funcionalitats que fan referència a l'activació del mode *on-line*, la actualització de les dades personals, la sincronització de les dades amb el servidor, ...
- **Gestió de les estadístiques:** aquesta funcionalitat contempla totes les accions necessàries per a l'explotació de les dades estadístiques de l'usuari, és a dir, les dades obtingudes al llarg de l'ús de la plataforma. Hi ha les estadístiques genèriques i les estadístiques per rutes. Algunes de les dades que es tracten son la distància total recorreguda, els temps total, la velocitat màxima, velocitat mitjana, ...



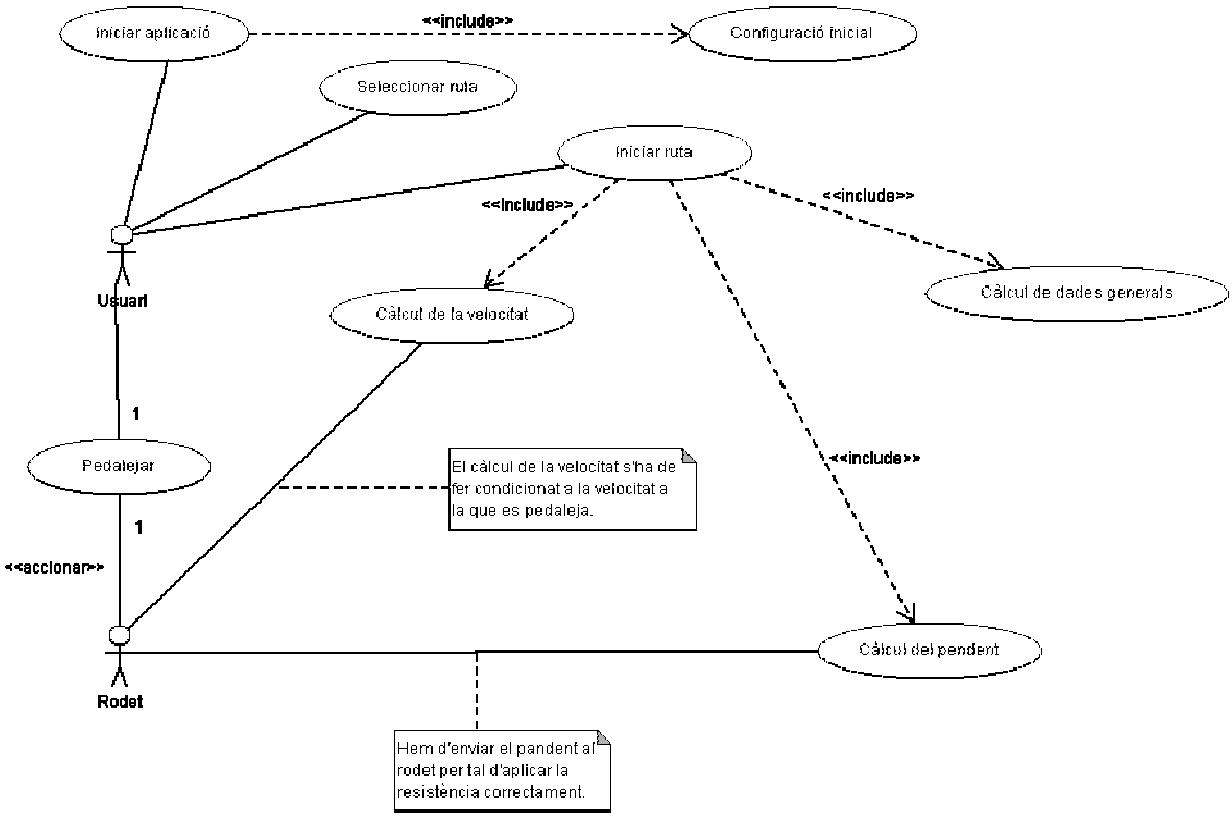
- **Adquirir noves rutes:** en aquest cas d'ús s'inclouen totes les funcionalitats que fan referència a l'adquisició de noves rutes per instal·lar-les en el sistema local de l'usuari. Es contemplen els casos d'adquisicions tant per suports físics com per mitjà de la xarxa per connectar amb el servidor remot.
- **Gestió d'amistats:** com que es pretén donar un aire social a la plataforma, cal un conjunt de funcionalitats per tal de gestionar tot el relacionat amb les amistats. Això implica les accions de afegir amics, eliminar amics, contactar amb amics, cercar amics, detectar amics, ...



- **Iniciar ruta:** en aquest cas d'ús es contemplen les funcionalitats necessàries per tal que l'usuari pugui iniciar una ruta segons les seves preferències, ja sigui en la ruta a realitzar, la modalitat o en la tria dels participants.
- **Realitzar ruta:** en aquest cas d'ús es contemplen les funcionalitats necessàries per tal que l'usuari pugui realitzar el recorregut que prèviament ha configurat. Tracta aspectes com l'actualització de les dades de la interfície, el càlcul de la velocitat de reproducció del vídeo, les dades de competició o la resistència que s'ha d'aplicar al rodet.
- **Convidar amistats:** aquest cas d'ús contempla totes les funcionalitats que l'aplicació ha d'implementar per tal de gestionar principalment la preparació del mode multi jugador i la comunicació entre els diferents usuaris d'un recorregut.

En el següent diagrama de casos d'ús es pot contemplar el prototip de l'aplicació, es correspon a la fita 2 prèviament descrita. En la realització del present projecte, es considera un objectiu de mínims a assolir en el desenvolupament. Observació: no es descriuran els casos d'ús ja que es tracten de funcionalitats similars a

les anteriorment descrites.



3.3. Anàlisis funcional complet

Comunicació Unitat de Control - PC

Tal com s'ha comentat en la introducció, la plataforma *Simulbike* es composa d'uns components hardware i uns software. La comunicació entre aquests elements es realitza mitjançant un cable sèrie que ens connecta la unitat de control amb l'ordinador on s'executa l'aplicació. A través d'aquest, es crearà una connexió bidireccional de forma que les dues parts puguin comunicar-se les peticions i respostes.

Primerament, cal definir la unitat de control com un encaminador de paquets. És a dir,la seva principal funció consisteix en l'enviament de dades: entregar les peticions i respostes als seus corresponents destinataris. Les diferents trames que poden circular son:

- *Velocitat*: son dades que el rodet envia a la computadora mitjançant la unitat de control. El valor que s'envia es correspon amb la velocitat de pedaleig de l'usuari.
- *Tecles*: la unitat de control disposa d'un teclat (amb 4 fletxes i la tecla de confirmar) que serveix per interactuar amb el software de la plataforma. La unitat de control ha d'enviar la tecla pressionada a l'ordinador.

- *Resistència del rodet*: la computadora ha de comunicar al rodet quin és el grau de resistència que s'ha d'aplicar en cada moment (ha de comunicar els canvis sobre el valor actual).
- *Valors de configuració*: en el inici de l'execució de la plataforma, s'ha de mantenir un diàleg amb els components (unitat de control i rodet) per tal d'inicialitzar-los en el mode de funcionament inicial i els paràmetres adients.

Cal desenvolupar un software que permeti la gestió de la unitat de control. Abans de descriure les accions a realitzar, cal definir exactament com està muntada l'arquitectura: la unitat de control és un microcontrolador que permet la comunicació entre el rodet i l'ordinador tal com es pot veure en la següent imatge:

Per tant, el software que s'ha d'implementar en la unitat de control consisteix en un encaminador de dades (entre el rodet i l'ordinador) i, a més, afegir l'enviament de les tecles polsades.

Inici de l'aplicació

Quan l'usuari vol iniciar l'aplicació, aquesta executa una seqüència de passos en el sistema:

- Només es pot tenir una instància oberta de l'aplicació; per tant, cal un control dels processos que estan en execució per tal de comprovar si ja estem executant la plataforma. En aquest cas, cal mostrar una alerta a l'usuari; en cas contrari, s'executa l'aplicació.
- El següent pas consisteix en determinar si el dispositiu està connectat o no. La plataforma requereix la presència del dispositiu per poder funcionar. Aquesta tasca es realitza mitjançant un escaneig de ports sèrie fins a identificar la unitat de control. En cas d'escanejar tots els ports i no trobar-ne cap que es correspongui amb el nostre dispositiu, l'aplicació ha de mostrar un avís indicant les causes (poden ser diverses: problemes en el port, aparell no connectat, aparell sense alimentació,... és responsabilitat de l'usuari verificar tots els possibles problemes). En cas de determinar el port on està connectada la unitat de control, cal poder enviar les senyals de configuració inicials de forma

que la unitat de control i el rodet quedin configurats amb els paràmetres inicials i llestos per poder treballar amb ells.

- El següent pas és determinar si es disposa de connexió a Internet o no. En cas d'èxit, s'habilita el mode *on-line*; en cas contrari, s'habilita el mode *off-line*. Si s'estableix la connexió, el servidor ens ha d'indicar si l'identificador del producte és vàlid o no. Si és vàlid, s'accedeix a la plataforma en mode *on-line*; sinó, es comunica a l'usuari i l'aplicació es posa en mode *off-line*. Cal notar que el primer cop que l'usuari executa l'aplicació, haurà d'introduir l'identificador del producte o saltar-se el pas i funcionar en mode *off-line* (en els posteriors intents, l'aplicació carregarà l'identificador emmagatzemat).
- Finalment, apareix la pantalla inicial d'identificació.

Cal afegir que mentre l'aplicació realitza aquestes comprovacions, es mostra a través de la pantalla una finestra amb la barra de progrés indicant les diferents accions que s'estan realitzant.

Observació: l'aplicació treballa per defecte amb l'idioma que s'ha definit a la instal·lació, però cal tenir present en tot moment que l'aplicació és multilingüe.

Teclat

Tal com s'ha comentat en apartats previs, la plataforma consta d'una unitat de control amb unes tecles: 4 fletxes i un botó de confirmació. Aquesta és la interfície d'entrada que usarà l'usuari per interactuar amb la plataforma; és a dir, la interfície gràfica s'ha de preparar per tal de que es pugui navegar simplement amb l'ús d'aquestes tecles.

Una de les principals limitacions que s'observen amb aquests botons és la funció de cancel·lar/tornar. Com que només tenim la tecla de confirmar, no hi ha el botó d'anul·lar; això implica que la interfície gràfica de l'aplicació sempre ens ha de mostrar aquesta opció per tal que l'usuari la pugui confirmar i, d'aquesta forma, realitzar una cancel·lació o un retorn.

El desplaçament a través dels diferents elements de la interfície es realitza mitjançant dos criteris:

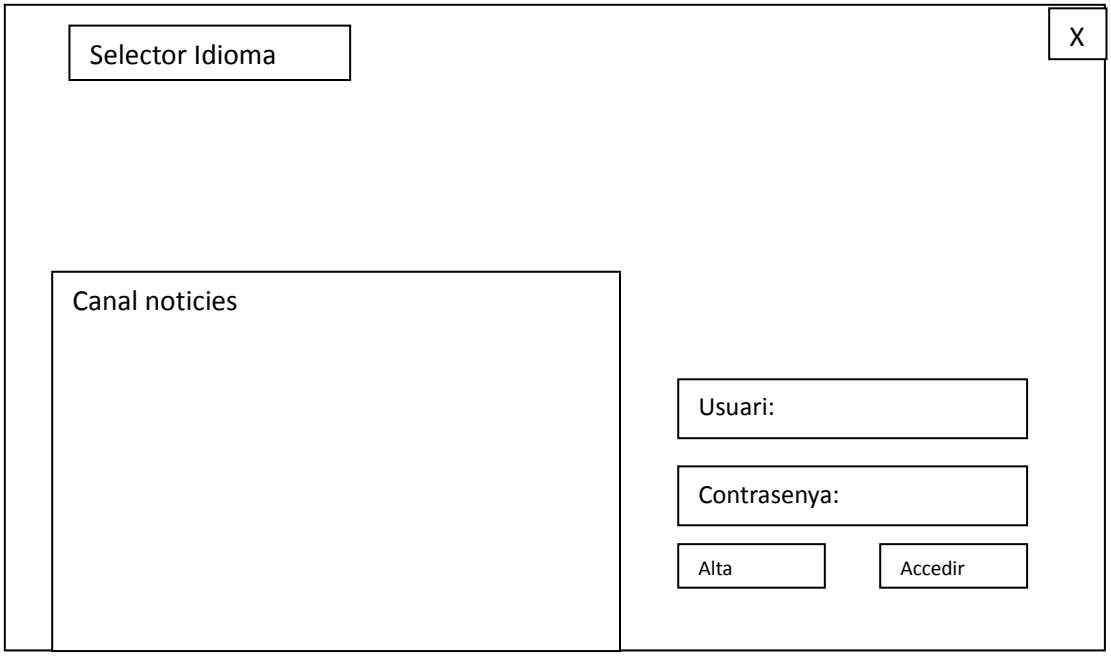
- Tecla polsada: la tecla polsada ens indica quina direcció vol prendre l'usuari mentre navega per la interfície.
- Confirmació: la tecla de confirmació ens indica que l'usuari vol realitzar una acció sobre l'element seleccionat.

Per tant, cal un índex dels elements per poder ordenar-los i saber en tot moment, a partir de la posició actual i la tecla polsada, a on ha de saltar el focus.

Els camps de text presenten una particularitat: l'usuari ha de ser capaç d'escriure text a partir dels 5 botons disponibles. Per això, cal un teclat virtual que aparegui quan l'usuari pressioni la tecla confirmar i l'element actual sigui un camp de text. El teclat virtual ha de permetre escriure tot els texts que l'usuari vulgui simplement desplaçant-se per els diferents dígits o caràcters i pitjant la tecla de confirmar (incloses les opcions de retrocedir, avançar, esborrar, majúscules i tancar).

Identificació

La pantalla d'identificació és la primera pantalla amb la que es troba l'usuari. Aquesta es composa de diferents parts tal i com es mostra en el següent esquema:



Identificació d'usuari

Tot seguit descriurem les diferents parts i les funcionalitats que es poden observar en l'esquema:

- Des de la pantalla inicial, l'usuari podrà seleccionar l'idioma de la interfície. Per defecte estarà marcat l'idioma que l'usuari va definir en la instal·lació. Quan l'usuari canvia l'idioma, l'aplicació el desa com a idioma predeterminat; a més, l'aplicació ha de refrescar les cadenes de text constants que es mostren a la interfície.
- Hi ha un apartat on es mostraran les noticies. Aquestes noticies s'actualitzaran des de la plataforma on-line si l'usuari esta en aquest mode o donaran informació de les avantatges d'activar el mode on-line.
- Hi ha la opció de donar d'alta un nou usuari. En aquest apartat cal tenir present el dos modes de funcionament de la plataforma. Quan l'usuari es dóna d'alta, l'aplicació ha de realitzar diferents comprovacions:

  - Una aplicació només pot funcionar amb un nombre limitat d'usuaris. Per tant, l'aplicació no podrà donar d'alta més usuaris dels que permet la plataforma. En cas de tenir tots els espais d'usuaris ocupats, l'aplicació mostrarà un missatge indicant la causa i informarà que s'ha d'eliminar un usuari existent abans de poder-ne crear un de nou.
  - En cas de tenir un espai disponible, l'aplicació mostra el formulari d'alta d'usuari (veure següent funcionalitat: Alta d'usuaris).

- Finalment hi ha l'apartat o formulari d'introducció de l'usuari i la contrasenya per tal d'accedir a l'aplicació. Per realitzar aquesta funció, l'usuari omple els dos quadres de text (on la contrasenya es mostra oculta amb asteriscs) i tot seguit pressiona sobre el botó d'entrar. L'aplicació valida si les dades son correctes. En cas d'èxit, carrega les dades de l'usuari i inicia l'aplicació; en cas contrari, retorna a la pàgina d'identificació i comunica a l'usuari que el nom o la contrasenya son incorrectes (intentant identificar la causa).

Alta d'usuaris

Tal i com s'ha comentat prèviament, l'aplicació permet el registre d'un nombre limitat d'usuaris. Aquest valor inicialment és un valor per defecte. Posteriorment es podrà ampliar en el mode d'ús on-line, realitzant ampliacions de la llicència.

Les úniques restriccions que hi ha a la hora de crear nous usuaris son: no es permet el duplicat dels noms, ni l'ús de caràcters especials i hi ha una longitud màxima en el nom. Cal observar en aquest punt que el nom d'usuari no es pot duplicar localment. Això és degut a que realment es defineixen dos usuaris:

- *Usuari local*: ve definit per el nom que s'assigna l'usuari; aquest valor és per accedir a la plataforma localment (independentment del mode on-line o off-line).
- *Usuari remot*: aquest identificador serveix per identificar a l'usuari en la plataforma on-line. En aquest cas, si que es tracta d'un valor que no es pot duplicar en cap usuari de la xarxa i requereix una contrasenya que l'usuari utilitzarà per habilitar el mode on-line (tant des de la plataforma software com des de l'aplicació web). Aquesta segona identificació només es realitza en el moment de la creació (en els següents cops, aquesta identificació es realitza de forma transparent a l'usuari quan aquest entra a l'aplicació). La avantatge d'aquesta doble identificació és que l'usuari pot reinstal·lar l'aplicació però conserva la sessió on-line i pot alternar entre els modes off-line i on-line.

L'aplicació ha de gestionar la informació dels usuari, aquesta és: nom, cognoms, *nick* (local), nom d'usuari (remot), contrasenya local, contrasenya remota, correu electrònic, sexe, classificació, imatge d'avatar, alçada, pes, dades de resum genèriques (kilòmetres totals recorreguts, temps total, velocitat màxima, velocitat mitjana, Kcal consumides ...) i dades de totes les rutes realitzades (es tracta de les mateixes dades però en aquest cas es divideixen per rutes).

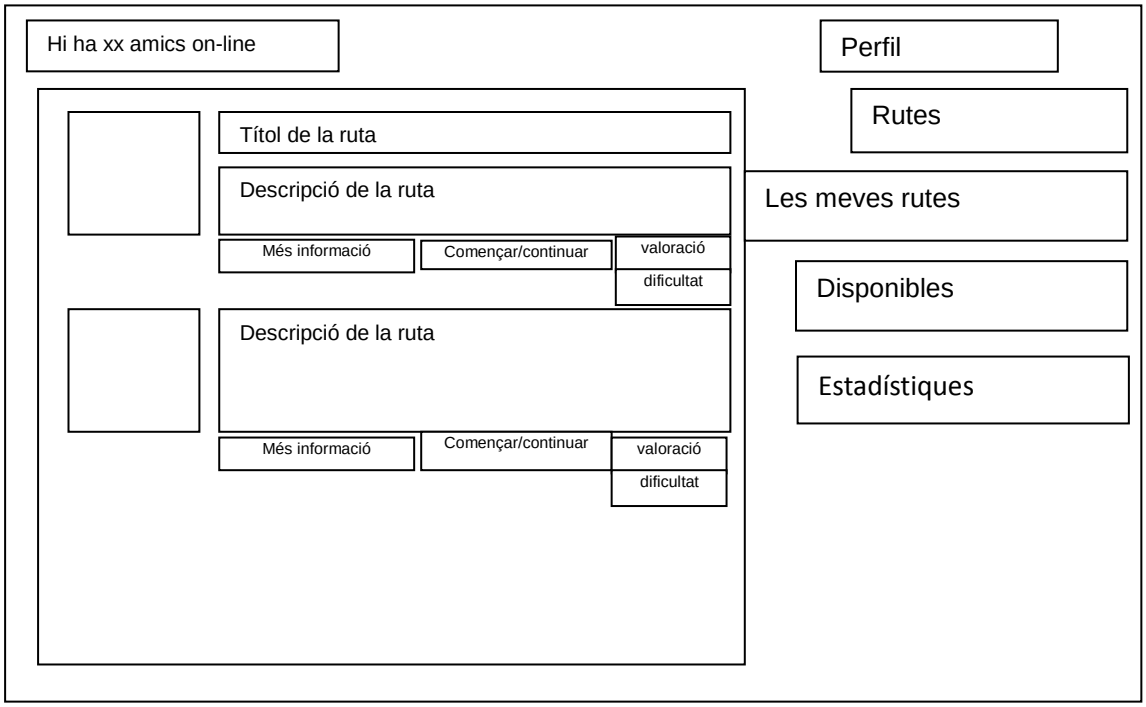
El formulari d'alta d'usuaris mostra els següents camps (tenint en compte la navegació i el teclat virtual descrits en la funcionalitat del teclat):

- Nom: camp de text.
- Cognoms: camp de text.
- Nick: camp de text alfanumèric.
- Contrasenya local: camp de text alfanumèric ocult amb asteriscs.
- Correu electrònic: camp alfanumèric amb validació de format.
- Sexe: menú desplegable que mostra les opcions de Home/Dona.
- Imatge: apareix un botó que obre un navegador de fitxers on l'usuari podrà seleccionar una imatge emmagatzemada a disc per usar com a avatar. L'aplicació validarà la mida de la imatge; si aquesta està dins les proporcions adequades, es desarà com a avatar de l'usuari.
- Edat: camp numèric.
- Pes: camp numèric amb decimals (unitats en quilos).
- Alçada: camp numèric amb decimals (unitats en metres).
- Idioma: es tracta d'un menú desplegable que presenta els idiomes disponibles.

Finalment hi ha els botons de desar (per crear l'usuari amb les dades indicades) i anul·lar (per retornar a la pàgina d'identificació sense desar els canvis).

Pàgina principal de la sessió

Amb aquesta funcionalitat, es contempla la situació en que l'usuari ha estat registrat a l'aplicació correctament i s'inicia la seva sessió. En el següent esquema es poden observar les diferents opcions que pot trobar-se en accedir a l'aplicació.



Les meves rutes

Les opcions seran descrites en els següents punts del present document.

Edició del perfil

Un cop l'usuari es troba dins la seva sessió (ja ha estat registrat correctament), es trobarà en la pàgina principal. En aquesta, hi ha la opció d'accedir al perfil de l'usuari. El perfil de l'usuari presenta una forma similar a la del formulari d'alta, però en aquest cas conté els valors. L'usuari pot editar les seves dades mitjançant el formulari i posteriorment desar els canvis.

Si l'usuari vol canviar la contrasenya, l'aplicació obre una nova vista on es sol·licita que s'introdueixi la clau antiga, la nova i que es repeteixi la nova. Si l'aplicació valida les dades i son correctes, s'aplicarà els canvis a l'usuari; en cas de tenir la clau actual errònia o les noves claus no son iguals, l'aplicació comunica l'error a l'usuari sense realitzar canvis en les dades d'aquest.

Un apartat molt important en aquesta vista és l'activació del mode on-line. Si l'aplicació detecta una connexió a Internet, apareixerà la opció de mode on-line. En cas contrari, apareixerà un missatge avisant que no s'ha localitzat una connexió a la xarxa. En cas d'existir una connexió, l'aplicació pot trobar diferents escenaris:

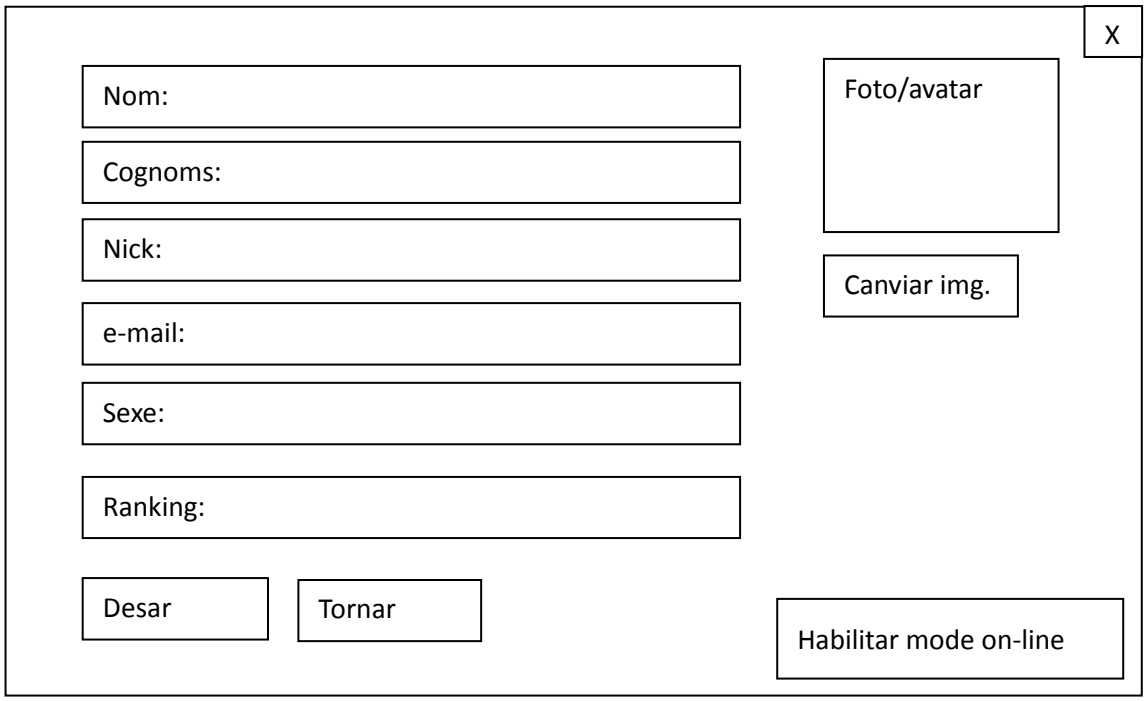
- L'usuari no està donat d'alta en el mode on-line: en aquest cas, l'aplicació obrirà el formulari d'alta, on l'usuari haurà d'indicar un nom d'usuari (únic a la xarxa, es valida quan el camp perd el focus), una contrasenya i la repetició d'aquesta; a més, apareixerà el camp de correu electrònic ja que l'usuari s'haurà d'activar. L'Aplicació enviarà les dades al servidor. Cal que aquest ens retorni el re-

sultat per tal d'indicar-li a l'usuari. Si tot és correcte, l'usuari i la contrasenya es desen en el sistema local, de forma que la doble identificació es realitza de forma transparent a l'usuari.

- En cas d'estar registrat en mode on-line: l'usuari veurà la opció d'habilitar-lo o no des del seu perfil (aquesta preferència queda memoritzada en la plataforma). En cas d'habilitar-lo, l'aplicació ha d'enviar la sol·licitud d'identificació al servidor i aquest ens respon segons el resultat (identificació fallida, usuari no actiu o identificació correcte).
- En cas de ser el primer cop que accedeix a l'aplicació local però l'usuari ja disposa d'una compte on-line prèvia: l'aplicació permet la opció de "ja disposo d'una compte on-line" i l'usuari simplement ha d'indicar el nom d'usuari i la contrasenya (es valida contra el servidor). En punts posteriors es comentarà la sincronització de les dades.

D'altra banda, apareix una opció d'eliminar usuari (localment). Apareix un missatge de confirmació on s'avisa a l'usuari que és un pas irreversible. L'usuari pot confirmar o anul·lar l'acció.

Finalment, apareix la opció de reiniciar les dades acumulades de l'usuari. Això implica netejar totes les rutes realitzades per l'usuari (tant localment com remotament). Novament es tracta d'un pas irreversible que requereix que l'usuari confirmi o anul·li l'acció.



El meu perfil

Llistat de les "meves rutes"

El llistat de rutes disponibles ens mostra la informació de les rutes que hi ha en la màquina local. Tal com s'ha descrit en els punts anteriors, les rutes es composen de diferents elements, concretament en aquesta llista es treballa amb les dades descriptives i la imatge que convé mostrar.

Es tracta de mostrar una llista de les rutes emmagatzemades, per això s'enumeraran les rutes que s'han instal·lat a l'ordinador.

Per cada ruta que es localitza, es crea una fila a la taula on es mostren les següents dades:

- Imatge.
- Nom de la ruta.
- Descripció: es mostra parcialment, apareix la barra de desplaçament vertical per tal que l'usuari pugui anar llegint la descripció completa.
- Més informació: obrirà una finestra amb informació ampliada de la ruta.
- Valoració: puntuació de la ruta en el moment de la descarrega si és mode off-line o l'actual si s'està en mode on-line (hi ha una fase de sincronització per la actualització de les dades que s'explicarà posteriorment).
- Dificultat: les rutes es classifiquen mitjançant diferents colors d'acord el grau de dificultat; aquests van des del verd (més fàcil), blau, vermell i negre (més difícil). Com s'ha comentat, la plataforma simularà els pendents aplicant major o menor resistència en el pedaleig. Per aquest motiu, segons l'esforç que requereixi l'usuari i la longitud del recorregut, la ruta es classificarà com més o menys dura.

Verd	Fàcil
Blau	Mitja
Vermell	Difícil
Negre	Molt difícil

Finalment es mostra un botó que ens permetrà iniciar la ruta si no està començada o continuar-la si es troba a mitges.

L'usuari pot adquirir les rutes de dues maneres diferents. Per un costat, pot descarregar-les des de l'aplicació o baixar-les des d'Internet i instal·lar-les posteriorment a l'aplicació. Les dues opcions es comentaran en el següent punt del present document.

Adquisició de rutes (rutes disponibles)

Tal com s'ha comentat, l'usuari té dues vies per la instal·lació de noves rutes:

- *Mitjançant la pròpia plataforma*: en la pantalla inicial hi ha la opció de rutes disponibles (només en el mode on-line). Si l'usuari pitja sobre aquest vincle, accedirà a una llista, idèntica a la llista de ru-

tes instal·lades però amb el botó de descarregar. Quan l'usuari pitgi sobre el vincle de descarregar la ruta, s'iniciarà el procés de descarrega en mode ocult (sense bloquejar l'aplicació). En aquest cas, cal indicar que s'ha d'analitzar si s'aplica una forma de pagament per adquirir les rutes.

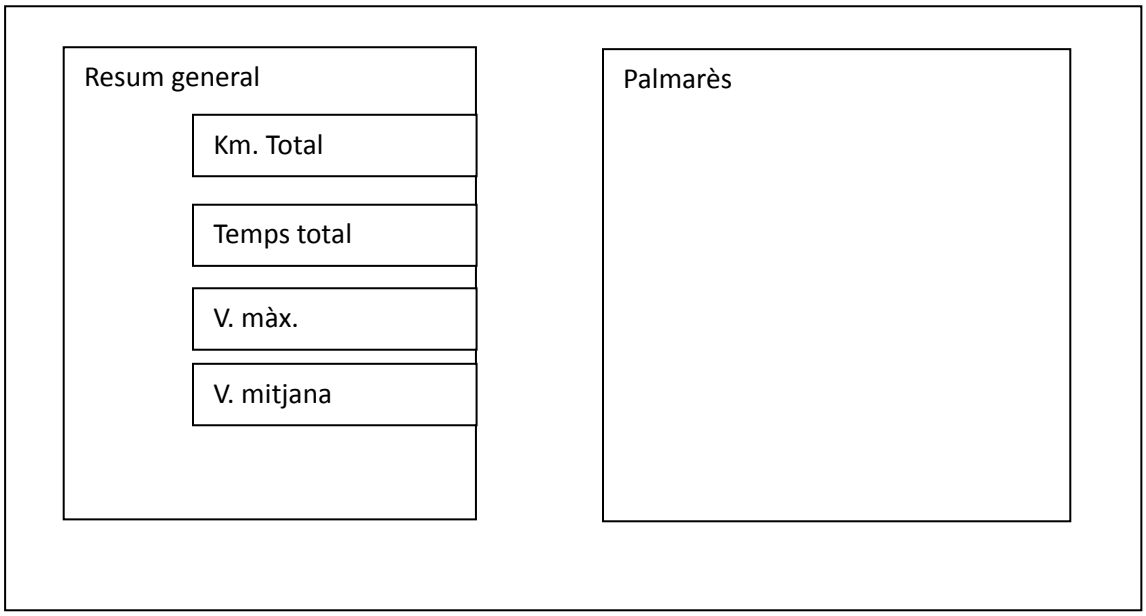
- *Mitjançant accés a Internet*: l'usuari pot descarregar rutes des d'una computadora connectada a Internet. Una ruta està formada per un paquet que conté els diferents fitxers que prèviament han estat descrits. El desempaquetar el paquet, els fitxer s'instal·laran en el sistema de fitxers de l'aplicació que automàticament la afegirà entre les rutes de l'usuari.

Estadístiques

Tal i com s'ha comentat, l'aplicació registra totes les dades dels usuaris a mesura que van usant la plataforma. Aquestes dades son utilitzades per l'obtenció d'estadístiques i l'anàlisi de l'evolució del ciclista.

Concretament es poden observar dues parts importants en referència a les estadístiques:

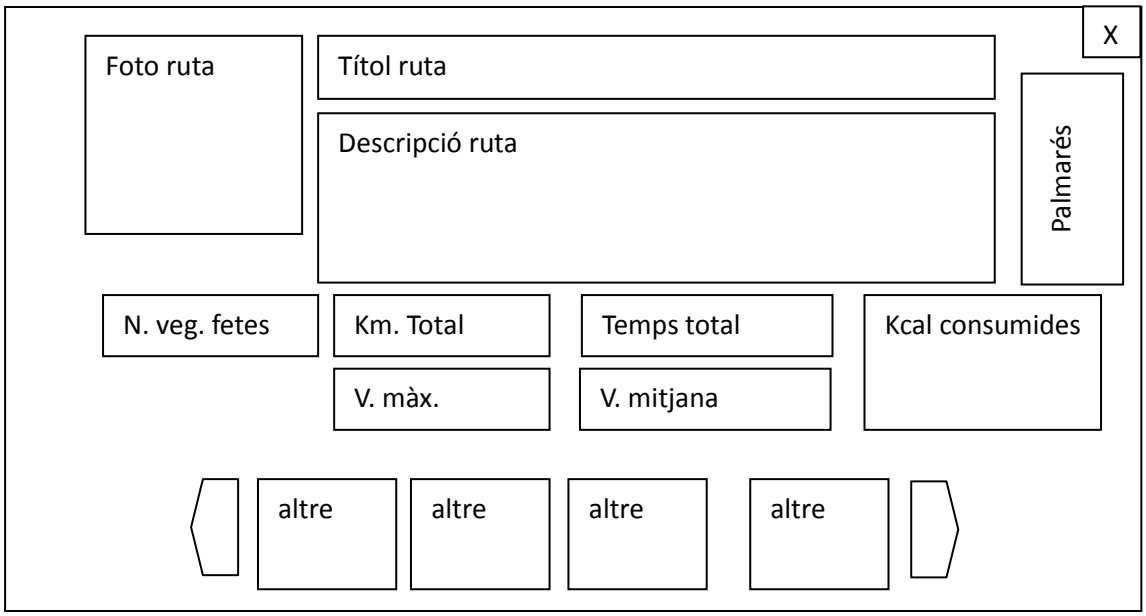
- Resum general: en aquest apartat, es pot observar un càlcul general de les dades de les rutes realitzades per el ciclista actual. Concretament es parla de la distància total recorreguda per l'usuari, el temps total que ha estat pedalejant en la bicicleta, la velocitat màxima aconseguida i la velocitat mitjana. Poden sorgir noves dades durant l'evolució del projecte. En la següent imatge es pot observar un esquema de la pantalla del resum general d'un usuari.



Estadístiques de resum general

A més, en aquesta pantalla apareixerà el nombre de medalles obtingudes (d'or, plata i bronze que es guanyaran segons les posicions en que es quedi en les competicions) i els trofeus (aquests es guanyaran segons les posicions en les que es quedi en els desafiaments). En altres paraules, es mostra el palmarès de l'usuari.

- Estadístiques per rutes: en aquest cas, l'usuari podrà estudiar les dades de les rutes que ha realitzat de forma individual. En l'esquema es pot observar que es mostra informació de la ruta, les vegades que l'usuari l'ha realitzat, la distància total, la velocitat màxima, la velocitat mitjana, el temps mínim, el temps total, el temps mitjà de durada de la ruta, les kilocalories consumides, ... Novament, les dades a calcular podran ser ampliades a mesura que l'aplicació evolucioni. L'element més important és el selector inferior de rutes, on l'usuari seleccionarà la ruta que vol analitzar.
  Novament, en aquesta vista també es veu el palmarès de l'usuari respecte a la ruta seleccionada.



Estadístiques per rutes

Un aspecte molt important respecte a les dades es troba en el mode on-line. En aquest mode, hi haurà una sincronització amb la base de dades del servidor, de forma que es puguin obtenir dades estadístiques genèriques des de l'aplicació web (nombre de vegades que tots els usuaris han realitzat una ruta, velocitat màxima, temps mínim, ...) per tal de presentar les rutes usant com a base l'experiència dels usuaris (recordem que es pretén donar una dimensió social a la plataforma).

D'altra banda, les dades dels usuari seran tractades per tal d'establir una classificació de ciclistes, establiment de nivells, categories o promocions.

Una observació que cal tenir present és que les estadístiques només les formen aquelles rutes que han estat finalitzades i que tenen una modalitat competitiva (aquestes modalitats es detallaran posteriorment en el present document).

Selecció i preparació de la ruta

Quan l'usuari vol realitzar una ruta, ha d'accedir al la llista de les meves rutes que s'ha descrit prèviament. Des d'aquesta llista, pot seleccionar la ruta que vol realitzar i inicialitzar-la (o continuar-la en cas de tenir-la a mitges). En cas que l'usuari pugui continuar la ruta, l'aplicació pregunta si es vol reiniciar o continuar (en cas de reiniciar, la ruta a mitges es perd).

El següent pas que ha de realitzar l'usuari consisteix en seleccionar la modalitat en la que vol realitzar la ruta. Hi ha diferents modalitats:

- *Modalitat passejar*: es tracta de la modalitat en que no hi ha cap exigència a l'usuari. En altres paraules, les dades que s'obtenen en aquesta modalitat no es tenen en compte a l'hora d'actualitzar les estadístiques. Aquest fet és degut a que l'aplicació ha de tenir un al·licient competitiu i si els valors del passejos s'usessin en les estadístiques, aquestes quedarien distorsionades. Tot i l'al·licient, es permet als usuaris realitzar rutes sense la pressió del cronòmetre. És una de les modalitats que es pot realitzar de forma individual o amb grup. Aquesta modalitat es pot aturar per ser continuada en un altre moment i pot ser usada en mode on-line i off-line.

- *Modalitat d'entrenament*: és la modalitat que l'usuari utilitzarà per practicar i anar millorant les seves estadístiques. És a dir, en aquesta modalitat, l'usuari ja entra en el repte que suposa el cronòmetre. És la forma que té l'usuari per realitzar un recorregut on el rival és el cronòmetre. Es tracta d'una modalitat exclusivament individual i no es pot interrompre ja que les dades quedarien distorsionades. Aquesta modalitat pot ser usada en mode on-line i off-line.

- *Modalitat competició*: si l'usuari tria aquesta modalitat, passarà a triar els usuaris amb els que competirà entre els seus amics (posteriorment es parlarà de la gestió de les amistats). En aquesta modalitat, els usuaris podran realitzar una ruta on l'objectiu consisteix en finalitzar el recorregut el més ràpid possible per tal de guanyar als contrincants. Es tracta d'una modalitat exclusiva del mode on-line.

- *Modalitat desafiament*: aquesta modalitat permetrà visualitzar els usuari connectats que tenen disponible la ruta seleccionada de forma que els puguis convidar al repte. Els usuaris es juguen puntuació per la classificació general de ciclistes segons la posició en la que es quedi (segons el rang de punts en que es trobin els usuaris, l'aplicació els assigna una categoria o altre). En aquest cas, cal indicar que els usuaris seran penalitzats amb la pèrdua de punts en cas d'abandonament (s'exigeix una implicació per part dels usuaris). És una modalitat estrictament on-line.

Tal com es pot observar, segons la modalitat de recorregut que es seleccioni, hi ha el pas previ de convidar a altres ciclistes. Això es realitza de dues maneres segons si és la modalitat competició (o passejar) on només visualitzem amics o la modalitat de desafiament on es visualitzen tots els usuaris connectats a la plataforma en mode on-line.

- *Modalitat competició o passeig*: en aquest cas es mostra una llista amb les amistats on es pot observar si estan connectades o no. En cas d'estar connectades, es podrà enviar una invitació que l'usuari destí veurà en forma de finestra sobreposada on s'indica: nom de l'emissor, ruta i modalitat. El destinatari tindrà dues opcions: acceptar la invitació (l'aplicació passa a mostrar la vista de preparació de la ruta) o refusar-la (l'aplicació continua tal i com estava abans de la interrupció). En qualsevol dels dos casos, l'emissor rep la resposta. Cal determinar si es posa un màxim de participants (això es determinarà en la fase de proves segons la càrrega de participants que es poden gestionar sense una caiguda del rendiment). Una observació important és que hi ha situacions en que un usuari destí està bloquejat (no pot rebre invitacions): quan es troba realitzant una competició o quan es troba realitzant un desafiament.

  Quan l'usuari emissor determina que ja hi ha prou usuaris, es pot iniciar la ruta (enviant el senyal a tots els participants).

- *Modalitat desafiament*: en aquest context,l'usuari ha de poder veure tots els usuaris connectats. Perquè aquesta funcionalitat pugui tirar endavant, es defineix el concepte de sala de ruta. Una sala és un espai comú per els usuaris on poden veure la resta d'usuaris que busquen altres ciclistes amb els que realitzar el desafiament (de la ruta que indica la sala). En la sala hi ha dos tipus d'usuaris: els que estan esperant desafiament i els que creen nous desafiaments. Els creadors creen un desafiament i son els responsables d'enviar les invitacions als usuaris que esperen (si aquests accepten, abandonen la llista d'usuaris en espera i entren en el desafiament). Un cop el creador del desafiament ha trobat els usuaris desitjats, pot indicar el començament de la ruta. La resta d'usuaris ha de confirmar per tal de que pugui iniciar-se (s'espera un temps determinat, un usuari que ignori l'aplicació no pot bloquejar a la resta).

  Cal dir que els usuaris que esperen, poden veure la llista de desafiaments oberts i entrar-hi en cas de ser públic. El creador del canal pot expulsar usuaris del seu desafiament o marcar el desafiament com a privat de forma que només ell pot enviar les invitacions.

En els dos casos, quan un usuari accepta una invitació o s'entra en el grup de desafiament, automàticament s'habilita la compartició de micròfon; d'aquesta forma, els membres dels grups poden parlar entre ells. Aquesta comunicació es manté al llarg de tota la ruta; així es potencia l'aspecte social de la plataforma. Tot i això, els usuaris tindran la opció de inhabilitar el micròfon.

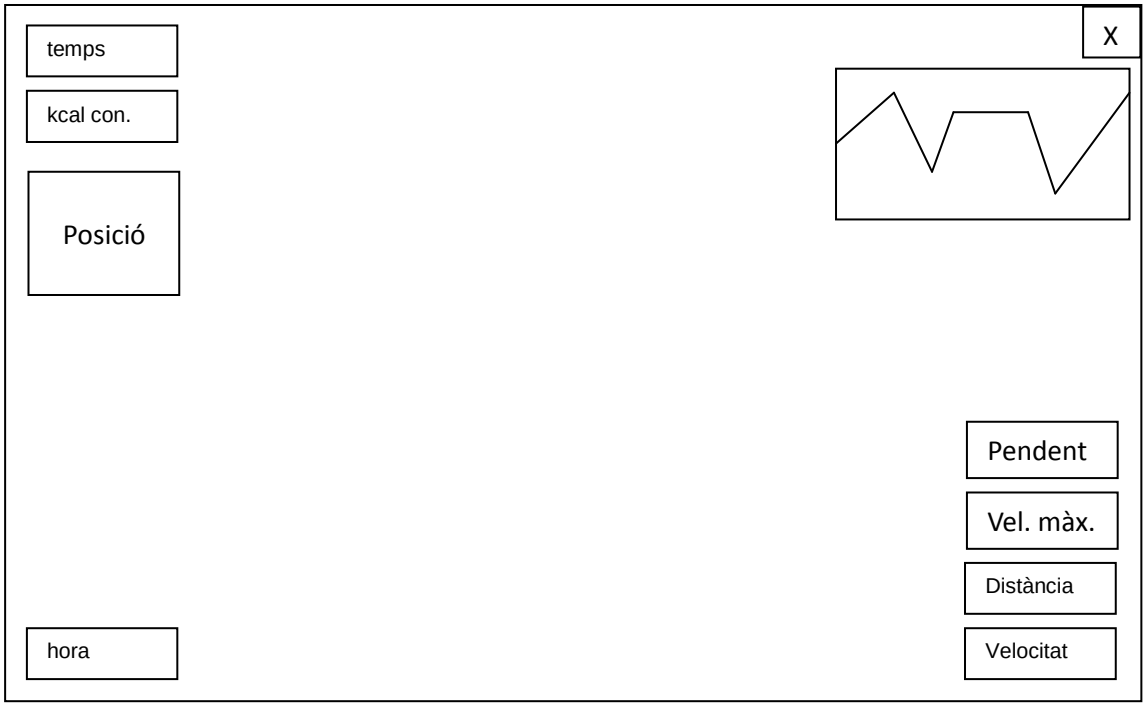
Un cop comença la ruta, es salta en el següent pas: *realització de la ruta*.

Realització de la ruta

La realització d'una ruta està molt condicionada per la modalitat en que es realitza i per el fet de si és una ruta individual o col·lectiva. Tot i així, hi ha uns aspectes comuns ens tots els casos. Primerament descriurem

aquests aspectes per posteriorment tractar les diferències entre els diferents escenaris.

La vista genèrica es mostra en la següent imatge:



Execució de la ruta

En la vista de l'execució de la ruta, sempre es podrà observar el temps que es porta realitzant la ruta i les kilocalories consumides (posteriorment s'analitzarà com calcular-les). A més, mostrarà la distància recorreguda en kilòmetres, la velocitat en kilòmetres per hora i la velocitat màxima que l'usuari ha aconseguit (també en kilòmetres hora).

En una de les cantonades, es mostrarà un mapa d'alçades amb la posició actual del ciclista i la pendent en graus (que a més és utilitzada per a determinar la resistència que s'aplica al rodet tal i com s'ha comentat prèviament).

Des d'aquesta vista, si l'usuari pitja el botó de confirmació, l'aplicació mostrarà un menú on es mostren les següent opcions:

- *Aturar la ruta*: aquesta opció només es troba disponible en les modalitats de passejar i entrenament (tot i que en aquest darrer cas, el cronòmetre continua avançant).
- *Desar i sortir*: aquesta opció només està disponible en la opció de passejar i serveix per interrompre la realització de la ruta, desar el punt actual de forma que l'usuari pugui continuar en algun moment posterior.

- *Abandonar*: aquesta opció serveix per deixar la ruta. Totes les modalitats tenen aquesta opció i les conseqüències depenen directament de la modalitat que l'usuari estigui realitzant en aquell moment.

En les rutes que es realitzen mitjançant una modalitat col·lectiva, cal indicar algunes variacions:

- En el mapa d'alçades, apareix la posició de tots els corredors (en la sala del desafiament, els usuaris poden triar el color que els identifica).
- Apareix la posició de l'usuari actual.
- Apareix els usuaris que van en les tres primeres posicions, així com la distància els separa entre ells i l'usuari actual.

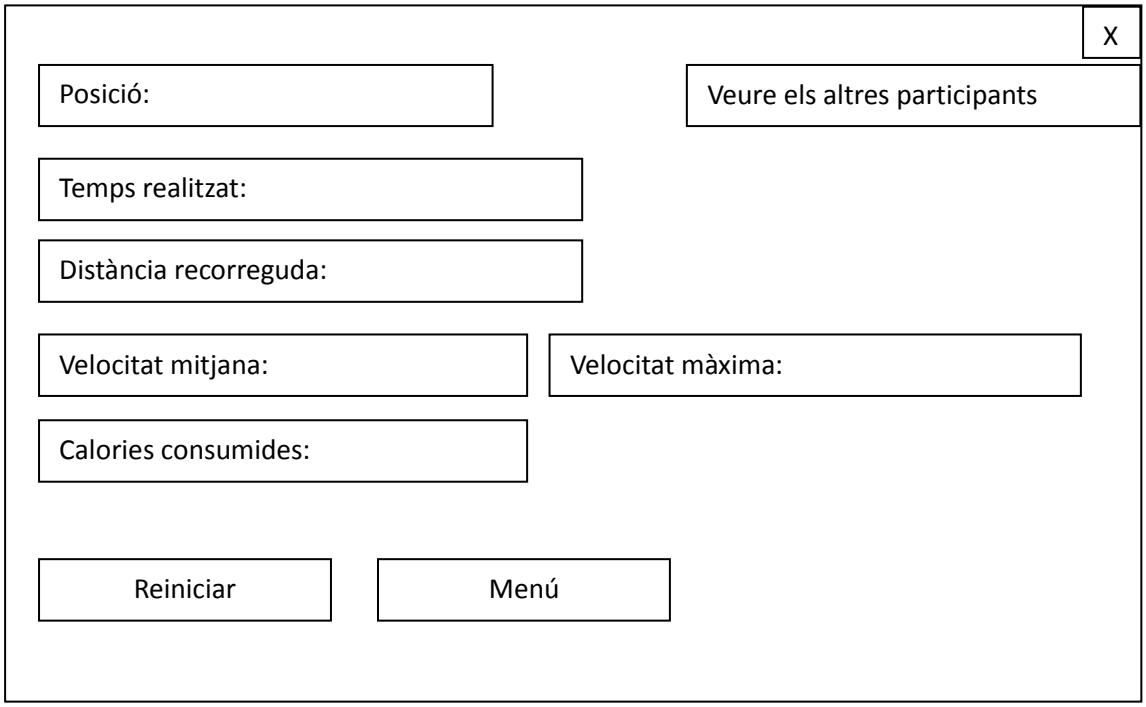
Durant aquest procés es realitzen una sèrie de càlculs per posicionar els usuaris, adaptar la velocitat actual a la que pedaleja l'usuari amb la velocitat de reproducció del vídeo de la ruta digitalitzada o la resistència que cal aplicar al rodet segons el pendent (càlcul del pendent). Les operacions i algorismes que s'aplicaran per a la obtenció d'aquests valors es descriuran en punts posteriors del present document.

Finalització de la ruta

L'aplicació podrà determinar quan es finalitza una ruta ja que aquest punt ve indicat per el final del vídeo. Tot i que es podrà marcar un final uns instants previs per tal d'anar difuminant el vídeo mentre es mostren les dades final de la realització de la ruta.

L'aplicació realitzarà diferents accions segons la modalitat en que l'usuari hagi realitzat la ruta:

- *Modalitat passejar*: en aquest cas es mostrarà un resum de la realització de la ruta. Les dades que s'inclouran seran: el temps realitzat, la distància recorreguda, velocitat mitjana, velocitat màxima i calories consumides. A més apareixeran els botons de reiniciar i el d'anar al menú principal. En cas de tenir la modalitat de passejar en grup, l'aplicació indicarà que els altres usuaris estan finalitzant la ruta; un cop tots hagin finalitzat, apareixerà un vincle per accedir a la llista on apareixeran els corredors segons la classificació, mostrant el temps i la velocitat mitjana de cada un. En cap cas hi haurà una actualització de les estadístiques dels usuaris.



- *Modalitat d'entrenament*: en aquest altre cas es mostrarà un resum de les dades de l'usuari. La principal diferència respecte el cas anterior és que en aquest, les dades si que son usades per actualitzar les estadístiques.

- *Modalitat de competició*: En aquest cas, les dades importants son les classificacions. Per tant, el primer que veurà l'usuari serà la taula classificatòria on es destacaran els tres guanyadors (que obtindran la medalla). Els usuaris tindran un vincle per accedir a una vista de dades idèntica a la descrita en el punt anterior per consultar les dades de cada corredor. Les dades de la modalitat competició també son usades en l'actualització de les estadístiques (a més, cal incloure l'assignació de la medalla).

- *Modalitat de desafiament*: aquest modalitat és molt similar a l'anterior. La principal diferència radica en que en la llista es veu la posició dins la classificació general on-line després de la realització de la ruta (és a dir, com una columna més de la llista, l'ordre dels corredors és el d'arribada). Els usuaris també podran veure les seves dades. Aquestes dades son usades per actualitzar les estadístiques dels corredors a nivell individual, per l'actualització de la classificació general i l'assignació del trofeus.

Gestió de les amistats

Al llarg del present document s'ha anat comentant l'aparició de les amistats amb els quals es poden realitzar les rutes i interactuar amb ells. En aquest punt parlarem de la seva gestió.

Primerament cal remarcar que la amistat és un concepte que apareix exclusivament en el mode *on-line*. La

forma de referenciar-los és mitjançant el nom d'usuari *on-line* ja que aquest els permet identificar-los de manera única (no es poden donar valors repetits).

En la pàgina principal, es pot veure que hi ha el comptador d'amics connectats. L'usuari pot accedir a aquest comptador i entrar en el gestor d'amistats quan pitja el botó de confirmació tenint aquesta opció marcada. En la vista apareix una llista d'amistats on l'usuari pot visualitzar els que es troben connectats i els que no (és a dir, el seu estat).

A més, l'usuari pot seleccionar una fila de la llista i posteriorment anar al botó d'eliminar. En aquest cas, l'aplicació preguntarà per quina opció es vol triar:

- Anul·lar: l'aplicació retorna a la llista sense realitzar cap acció.
- Eliminar: l'aplicació esborra l'amistat de l'usuari.
- Eliminar i bloquejar: l'aplicació elimina l'usuari i bloqueja els intents d'aquest de contactar amb l'usuari (per exemple, mitjançant la invitació a realitzar rutes).

Finalment, també hi ha una opció per tal que l'usuari pugui afegir noves amistats. Es mostrarà un camp de text on l'usuari escriurà el nom de l'amistat. L'aplicació enviarà una notificació a la nova amistat i que haurà de confirmar (afegir a l'emissor de la petició a la seva llista) o refusar (no l'accepta a la seva llista).

Dades

Quan treballem amb la plataforma Simulbike, es pot observar que el volum principal de dades amb les que tracta son les rutes: la seva definició, descripció i posicionament.

Tal com s'ha comentat prèviament en el present document, una ruta ve definida per un seguit de fitxers on cada un té la seva funcionalitat.

El primer fitxer que es té en compte és el vídeo on hi ha enregistrada la ruta. Es tracta d'un vídeo convencional que té la característica d'haver estat enregistrat en una velocitat constant, o mantenint el màxim aquesta constància. Això és degut a que la velocitat de reproducció s'ha d'ajustar a la velocitat de pedaleig; per tant, necessitem com a mínim una velocitat de referència (aquest aspecte serà descrit en un punt posterior).

Les següent dades que s'associen a una ruta estan encapsulades dins el que s'anomena *track*. Un *track* és un arxiu que té gravades les coordenades GPS. Aquest s'aconsegueix en el moment de enregistrar la ruta, ens acompanya un lector GPS que ens va capturant les coordenades (latitud i longitud) i l'alçada de forma discreta, és a dir, a cada interval prèviament definit.

Com que la plataforma es pretén que segueixi el màxim possible els estàndards existents (oberts), s'ha contemplat l'existència de l'estàndard GPX (GPs eXchange) que permet l'intercanvi de dades GPS entre diverses aplicacions. Està basat amb XML (eXtensible Markup Language).

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx        creator="WikiLoc        -        http://www.wikiloc.com"        version="1.1"
xmlns="http://www.topografix.com/GPX/1/1">
    <trk>
        <trkseg>
            <trkpt lat="42.615355" lon="0.766079">
                <ele>1557.528000</ele>
            </trkpt>
            <trkpt lat="42.614994" lon="0.766869">
                <ele>1556.004000</ele>
            </trkpt>
            <trkpt lat="42.613488" lon="0.768492">
                <ele>1554.175200</ele>
            </trkpt>
            ......
       </trkseg>
    </trk>
</gpx>
```

Exemple track amb format pgx

Si s'analitza el XML d'exemple, es pot observar que aquest XML simplement consta d'una sèrie de mostres capturades on apareixen els tres camps que s'han comentat prèviament (latitud, longitud i elevació). Cada element *trkpt* correspon amb un interval del temps al llarg de la ruta.

Un cop tenim els dos fitxers, i sabent que:

- Els vídeos estan formats per *frames*. Podem saber quants *frames* formen el vídeo i quants es reprodueixen per segon.
- El fitxer de tracks ha capturat les dades en intervals regulars (discret).

Si ajuntem aquestes dues premisses, s'observa que en tot moment podem mantenir una sincronització entre *frames* i coordenades on la relació s'estableix en base al temps (Exemple: si un vídeo reprodueix quinze *frames* per segon i el *track* té mostres capturades cada segon, sabem que cada quinze *frames* que es reprodueixin del vídeo impliquen un salt a les següents coordenades del *track*). Aquest sincronització entre la reproducció del vídeo i el seguiment dels valors del GPS resulta determinant per la reproducció realista de la ruta tal i com es veurà en punts posteriors.

També s'ha analitzat el format CRS. Aquest és un altre format que té la mateixa finalitat que l'anterior. Es caracteritza per estat àmpliament adoptat en el mercat ja que és el format que usen els dispositius GPS de

la marca GARMIN (un dels majors fabricants de GPS a nivell mundial).

```
<?xml version="1.0" encoding="UTF-8"?>
<TrainingCenterDatabase
xsi:schemaLocation="http://www.garmin.com/xmlschemas/TrainingCenterDatabase/v1
http://www.garmin.com/xmlschemas/TrainingCenterDatabasev1.xsd"
xmlns="http://www.garmin.com/xmlschemas/TrainingCenterDatabase/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Courses>
        <CourseFolder Name="Courses">
            <Course>
                <Name>Pedals de foc</Name>
                <Track>
                    <Trackpoint>
                        <Time>2009-12-12T09:05:04.167-07:00</Time>
                        <Position>
                          <LatitudeDegrees>42.615355</LatitudeDegrees>
                          <LongitudeDegrees>0.766079</LongitudeDegrees>
                        </Position>
                        <AltitudeMeters>1557.528</AltitudeMeters>
                    </Trackpoint>
                    <Trackpoint>
                        <Time>2009-12-12T09:05:05.167-07:00</Time>
                        <Position>
                          <LatitudeDegrees>42.614994</LatitudeDegrees>
                          <LongitudeDegrees>0.766869</LongitudeDegrees>
                        </Position>
                        <AltitudeMeters>1556.004</AltitudeMeters>
                    </Trackpoint>
                        <TrackPoint>....
                </Track>
            </Course>
        </CourseFolder>
    </Courses>
</TrainingCenterDatabase>
```

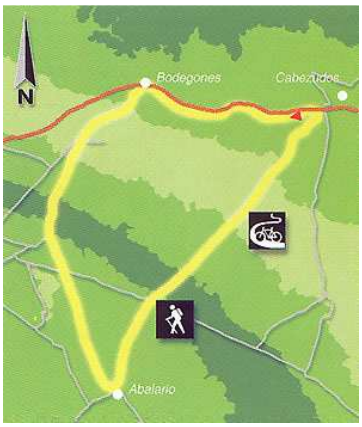
Exemple track amb format CRS

Tal com es pot observar, en l'essència de les dades contingudes en el exemple son les mateixes en ambdós casos. En una fase posterior, l'anàlisi i disseny de la plataforma, es decidirà quin és l'estàndard que finalment s'implementa.

Un altre dels fitxers que composen la ruta és el XML d'informació on es desaran les dades associades a la ruta que cal mostrar en la interfície de la plataforma (modalitat off-line principalment, en la modalitat on-line, aquestes dades s'actualitzen amb les del servidor). Aquest XML conté informació com ara: codi de la ruta, nom de la ruta, localització, descripció, longitud, dificultat, puntuació dels usuaris, ...

Finalment, hi ha dos fitxers més que permetran l'ús d'una miniatura en la realització de la ruta. Es tracta

d'una imatge amb la vista aèria de la ruta on es dibuixa el traçat. En el següent esquema es pot observar la imatge d'una miniatura a tall d'exemple:



A més, es generarà un fitxer XML que contindrà el número de *frame*, la coordenada X i la coordenada Y de la imatge. D'aquesta forma, quan l'usuari es trobi visualitzant un *frame* concret es podrà inserir una marca en les coordenades corresponents sobre la miniatura per tal que l'usuari vegi en quina posició es troba. En el següent quadre es pot veure una mostra:

```
<?xml version="1.0" encoding="UTF-8"?>
<imatge>
    <ruta>
        <frame num="1">
                <x>15</x>
                <y>32</y>
        </frame>
        <frame num="2">
                <x>23</x>
                <y>38</y>
        </frame>
        <frame num="3">
                <x>27</x>
                <y>45</y>
        </frame>
        <frame num="4">
                <x>31</x>
                <y>54</y>
        </frame>
        .....
    </ruta>
</imatge>
```

Exemple de coordenades o frames d'una miniatura.

Per tal de gestionar millor l'intercanvi i transferència de fitxers de rutes, en l'aplicació s'estableix un paquet que integra els diferents fitxers parcials. Aquest fitxer té una extensió *fw*. Per tant, el fitxer resultant està

composat per diferents parts:

Capçalera d'informació (XML d'informació)
Vídeo
Track
Imatge miniatura
XML de la miniatura

Estructura del nou format .fw

Ara bé, quan l'aplicació instal·la la ruta compactada en un únic fitxer, aquesta es desglossa en fitxers i s'ubiquen en diferents parts del directori de fitxers de l'aplicació. El motiu per el qual es fragmenta en diferents parts és per millorar el tractament ja que d'aquesta forma es poden manipular diferents fitxers am mateix temps.

3.4. Requeriments no funcionals

Comunicació Rodet – Unitat de Control

Primerament mostrarem en la següent imatge el sistema que es vol muntar:

Tal i com es pot veure en la imatge, la plataforma consta de 3 components físics:

- *Rodet*: El rodet que utilitzem és de la casa TACX; es tracta d'una empresa que es dedica a la comercialització de plataformes d'entrenament per a ciclistes, però estan especialitats en ciclisme de carretera (aquest és el punt que la nostra plataforma vol aprofitar i captar el mercat de ciclisme de muntanya). El rodet està basat en un fre electromagnètic que pot ser accionat sota demanda a tra-

vés de la unitat de control. Està construït sobre una estructura que permet recolzar la bicicleta mitjançant l'eix de la roda del darrera d'aquesta.

Esquema del rodet

- Unitat de control: La unitat de control es un dispositiu electrònic que es responsabilitza de enviar i rebre dades des del rodet a l'ordinador i a l'inrevés. Conté els botons per tal que l'usuari interactuï amb l'aplicació. Enviament de dades: el descodificador del rodet envia una senyal a cada volta del rodet, aquesta senyal és recuperada per la unitat de control i enviada a l'ordinador que convertirà la senyal en velocitat. Recepció de dades: la unitat de control rebrà de l'ordinador la magnitud de frenada a aplicar al rodet segons la pendent de la ruta. Aquesta magnitud és un valor de 0 a 9. Un cop rebuda la informació, l'envia al rodet per fer actuar el fre electromagnètic. Contindrà 5 tecles. 4 tecles de direcció: amunt, avall, dreta i esquerra, i una tecla per confirmar. Amb les tecles de direcció es podrà desplaçar per les opcions de la pantalla, amb la tecla de confirmació s'accionarà l'element seleccionat.

Unitat de control

- PC o computadora: es descriurà en el punt posterior de requisits tecnològics.

Ordinador

Aquests dos components han d'estar connectats entre ells mentre l'aplicació està en execució.

Obtenció de la velocitat

La obtenció de la velocitat requereix un estudi de les dades que envia el rodet per tal d'analitzar quin és el tractament que s'ha d'aplicar sobre aquestes. Aquest anàlisi es realitzarà de forma experimental per posteriorment extreure les conclusions.

Càlcul del desnivell

Per calcular el desnivell de la posició actual en la que es troba l'usuari es necessita el fitxer track que conté les dades de geoposicionament. D'aquesta forma, en tot moment sabem en quina posició es troba l'usuari. L'Aspecte més important que requereix el càlcul de la pendent és el fet que la ruta està preestablerta, per tant podem fer prediccions del que vindrà a continuació.

La estratègia consisteix en:

- Tenint com en compte el punt actual, sabem les coordenades i la elevació.
- Sabem la elevació que vindrà a continuació.

Un cop tenim aquestes dades definides, podem saber exactament quina serà la diferència en al elevació aplicant la següent fórmula:

$$desnivell_i = elevació_{i+C} - elevació_i$$
$$pendent = desnivell_i * 100 / distància$$

On es pot veure que la pendent en el instant i és la diferència entre l'elevació a un instant posterior (on l'offset ve definit per la constant C que s'ha de determinar en funció del rendiment) i l'elevació de l'instant actual dividit per la distància que separa els dos punts.

Un cop tenim el resultat, sabem que:

- pendent = 0: la ruta es realitza en el pla
- pendent > 0: el ciclista es troba en una pujada.
- pendent < 0: el ciclista es troba en una baixada.

Un cop és té calculat el desnivell (on els valors son percentatges), aquest valor ha de ser convertit a l'equivalent dins el rang entre 0 i 9 (graus de resistència aplicable al pneumàtic que accepta el rodet). Aquest

valor serà el que posteriorment s'enviarà al rodet per tal d'ajustar l'esforç que ha de realitzar l'usuari en funció del tram en que es trobi de la ruta. S'observa la següent taula:

Valor numèric	Desnivell (m)
0	Més petit
1	[-4 - -3]
2	[-3 - -1,5]
3	[-1,5 – -0,75]
4	[-0,75 – -0,25]
5	[-0,25 – 0,25]
6	[0,25 – 0,75]
7	[0,75 – 1,5]
8	[1,5 – 3]
9	Més gran

Si ens adaptem a aquesta taula, veurem que l'usuari té una resistència inicial en el tram horitzontal. En cas de trobar-se en un pendent de pujada, s'augmenta la resistència simulant l'esforç addicional que s'ha de realitzar per pujar. En cas de ser una baixada, es redueix el grau de resistència de forma que el ciclista tingui la sensació d'estar en una baixada.

Tot i la taula, els rangs hauran de ser ajustats de forma experimental.

Càlcul de la velocitat de reproducció de la ruta

Un altre càlcul important que s'ha de tenir en compte en el desenvolupament de l'aplicació és el càlcul de la velocitat de reproducció del vídeo de la ruta. L'objectiu és que el vídeo no es reprodueixi massa ràpid o massa lent i s'ajusti a la velocitat de pedaleig de l'usuari.

Per aquesta raó, es parteix del principi que la ruta ha estat enregistrada a una velocitat constant i que aquesta és coneguda. De forma que la reproducció a 1X (reproducció normal) es correspon amb la velocitat d'enregistrament.

Per exemple: *si una ruta ha estat enregistrada a una velocitat de 20 Km/h i el ciclista pedaleja a una velocitat de 20 km/h; llavors el vídeo de la ruta es reprodueix a una velocitat de 1X.*

Ara bé, si l'usuari accelera o redueix la velocitat de pedaleig, la velocitat de reproducció es redueix o s'accelera de forma proporcional mitjançant una regla de tres. En l'exemple es veurà clarament:

Si l'usuari que anava a 20 km/h accelera i es posa a una velocitat de 30 km/h, la velocitat de reproducció del

vídeo es calcula com:

$v_1 \rightarrow$ 20 Km/h

$v_2 \rightarrow$ 30 km/h

$F_{rep\_v1} = v_2 / v_1 = 30 / 20 = 1.5\ F_{rep\_v1}$

De forma que el vídeo es reprodueix més ràpid simulant l'augment de la velocitat (la ruta acabarà abans).

Per cas contrari, si decideix frena i anar més lent reduint la velocitat de 20 km/h a 10 km/h, l'aplicació ajustarà la velocitat de reproducció amb:

$v_1 \rightarrow$ 20 Km/h

$v_2 \rightarrow$ 10 km/h

$F_{rep\_v1} = v_2 / v_1 = 10 / 20 = 1.5\ F_{rep\_v1}$

De forma que el vídeo es reprodueix més lentament simulant la reducció de la velocitat (la ruta acabarà més tard)

Càlcul de les calories consumides

El càlcul de les calories consumides és una funcionalitat addicional de la plataforma que ofereix un valor aproximat. Sabem que les calories consumides es calculen en funció de la activitat, el pes del usuari, el temps que es realitza l'activitat i la velocitat a la que es realitza.

En el nostre cas, l'activitat que es realitza és el ciclisme. Per tant tenim la taula següent:

Activitat	50 kg	60 kg	70 kg	80 kg	90 kg	100 kg	110 kg	120 kg
Ciclisme 14 km/h	5	6	7	8	9	10	11	12
Ciclisme 20 km/h	8	9,6	11,2	12,8	14,4	16	17,6	19,2
Ciclisme 8 km/h	3,2	3,84	4,48	5,12	5,76	6,4	7,04	7,68

On les unitat estan en Kilocalories per minut.

Per tant, l'aplicació haurà d'anar comptant les kilocalories consumides mitjançant la següent estratègia:

1. Obtenir la velocitat mitjana a cada minut.
2. Cercar quina és la fila que més s'apropa a la velocitat (la taula ha de ser ampliada).

3. Obtenir el valor segons el pes de l'usuari (el pes es troba a la fitxa del perfil).
4. Sumar el valor de kilocalories per minut al comptador acumulatiu.

D'aquest forma, cada minut es refrescarà les calories consumides al llarg de la ruta.

Càlcul de les distàncies

Una de les funcionalitats que té l'aplicació és el càlcul de la distància recorreguda per l'usuari. Aquest càlcul s'ha de realitzar constantment al llarg del recorregut de la ruta. A més, cal portar el control del punt on ens trobem (en kilòmetres).

Per desgràcia, l'aplicació no disposa directament de les dades de distància. Recordem que les dades de geoposicionament estan formades per la latitud, la longitud i la elevació. Per tant, cal realitzar una càlculs previs per obtenir la distància.

Per tal de poder calcular la distància entre els dos punts indicats amb latituds i longitud, s'ha de fer una conversió al sistema UTM (*Universal Transverse Mercator*, UTM) ja que aquestes no expressen el punt en angles sinó que l'expressen en metres a nivell del mar. Per tant, si treballem amb metres, podem calcular la distància entre dos punts usant la fórmula de punt final menys punt inicial.

Si en aquest procés hi afegim les dades de l'elevació en el dos punts (calculant la diferència) llavors podem aplicar el teorema de Pitàgores per obtenir la distància exacta que ha recorregut l'usuari.

Finalment, el procés simplement consisteix en anar acumulant aquest valor.

Observació: cal estudiar la conversió de coordenades latitud - longitud ja que el globus terrestre està dividit en zones i s'han de determinar segons on ens trobem.

Plataforma d'explotació

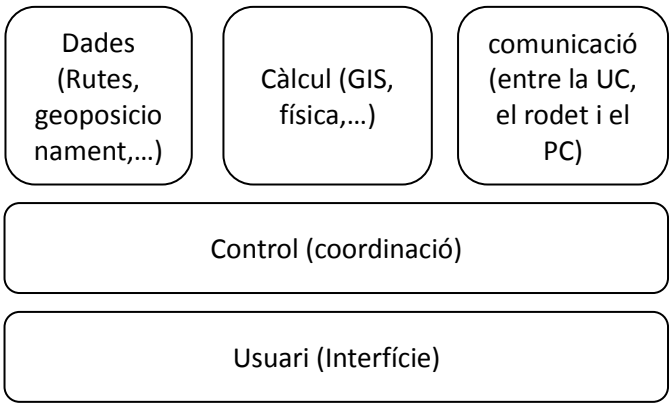
En la present fase del projecte, la plataforma d'explotació no està definida ja que s'han de prendre decisions sobre el llenguatge d'implementació. La intenció és crear un software que utilitzi un entorn de codi obert (Open Source), que no requereixi l'ús de llibreries de software privat. A més, es valorarà la independència del Sistema Operatiu, de forma que pugui ser instal·lat àmpliament estalviant el cost de les llicències addicionals.

Plataforma de desenvolupament

La decisió sobre la plataforma de desenvolupament es troba en fase d'estudi a falta de valorar les possibilitats.

Arquitectura

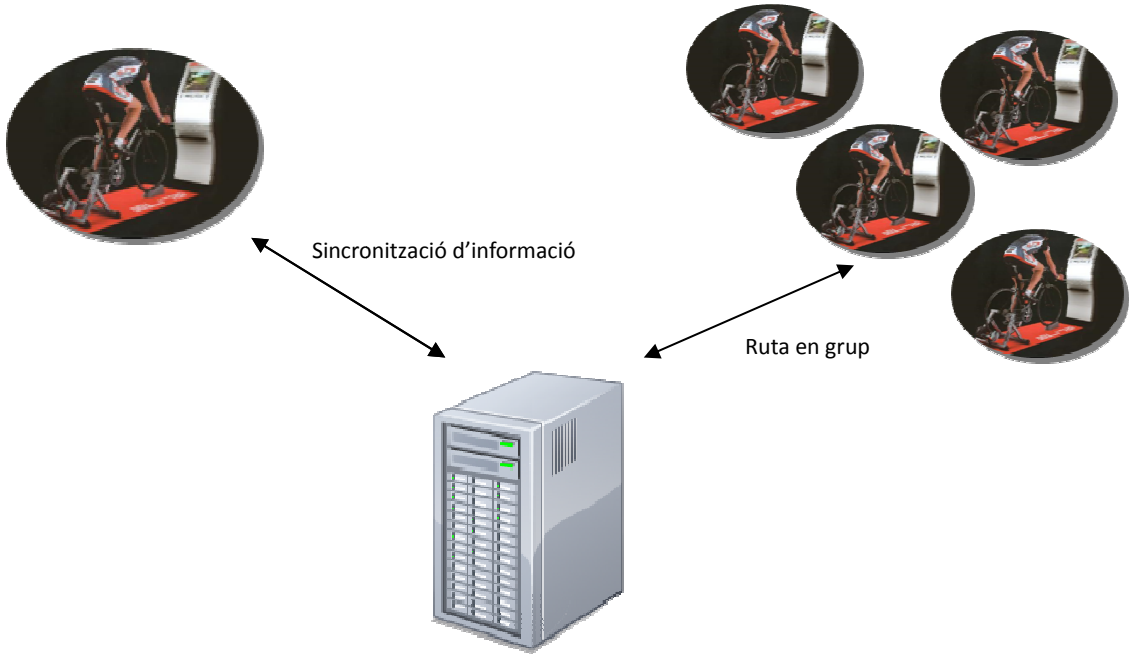
L'arquitectura de la plataforma es pot veure en el següent esquema:



Tal i com es pot veure en l'esquema, l'aplicació treballa coordinant tres blocs base: les dades, el càlcul i la comunicació. A més, hi ha el bloc de sincronització i coordinació que permet que els 3 blocs previs puguin interactuar entre ells.

Finalment hi ha l'usuari o interfície que mantindrà una comunicació amb el control sol·licitant informació o indicant instruccions que les capes anteriors realitzaran.

Aquesta arquitectura es correspon amb la plataforma local. Però d'altra banda, també hi ha una arquitectura client –servidor. De forma que entre ells es puguin intercanviar informació o realitzar les rutes conjuntament tal i com s'ha descrit prèviament.



Model del sistema

El model de la plataforma es correspon amb el model de programació de tres capes:

- Capa de presentació: és la capa que visualitza l'usuari, presenta la informació i captura la informació que l'usuari introdueix amb un procés de càlcul mínim. Aquesta capa es comunica exclusivament amb la capa de negoci. És coneguda com interfície gràfica. Equival al bloc Usuari vist prèviament.
- Capa de negoci: és on resideix tota la operativa de l'aplicació. Rep i resol les peticions de l'usuari i envia els resultats a la capa de presentació. Es comunica amb la capa de presentació per rebre les peticions i enviar els resultats, i amb la capa de dades per obtenir la informació que necessita en els càlculs. És equivalent als blocs de comunicació, càlcul i sincronització vistos prèviament.
- Capa de dades: és on resideixen les dades i la capa encarregada d'accedir a ells. En aquesta plataforma, està composada per els XMLs, els fitxers que composen les rutes, ...

Eficiència

Com que es tracta del desenvolupament d'un simulador, es podria considerar que el principal criteri d'eficiència que cal tenir en compte en l'aplicació és la velocitat de càlcul (simplicitat de la complexitat i temps d'operació) ja que la resposta ha de ser pràcticament a temps real.

Afortunadament, tenim la avantatge de poder-nos anticipar en algunes operacions ja que coneixem la ruta, però en altres casos, cal un funcionament de resposta ràpida en temps real.

Seguretat

L'aplicació utilitza el concepte d'usuaris, per tant, cal garantir les sessions d'usuari. També apareix el concepte d'arquitectura client – servidor. Això implica un intercanvi de dades amb un servidor remot.

En l'aspecte dels usuaris a nivell local, la protecció de les sessions es realitzarà mitjançant el xifratge d'informació a partir de la clau que introdueixi l'usuari. Per tant, les dades dels diferents usuaris estaran xifrades.

La connexió amb el servidor es realitzarà mitjançant una autenticació d'usuari mitjançant un nom d'usuari i contrasenya.

Cal analitzar protocols de comunicació segurs i algorismes de xifratge de dades.

REQUERIMENTS TECNOLÒGICS

Els requeriments tecnològics de l'aplicació no son massa exigents en quan a processador, però si que és important altres aspectes com ara la memòria RAM o la connexió a Internet. Per posar un sistema base mínim, es considera:

- Pentium 4 a 1.6 GHz
- 2 Gb de RAM
- Disc dur: en funció del nombre de rutes, que és el que realment consumirà l'espai.
- Port USB: per la connexió de la Unitat de Control.
- Connexió de banda ample a Internet per la obtenció de dades en temps reals (54 Mbps)

El Sistema Operatiu i el software addicional va en funció de la plataforma de desenvolupament que es triï finalment.

4. Arquitectura de la solució

Aquest capítol pretén mostrar al lector quines son les decisions que s'han anat prenent al llarg del desenvolupament de la plataforma i que s'haurà d'implementar per tal de dur a la pràctica el prototip. Inicialment es fa una recerca tecnològica on s'analitzen diferents components que permetrien realitzar tasques requerides per el prototip. En segon terme, es descriu la base de dades que usarà l'aplicació. Finalment, es podrà veure el disseny complet del prototip.

Cal indicar al lector que a l'annex del present document es veurà una descripció completa de les classes i mètodes que requereix la implementació de la solució.

4.1. Recerca tecnològica

Aquest apartat està destinat a mostrar un anàlisi de les diferents tecnologies que s'han estudiat per tal d'aplicar-les al desenvolupament de la solució final. Es pretén mostrar una explicació de les diferents opcions que s'han tingut en compte i justificar el motiu per el qual han estat acceptades o descartades en la implementació final.

4.1.1. Reproductor de vídeo

El reproductor de vídeo ha estat un dels elements que més a condicionat el desenvolupament de la plataforma ja que constitueix el nucli de la mateixa. Per triar entre unes llibreries o altres, calia que es complís un requisit imprescindible: s'havia de poder variar la velocitat de reproducció, és a dir, el nombre de frame per segon (*frame rate*) de forma suau. Aquest ha estat un factor que ens ha condicionat a una única tria possible.

Primerament presentem una breu introducció de les llibreries que s'han contemplat:

- **PyGAME:** es tracta d'un conjunt de mòduls escrits en Python que estan orientats al desenvolupament de videojocs d'una forma senzilla. Es va tenir en compte ja que permet el desenvolupament d'interfícies d'usuari i disposa de components multimèdia. Es va haver de descartar ja que la versió actual no disposa d'un reproductor de vídeo per la versió de Windows (només està disponible per els sistemes UNIX)(http://www.pygame.org).
- **Phonon:** es tracta d'un marc multimèdia estàndard en el entorn KDE. Es tracta d'una capa intermèdia que facilita l'accés de les aplicacions als motors multimèdia. QT l'ha adoptat per disposar de components multimèdia, el que ha facilitat la seva exportació a múltiples plataformes. No ha estat seleccionat a causa de no permetre la modificació de la velocitat de reproducció d'una forma flexible (http://phonon.kde.org/).
- **PyMedia:** es tracta d'un framework multimèdia escrit en Python que ràpidament es va descartar a

causa de trobar-se actualment obsolet (http://pymedia.org/).

- **DirectShow:** es tracta d'un marc multimèdia desenvolupat per Microsoft (antigament era propietat de DirectX). Ofereix una API que permet un accés més fàcil a les funcions del sistema que accedeixen els recursos multimèdia.

Finalment em vaig decidir per l'ús de les llibreries de DirectShow per els següents motius:

- Variació de la velocitat: de totes les llibreries que s'han analitzat, DirectShow ha estat la única que ens ha permès variar la velocitat de reproducció amb la màxima flexibilitat adaptant-se perfectament al nostra propòsit.
- Programació en llenguatge C++: la API del DirectShow està dissenyada per accedir-hi directament usant el llenguatge C++ (evitant l'ús de middleware). Aquest aspecte ens va bé de cara al desenvolupament del projecte ja que ens permet usar un llenguatge compilat (més velocitat en la execució que un llenguatge interpretat); aquest aspecte és important per tal de poder executar l'aplicació el més proper possible al temps real (evitant retards en l'execució de les iteracions produïts per la interpretació del codi).

Per contra, també presenta alguns inconvenients:

- Multi plataforma: l'aplicació que es desenvoluparà deixarà de ser multi plataforma (una de les característiques que es volia tenir en compte) ja que ens centrem en els sistemes Windows. Analitzant la situació, s'ha valorat com un inconvenient menor ja que la majoria d'usuaris potencials de l'aplicació son usuaris domèstics sense coneixements profunds d'informàtica, per aquesta raó, el seu sistema operatiu és el Windows.

4.1.2. Interfície

Un altre aspecte que s'ha de desenvolupar és el de la interfície d'usuari. Tal com s'ha comentat, l'aplicació ha d'oferir un aspecte de videojoc i s'ha d'interactuar amb ella mitjançant una unitat de control. Per poder realitzar la interfície, s'han considerat una sèrie de frameworks per facilitar el desenvolupament. Algun d'ells son:

- **QT:** es tracta d'una biblioteca multi plataforma per el desenvolupament d'interfícies gràfiques d'usuari. QT utilitza el llenguatge de programació C++ de forma nativa, tot i que pot ser usat en varis llenguatges mitjançant l'ús de *bindings*. Es va descartar a causa de la costosa integració del DirectShow en el components del QT (com a llibreria multimèdia implementa les llibreries Phonon).
- **Microsoft XNA (XNA's Not Acronymed):** és un conjunt d'eines amb un entorn d'execució administrat que subministra Microsoft. Està destinat a facilitar el desenvolupament de videojocs. Amb aquesta eina es va considerar que es podria desenvolupar una interfície gràfica més pròxima a

la dels videojocs, però va aparèixer l'inconvenient del llenguatge de programació ja que només es permet l'ús del C#.

- **Ogre 3D (Object-Oriented Graphics Rendering Engine):** es tracta d'un motor de renderitzat 3D orientat a escenes escrit amb el llenguatge de programació C++. Les seves biblioteques faciliten l'ús de les llibreries OpenGL i Direct3D i ofereixen una interfície basada en objectes i classes d'alt nivell. Es tracta de software lliure. Cal dir que disposa d'extensions que permeten la obtenció de menús d'estil videojoc. Es va descartar degut a la impossibilitat de reproduir un vídeo.
- **Windows Form:** plataforma bàsica per el desenvolupament d'aplicacions d'escriptori en entorns Windows.
- **MFC (Microsoft Foundation Classes):** és un conjunt de classes que proveeixen un accés més senzill a les API de Windows. Van ser introduïdes per Microsoft el 1992 i, des de llavors, han anat apareixent noves versions a mesura que evolucionava l'entorn de programació Visual C++.

Finalment em vaig decidir per l'ús de les MFC degut a: es tracta del desenvolupament d'un prototip i no es volia destinar excessius esforços en formar-me en noves tecnologies quan tenia un domini previ de les MFC. A més, em permet la integració de components *DirectShow* d'una forma senzilla.

El Windows Forms ofereix uns avantatges molt similars al MFC, però molt més senzill. Es va descartar a darrera hora ja que implicava l'ús del *framework* .NET enlloc de realitzar les crides directament al sistema win32.

4.1.3. Llenguatge de programació

Una altra decisió important que s'ha hagut de prendre a llarg del desenvolupament del projecte és la del llenguatge de programació en que s'implementarà la solució. Les característiques que presenta cada llenguatge influeixen de forma considerable en la solució final. Per aquest motiu s'han tingut en compte les següents opcions:

- **C/C++:** llenguatge de propòsit general amb mecanismes que permeten la manipulació d'objectes. Es tracta d'un llenguatge del paradigma de la programació orientada a objectes. Cal dir que és un llenguatge compilat. Per tant, els seus punts forts son la velocitat d'execució i la extrema flexibilitat que deixa tant a baix com a alt nivell. Per contrapartida, presenta el problema de que els desenvolupaments son més lents ja que no existeixen tants automatismes com en els llenguatge d'alt nivell.
- **C#:** es tracta d'un llenguatge de programació orientat a objectes desenvolupat i estandarditzat per Microsoft com a part de la plataforma .NET. Es tracta d'un llenguatge interpretat (requereix una màquina virtual per executar les aplicacions) que permet un desenvolupament àgil d'aplicacions d'escriptori. El seu punt fort és la rapidesa d'implementació de les solucions; però en contra té la gran dependència a entorns Microsoft i és un llenguatge lent en execució.

- **Python:** es tracta d'un llenguatge d'alt nivell, la filosofia del qual posa especial atenció en una sintaxi neta que afavoreixi la llegibilitat del codi. Es tracta d'un llenguatge de programació multi paradigma ja que permet la programació orienta a objectes, la imperativa i, de forma limitada, la funcional. Els avantatges que presenta son la multi plataforma i la senzillesa de la sintaxi; però té l'inconvenient de la velocitat d'execució ja que també és un llenguatge interpretat.
- **Java:** es tracta d'un llenguatge orientat a objectes desenvolupat a principis dels anys 90. El llenguatge agafa molta sintaxi de C i C++, però té un model d'objectes més simple i elimina les eines de baix nivell que solen induir a errors com ara la manipulació directa d'apuntadors o memòria. Es tracta d'un llenguatge interpretat que requereix que el sistema disposi de la màquina virtual de Java JVM. El llenguatge presenta un inconvenient que el va descartar automàticament: les aplicacions d'escriptori de Java son extremadament lentes.

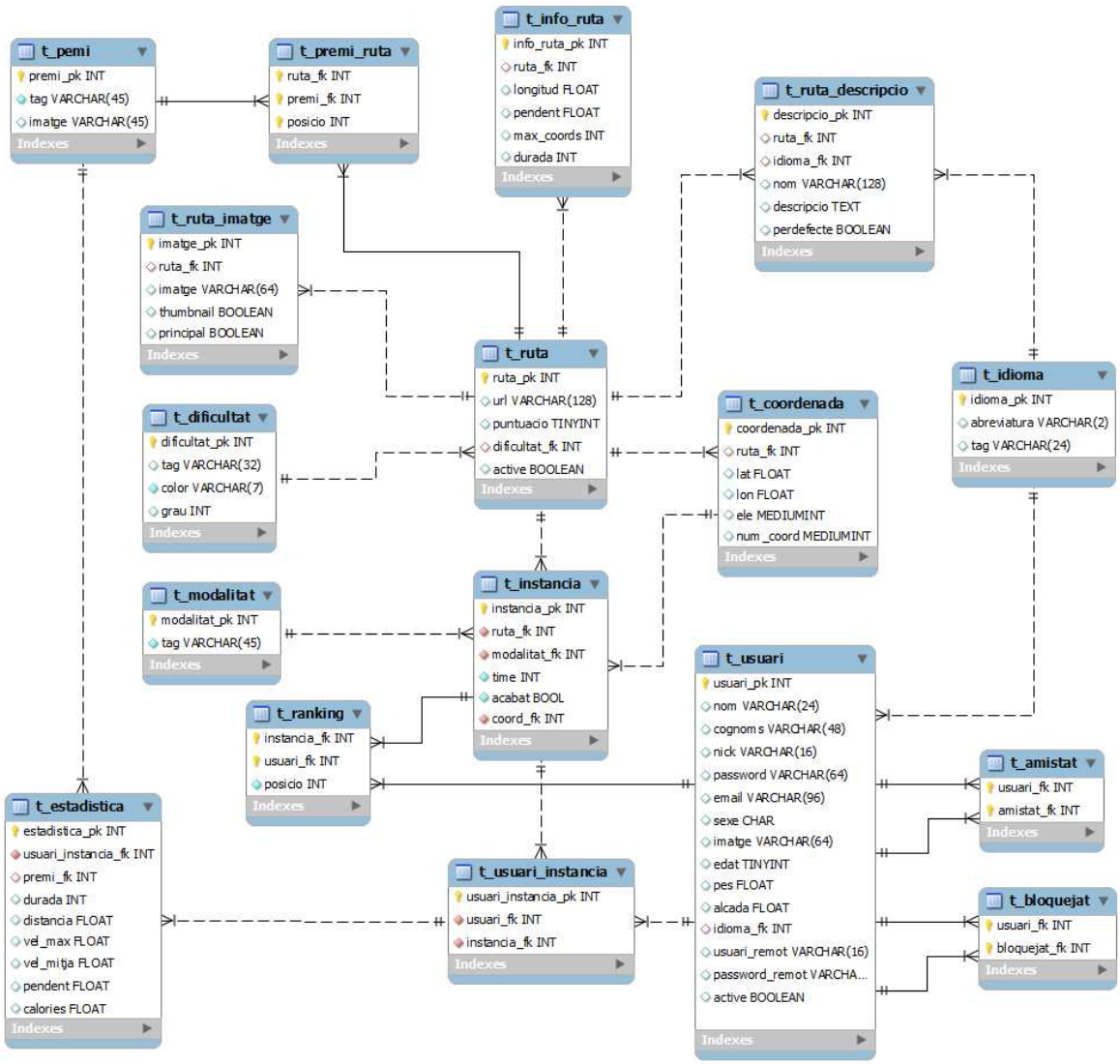
Finalment, i tenint en compte la selecció del DirectShow, s'ha decidit implementar la solució amb C++ ja que ens permet una alta velocitat d'execució, control absolut de l'aplicació i un accés directa la API de Windows per controlar els components multimèdia i els ports sèrie requerits per l'aplicació. D'alta banda, també s'evita l'ús innecessari de frameworks i bindings entremitjos que suposarien retards en l'execució.

4.2. Base de dades

El prototip que es pretén realitzar, necessita gestionar diferents tipus de dades per el correcte funcionament. Per emmagatzemar aquestes, he triat el gestor de base de dades SQLite per el següents motius:

- Transaccions de base de dades atòmiques, consistència de dades, aïllament i durabilitat (ACID).
- Lleuger, només requereix una única biblioteca (ideal per aplicacions amb bases de dades incorporades).
- Velocitat en les operacions.
- Portabilitat a múltiples plataformes.
- Es tracta d'un motor de base de dades que pertany al domini públic.

La base de dades que requereix l'aplicació es pot veure en el següent diagrama:



En la base de dades existeixen una sèrie de taules que representen diccionaris que contenen dades base que usa l'aplicació (la avantatge de tenir aquestes dades a la base de dades és que permet una ràpida ampliació):

- t_premi: conté el títol i la imatge dels diferents trofeus que es poden obtenir en la realització de les competicions.
- t_dificultat: conté els nivells de dificultat que contempla l'aplicació. Hi ha l'atribut grau, que indica l'ordre de dificultat, i l'atribut color que indica el color que el representa en format #RRGGBB.
- t_modalitat: fa referència a les diferents modalitats amb les que es permet la realització de les rutes.
- t_idioma: conté les dades dels idiomes que va acceptant l'aplicació. És utilitzat per les preferències dels usuaris i per la classificació de les descripcions.

Per altra banda, la base de dades conté una sèrie de taules que representen les entitats bàsiques requerides pel funcionament de l'aplicació:

- t_usuari: aquesta taula recull les dades bàsiques que descriuen els usuaris.

- t_ruta: aquesta taula recull les dades bàsiques que defineixen una ruta.
- t_info_ruta: aquesta taula recull informació numèrica (quantitativa) de la ruta.
- t_ruta_imatge: taula que conté les imatges de les diferents rutes.
- t_ruta_descripcio: taula que conté les descripcions de les rutes, que poden estar en diferents idiomes.
- t_coordenada: aquesta taula recull les coordenades que composen el recorregut d'una ruta (track).
- t_instancia: aquesta taula conté les dades que defineixen una instància d'una ruta, és a dir, la realització de la ruta.
- t_estadistica: cada usuari que realitza una ruta (instància) té una entrada en aquesta taula on es registren les estadístiques de la realització.

Finalment, hi ha una sèrie de taules que permeten la relació entre entitats base o entitats base amb taules diccionari. Aquestes son:

- t_premi_ruta: aquesta taula conté una relació entre les rutes i els premis. És a dir, per cada ruta, ens diu quin premi correspon a les diferents posicions.
- t_ranking: ens indica en quina posició va quedar cada usuari al realitzar una ruta (instància d'una ruta).
- t_usuari_instancia: aquesta taula ens indica quins usuaris han realitzar quina ruta (instància d'una ruta).
- t_amistat: aquesta taula conté una relació entre dos usuaris per tal d'indicar que aquests son amics.
- t_bloquejat: similar a la taula anterior, conté una relació entre els usuaris per indicar que l'usuari X ha bloquejat a l'usuari Y.

4.3. Disseny de la plataforma

SISTEMA HARDWARE - UC - PC

Visió general del sistema

Aquest sistema consisteix en el desenvolupament de la comunicació que hi ha d'haver entre els diferents components hardware que requereix la plataforma. Tal com s'ha descrit en el document de requeriments, la plataforma disposa de diferents components hardware que cal connectar entre si per el correcta funcionament de l'aplicació (la interconnexió serà descrita en la descripció de la arquitectura del sistema).

Un cop s'ha muntat el sistema, cal establir uns protocols de comunicació per tal que els diferents elements es puguin entendre. El protocol es definirà en l'apartat d'interfícies del present sistema.

La funció general d'aquest sistema és el de transferir la informació del rodet al software del PC i del PC

donar les ordres al rodet.

Arquitectura del sistema

L'arquitectura del sistema es composa de diferents parts: el rodet, la unitat de control y el PC.

La connexió entre els rodet i la unitat de control es realitza mitjançant un cable Ethernet. La connexió entre la unitat de control i el PC es realitza mitjançant un cable sèrie (amb adaptadors USB). La connexió amb l'ordinador es realitza a través d'un port COM sèrie virtual amb les següents propietats:

- 9600b/s de velocitat (Bauds)
- 8 bits de dades
- Sense paritat
- 1 bit de STOP

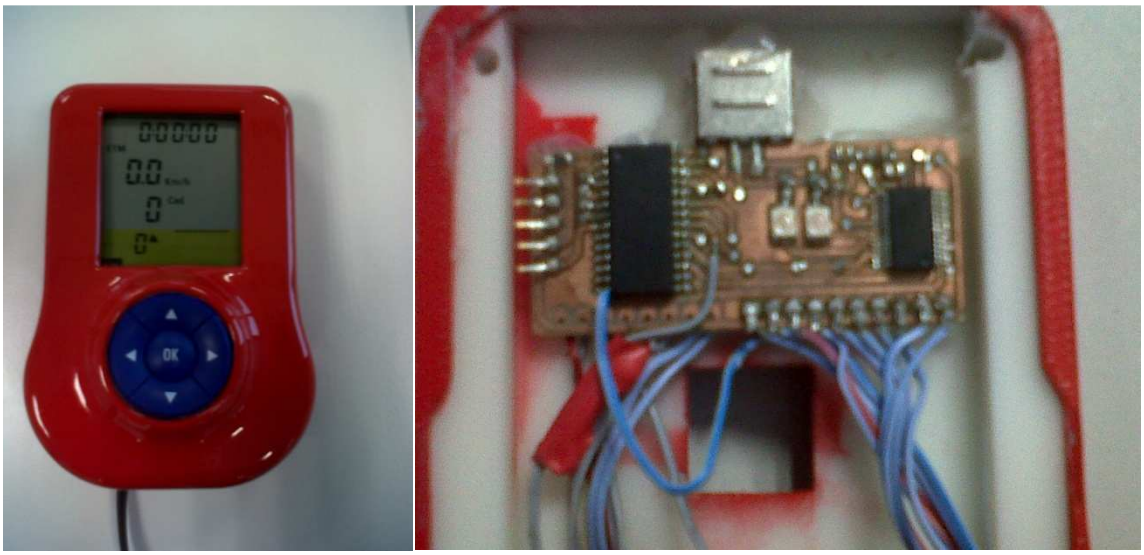
La Unitat de control està basada en aprofitar un mòdul existent en el mercat que permet la obtenció de la velocitat del rodet. Sobre aquest mòdul s'ha afegit una nova capa hardware que ens permet la comunicació amb l'ordinador, l'enviament de dades entre els PC i el rodet i, finalment, afegir el teclat de navegació.

La descripció del disseny electrònic queda fora del projecte ja que és un projecte d'enginyeria informàtica, la tasca realitzada és la programació del microcontrolador. El que si que enumeraré serà els components que s'han integrat:

- Microcontrolador PIC 16F883 (fabricant: Microchip, www.microchip.com): en aquest microcontrolador he implementat el codi que permet l'enviament i recepció de les dades. Es programa en C o Assembler, on he triat la implementació en C. L'entorn de desenvolupament és el Mplab IDE.
- Inclou un integrat FTDI, que permet fer la conversió de USB a UART (**U**niversal **A**synchronous **R**eceiver-**T**ransmitte). Aquest permet a l'ordinador comunicar-se amb el rodet mitjançant un port COM virtual com a interfície.
- La comunicació del microcontrolador amb el rodet es realitza mitjançant ponts en les connexions existents en el mòdul original per obtenir els valors del rodet i enviar-li les dades.
- El teclat que s'ha afegit és el *Navimec with multimec 3A/3F* (*Navimec series*). La URL del fabricant és http://www.mec.dk/switches/pushbutton_switches/round_pushbutton_switches/navimec/multimec

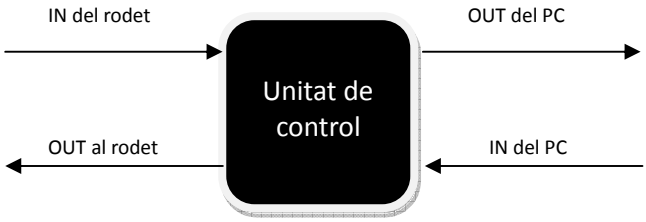
La pantalla que té la unitat de control és la del mòdul original. En cap moment s'ha programat aquest component.

En la següent imatge es pot veure el hardware de la unitat de control:



Interfícies

La unitat de control actua com una caixa negre que disposa de dues entrades i dues sortides (hi ha una comunicació bidireccional. La següent imatge mostra un esquema del sistema:



Per tant, la Unitat de control conté una sèrie de funcions que llegint unes entrades, tracta els valors i envia els resultats a la sortida corresponent.

Funcionalitat

La unitat de control ha de implementar les següent funcionalitats:

Enviament del token:

Quan s'executa l'aplicació de la unitat de control, aquesta ha d'enviar constantment una senyal per tal de poder ser identificat; és a dir, per tal que l'aplicació del PC pugui detectar en quin port està connectat el dispositiu.

En arrancar l'aplicació, aquesta ha d'enviar una sèrie de bytes i esperar la resposta uns instants. Si no es rep

una resposta (o no es rep la resposta esperada), torna a enviar el token. En cas de rebre la resposta correcta, ha de saltar a la funcionalitat d'inicialitzar el dispositiu.

El token és la trama formada per: 0xA1B2 (1010 0001 1011 0010)
La resposta és el complement a 1: 0x5F4D (0101 1110 0100 1101)

El codi del token ha estat triat seguint el criteri d'usar una combinació on apareguessin 1s i 0s de forma aleatòria (evitant les combinacions bàsiques de tot 0s, tot 1s, ...).

Inicialització del dispositiu:
Inicialitzar la unitat de control significa posar els dispositiu en un estat neutral. Aquest es correspon en que la resistència del fre ha de tenir el valor de la resistència en el pla.

El fre pot tenir 10 valors possibles (del 0 al 9). La resistència en el pla es correspon amb el grau 5. Per ajustar el grau de la resistència, cal enviar impulsos de créixer i decréixer el grau actual. Per tal de poder garantir que el dispositiu es posa en l'estat inicial, cal suposar el pitjor cas: el grau actual és 9.

Si es dóna aquest cas, l'aplicació, per tal de garantir el resultat, ha d'enviar 9 impulsos de decreixement i 5 de creixement. D'aquesta forma, el grau resultant serà el 5 en tots els casos.

Cal ressaltar que aquesta funció es crida al rebre la resposta al token (veure funcionalitat anterior), però el programari de l'ordinador central podrà sol·licitar el reinici de la unitat de control.

L'ordre per reiniciar la unitat de control té el següent format: 0xCD (1100 1101)

A més, cal realitzar la tasca de posar el dispositiu en modalitat resistència.

Increment de la resistència:
Funció auxiliar que conté tota la rutina necessària per l'enviament d'un impuls o senyal al rodet de forma que incrementi el grau de resistència en una unitat.

L'ordre per incrementa la resistència té el següent format: 0x1F (0001 1110)

Decrement de la resistència:
Funció auxiliar que conté tota la rutina necessària per l'enviament d'un impuls o senyal al rodet de forma que decreixi el grau de resistència en una unitat.

L'ordre per reduir la resistència té el següent format: 0xAA (1010 1010)

Enviament de la velocitat:

Aquesta funcionalitat s'executa cada cop que el rodet envia el valor de la velocitat actual. L'aplicació ha de realitzar els càlculs necessaris per tal de convertir el valor en una representació comprensible. Posteriorment, aquest valor s'ha d'enviar al PC.

El format de l'enviament de la velocitat és el següent: 0xBBXXXX on les XXXX son 2 bytes que contenen el valor de la velocitat.

Enviament de la tecla polsada:

A més de la velocitat, l'aplicació ha d'enviar la tecla que l'usuari pitja. Cal que la unitat de control detecti les pulsacions i enviï la tecla correcta a l'ordinador.

El format de la trama d'enviament de la tecla pitjada és: 0xCCXX on XX és un byte que indica la tecla. Els valors possibles son:

- Amunt: 0x77
- Avall: 0x73
- Dreta: 0x64
- Esquerra: 0x61
- Confirmació: 0x20

Rutina principal:

El bucle principal de l'aplicació consisteix en:

- Enviament del token.
- Esperar fins a rebre la resposta correcte.
- Inicialitzar el dispositiu.
- Repetir:
    - Enviament de la velocitat
    - Enviament de la tecla pitjada
    - Escoltar ordre
    - Si hi ha una odre, respondre-la

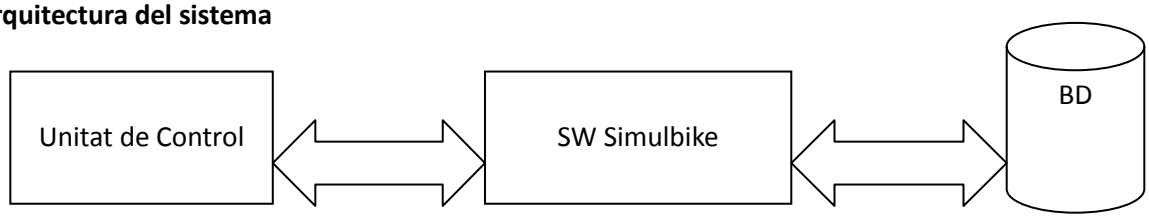
Cal destacar que detectar la tecla pitjada i detecció d'un valor de la velocitat es realitza mitjançant interrupcions.

SISTEMA COMPONENTS GLOBALS

Visió general del sistema

En aquest bloc es descriuran quin son els aspectes globals que es mantindran al llarg de tots els components de l'aplicació. Concretament, es fa referència a aquells aspectes no funcionals de l'aplicació però que s'han de definir per tal que l'aplicació en faci ús. Aquests poden ser: els formularis, l'ús de la unitat de control com a element per interactuar amb la interfície o la base de dades.

Arquitectura del sistema



Tal com es pot veure, estem definint com treballarà l'aplicació amb els elements principals d'adquisició de les dades. Per un costat, hi ha la comunicació entre la unitat de control i el software que es realitza mitjançant l'ús d'un port sèrie. Per altra banda, hi ha la base de dades (SQLite en el nostra cas) des d'on l'aplicació llegirà i escriurà dades.

Interfícies

En aquest cas no té sentit fer referència a les interfícies ja que son aspectes del funcionament de l'aplicació que son transparents a la interfície.

Funcionalitat

Els diferents aspectes que vull implementar en el prototip es poden veure descrits en la següent llista:

Accés a la Unitat de control:

Tal com s'ha comentat, l'aplicació es comunica amb la unitat de control mitjançant l'ús d'un port sèrie. Per tal de facilitar la tasca de la comunicació amb aquest darrer es crea un component de ho gestioni d'una forma transparent.

Les operacions que ha de realitzar aquest component son:

- Obrir el port sèrie: aquesta tasca ha d'obrir el port sèrie indicat aplicant la configuració també indicada.
- Tancar el port sèrie: aquesta tasca ha de tancar la connexió amb el port sèrie si aquesta es troba oberta.
- Lectura del port sèrie: aquesta tasca és la responsable d'omplir un vector de bytes de longitud indicada amb els valors que es reben a través del port sèrie.
- Escriptura del port sèrie: aquesta tasca és la responsable d'enviar a través del port sèrie el bytes que s'han escrit en el vector, cal indicar que el mètode coneix la longitud de bytes que s'han d'enviar.

Aquestes funcionalitats es troben encapsulades dins una classe *Rodet*, una instància d'aquesta classe es declara de forma global per tal que tota l'aplicació tingui accés a la mateixa i tota la aplicació treballi amb una única connexió al port sèrie.

Cal afegir que per tal de controlar el rodet, s'afegeix un flag global (active_rodet) que permet que els threads que llegeixen el port sèrie de forma repetitiva puguin ser aturats mitjançant el citat flag.

Emulació del ratolí i el teclat a partir de la unitat de control:

Tal com ja s'ha comentat, l'aplicació no fa ús del ratolí i el teclat per permetre a l'usuari interactuar amb la interfície, de forma que l'usuari ho ha de fer mitjançant la unitat de control. Com que els formularis del MFC estan preparats per respondre els events que produeixen els teclats i ratolins, s'ha pensat un mecanisme per tal de substituir aquests dispositius a favor del hardware de la plataforma.

Una primera observació que cal tenir en compte, és que la unitat de control només disposa de cinc tecles per interactuar amb la interfície; aquestes son: les quatre fletxes de direcció (dreta, esquerra, amunt i avall) i una tecla de confirmació.

Una altra observació important és que tots els elements que s'afegeixen en un formulari MFC estan identificats mitjançant un identificador únic.

La estratègia que s'ha proposat usa com a base una estructura que es defineix com:

Estructura Key_st {
id: Enter
tipus: Enumeració KeyTipus
amunt: Estructura Key_st
avall: Estructura Key_st
dreta: Estructura Key_st
esquerra: Estructura Key_st
}

Enumeració KeyTipus {
0 → tipus botó
1 → tipus camp text
}

Inicialment l'aplicació conté un apuntador de tipus Key_st que indica quin és l'element seleccionat actualment per tal de continuar a partir d'aquest.

Tots els elements que son susceptibles de ser usats o accionats per l'usuari tenen una estructura Key_st associada on s'indica l'identificador i el tipus. L'identificador és per tal de poder obtenir l'element a partir

d'aquest valor i el tipus s'utilitza per tal de poder manipular l'element correctament segons el tipus. A continuació hi ha quatre apuntadors que ens indiquen on ha de saltar el focus de l'aplicació quan es pitja cada una de les tecles. D'aquesta forma, l'aplicació pot anar seguint el recorregut de l'usuari en la interfície.

Llavors, l'algorisme que es segueix és:

- Quan es carrega el formulari, inicialitzem totes les estructures de tecles i les interelacionem.
- Es tria l'element inicialment seleccionat i es selecciona.
- Es crea una thread que va llegint les tecles que l'usuari pitja a partir de l'objecte global Rodet prèviament descrit.
- Per cada tecla:
    - Es mira si es la tecla confirmació, en aquest cas es realitza l'acció corresponent segons l'element actualment seleccionat.
    - Es mira si és direcció, en aquest cas, es mira a quin element apunta la estructura de l'element actual (si apunta a un valor nul, s'ignora la pulsació).

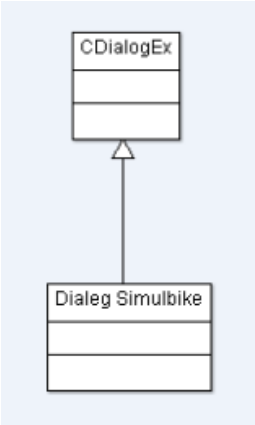
Cal afegir que a partir de l'element actualment seleccionat, es pot saber exactament quina és la acció a realitzar ja que els identificadors dels elements son únics.

Els aspectes que queden sense especificar es descriuen en el annex de la present memòria; en aquest apartat s'ha intentat mostrar el funcionament conceptual, no pas el detall tècnic.

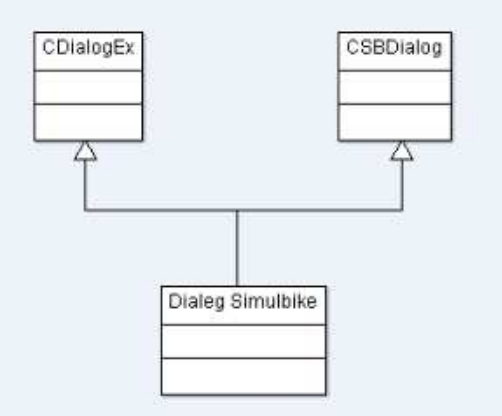
Esquema genèric de la estructura dels formularis:

Hi ha uns aspectes comuns en tots els formularis dels que consta l'aplicació per tal d'implementar les funcions comunes en tots ells. Aquestes fan referència a funcionalitats de la interacció de la unitat de control amb la interfície.

Tots els formularis de l'aplicació son classes que deriven a la classe base CDialogEx. En el següent diagrama es pot veure l'esquema:



Per tal d'afegir totes les funcionalitats d'interacció amb la interfície d'una forma més encapsulada i fàcilment exportable a tots els formularis que es creïn, s'aplica una herència múltiple que afegeix a la classe del formulari aquells atributs i mètodes comuns de gestió de la interacció. En el següent esquema es pot veure el concepte:



La descripció de la classe CSBDialog es pot veure en l'apartat d'especificacions. Tot i que es poden extreure certs atributs comuns en tots els diàlegs que creem, la classe "Diàleg Simulbike" ha d'implementar unes funcions que no es poden extreure però que tots els formularis comparteixen. Aquestes també seran afegides en l'apartat d'especificacions del present document.

Teclats:

Tal com ja s'ha comentat prèviament, l'usuari interactua amb la aplicació a partir de la unitat de control. Un dels principals problemes que es presenten és a la hora de la introducció de les dades per part de l'usuari. Per tal de realitzar aquestes operacions, es dissenyen dos classes (diàlegs): el teclat i el teclat numèric.

La estratègia que cal seguir consisteix en:

- L'usuari navega per la interfície.
- L'usuari selecciona un camp de text (numèric o alfanumèric)
- L'usuari pitja la tecla confirmació de la unitat de control.
- L'aplicació crea un objecte de la classe teclat (numèric o alfanumèric segons el cas).
- S'assigna l'element actual (el camp de text actualment seleccionat) a l'objecte teclat.
- S'assigna la posició on ha d'aparèixer el teclat respecte el camp de text.
- Es mostra el teclat en mode modal (per tal de bloquejar la el diàleg pare i que aquest surti per davant, bloquejant l'anterior).

Els teclats tenen el mateix funcionament que s'ha descrit prèviament per a tots els diàlegs, però a més tenen algunes particularitats. En la especificació de les classes del present document es pot veure la seva

descripció i la implementació.

Macros globals i logs:

Per tal de facilitar el desenvolupament, en l'aplicació s'han de definir una sèrie de macros per gestionar el tema dels logs de forma més eficient.

```
//Estructura dels logs
#define FLOG //comentar per desactivar el log
#ifndef INITLOG
#ifdef FLOG
      #define LOG(x)    flog << x << std::endl
#else
      #define LOG(x)
#endif
#define INITLOG
#endif
#define ERR(x) ferr << x << std::endl

//importem els fitxers
#ifdef FLOG
extern std::ofstream flog;
#endif
extern std::ofstream ferr;
```

Tal com es pot veure en el codi, s'han definit dos nivells de recol·lecció d'informació al llarg de l'execució de l'aplicació: el log i l'error.

El log és opcional i, en la distribució de producció, es trobarà inhabilitat. Per inhabilitar-ho, només s'ha de comentar la línia on es defineix FLOG. Segons si existeix aquest valor o no, es crearà la macro LOG que imprimirà el contingut o l'ignorarà.

El funcionament de l'error és idèntic al log, però en aquest cas no és tracta d'un opció, sempre està actiu. És la funció ERR.

Finalment, es defineixen dos variables globals que son els fitxers de log (es creen i es tanquen en el codi main de l'aplicació). Al llarg del desenvolupament, simplement s'haurà de cridar les funcions LOG o ERR amb el contingut que es vol imprimir.

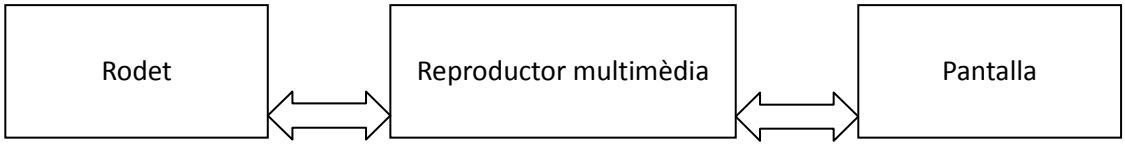
Seguint aquestes pautes, es pot tenir un registre de les incidències que es generin en l'aplicació al llarg del seu funcionament.

REPRODUCTOR MULTIMÈDIA

Visió general del sistema

En aquest bloc es descriurà con funcionarà un dels components principals que formen el prototip de l'aplicació, es tracta del reproductor multimèdia que ha de permetre la reproducció de la ruta a una velocitat variable.

Arquitectura del sistema



Tal com es pot veure, el funcionament de l'aplicació consisteix en: el rodet rebrà la velocitat a la que pedaleja l'usuari, aquest valor es comunicarà en el reproductor multimèdia de forma que pugui ajustar la velocitat de reproducció a la velocitat rebuda; finalment, el reproductor multimèdia canviarà el *frame rate* de la reproducció.

Interfícies

La classe que s'implementa en el present bloc ofereix una interfície que permetrà a la aplicació fer ús de les llibreries DirectShow, ocultant la complexitat que hi ha al darrera. Es contemplen els diferents mètodes:

- Obrir el fitxer: s'indica quin és el vídeo que es vol reproduir.
- Play: iniciar la reproducció.
- Stop: parar la reproducció.
- Pause: aturar la reproducció en el punt que actualment s'està reproduint.
- Reset: recomençar la reproducció del vídeo.
- Canviar *frame rate*: canviar la velocitat de reproducció.

Funcionalitat

Un dels aspectes clau que s'ha d'implementar en el reproductor multimèdia és la possibilitat de canviar la velocitat de reproducció sense interrompre-la. Tal com s'ha comentat prèviament, aquest era un del criteris clau que s'ha tingut en compte a l'hora de seleccionar el *framework* multimèdia que calia utilitzar.

En el prototip que desenvoluparé, he triat l'ús de les llibreries del DirectShow ja que implementen la funcionalitat de variar el *frame rate*, fet que les diferenciava de les altres opcions. Després de realitzar una sèrie de proves, he comprovat que el rang de valors entre els que ens podem moure està entre el 0,01 i el 4,35.

Si tenim en compte el mètode que s'ha comentat prèviament per el càlcul de la velocitat de reproducció i

partint d'una velocitat base. Podem veure que:

Fórmula original:	$F_{rate} = V_{real} / V_{base}$
Velocitat mínima:	$V_{min} = 0,01 * V_{base}$
Velocitat màxima:	$V_{max} = 4,35 * V_{base}$

En un exemple pràctic amb velocitat base de 20 km/h, tenim:

$V_{min} = 0,01 * 20 = 0,2$ km/h

$V_{max} = 4,35 * 20 = 87,0$ km/h

Per tant, el rang de velocitats amb les que podria treballar el simulador estaria entre els 0,2 i el 87 km/h.

Hi ha una sèrie de casos particulars com ara quan la velocitat és 0. Si apliquem la fórmula, veiem que:

$F_{rate} = V_{real} / V_{base} = 0,0 / V_{base} = 0$ km/h

El DirectShow no permet assignar el valor 0 com a *frame rate*; per aquest motiu, cal controlar-lo i assignar el *frame rate* mínim quan es produeixi aquest cas.

El cas general és que: *sempre que el frame rate obtingut sigui menor que el frame rate mínim tolerat (0,01), cal assignar el frame rate mínim enlloc del obtingut.*

Les altres funcionalitats d'aquest bloc es poden veure a l'Annex, en la descripció de la classe MediaPlayer.

APLICACIÓ GENERAL

Visió general del sistema

En aquest bloc es descriuran els algorismes principals que implementa l'aplicació per tal de dur a terme totes les tasques que ha de realitzar d'un forma coordinada. Les tasques que s'han de realitzar son principalment la gestió de les accions que realitza la interfície i la sincronització de totes els passos per tal d'emular l'execució de la ruta.

Arquitectura del sistema

Interfícies

El prototip que estem desenvolupant consta de una única interfície que es mostra en la següent imatge:

En la imatge es poden veure els diferents components de que consta la interfície:

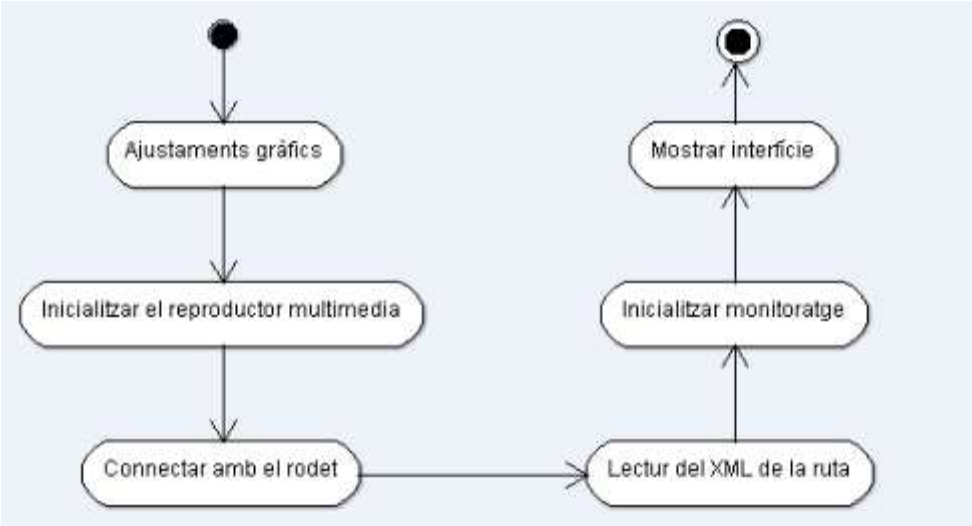
- En el lateral es mostren les diferents dades numèriques que es van produint al llarg de la realització de la ruta.
- Hi ha un mapa d'alçades on es dibuixa el perfil de la ruta i es va marcant el tram que l'usuari ja ha realitzat.
- L'espai central està reservat a la reproducció del vídeo on hi ha la ruta.
- Hi ha una barra de botons que apareix quan l'usuari vol mostrar el menú (pitjant la tecla de confirmació de la unitat de control). Els botons que apareixen son: play, stop, pause, reset i exit.

Funcionalitat

Quan s'inicia l'aplicació, aquesta ha d'executar una sèrie d'accions per tal d'inicialitzar totes les dades. Aquestes son:

1. Com que l'aplicació s'executa en pantalla completa. Primerament cal detectar quina és la resolució de la pantalla que usa l'usuari i, tot seguit, col·locar tots els elements en el lloc corresponent.
2. Cal inicialitzar el reproductor de vídeo; és a dir, assignar-li el vídeo de la ruta que es reproduirà i posar-lo en pausa. A més, cal associar el reproductor a un component de la interfície gràfica.
3. S'ha de establir una connexió amb el rodet per tal de poder rebre les ordres de l'usuari (mitjançant els botons de la UC), la velocitat i poder enviar la resistència corresponent al rodet. Això es realitza mitjançant un port sèrie.
4. S'ha d'accedir al XML que conté les dades de la ruta que utilitza el prototip i desar-ho a les estructures corresponents i realitzar els càlculs necessaris amb les dades.
5. Col·locar els valors per defecte en els camps de les dades de monitoratge.
6. Inicialitzar les estructures de dades necessàries per permetre la navegació per la interfície

mitjançant la unitat de control.



L'altre part del funcionament de l'aplicació és el que depèn de les accions que es produeixen quan es pitgen el diferents botons que conté la interfície; és a dir, les accions que executen cada un:

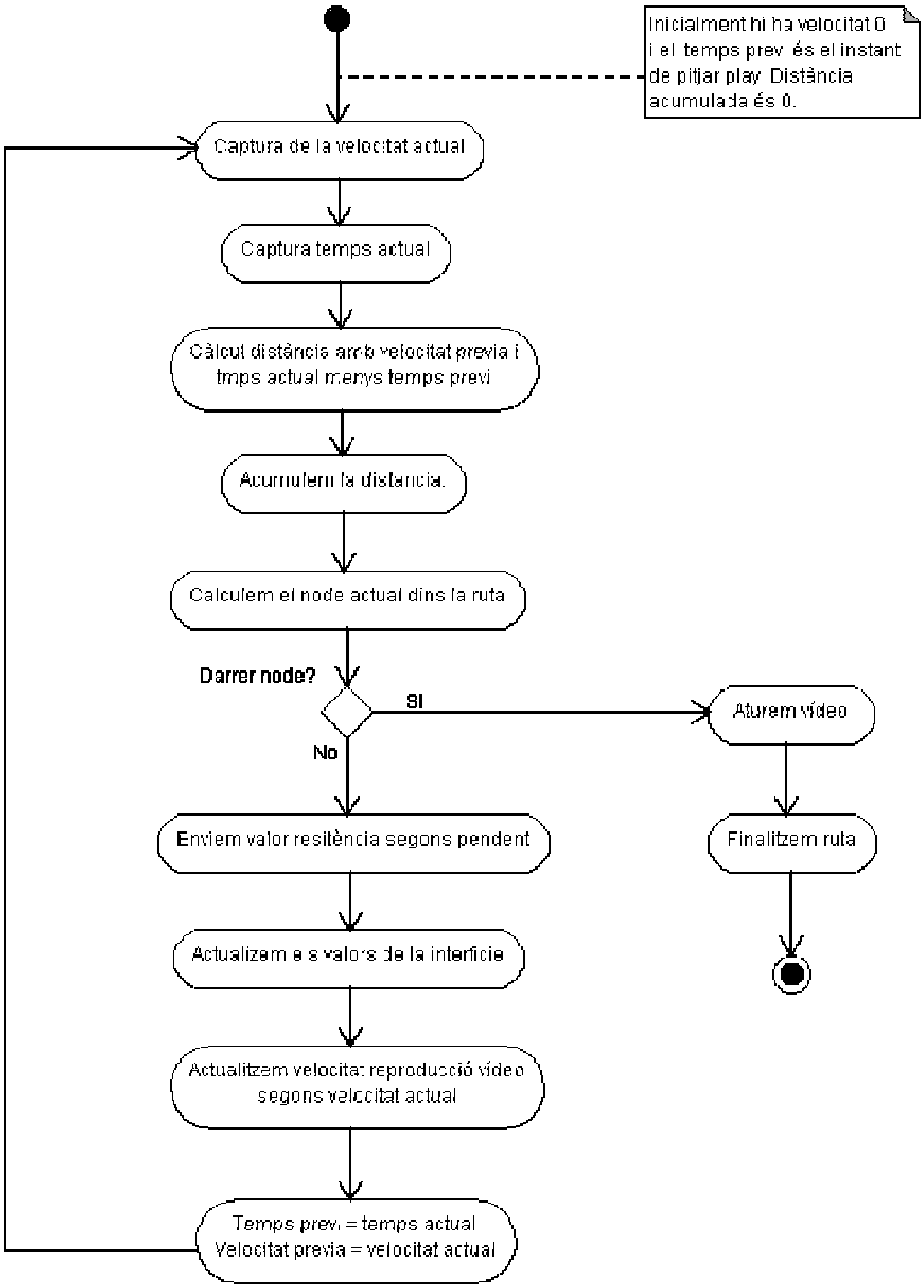
- Play: quan l'usuari pitja aquesta opció, l'aplicació inicia l'execució de la ruta. Els procés de realització de la ruta es descriurà a continuació.
- Pause: quan l'usuari pitja aquest botó, l'aplicació atura temporalment la realització de la ruta. Això vol dir que el vídeo s'atura i es deixa de realitzar els càlculs per actualitzar el monitoratge.
- Stop: quan l'usuari pitja aquest botó, aplicació retorna a l'estat inicial.
- Reset: aquesta opció simplement retorna la ruta al principi i reinicia els valors calculats. La diferència respecte a la parada és que en aquest cas, no canvia l'estat de l'usuari; és a dir, si aquest es troba pedalejant, l'aplicació continua igual, simplement recomença.
- Exit: quan l'usuari pitja aquest botó, l'aplicació ha de realitzar tots els passos necessaris per alliberar tots els recursos ocupats com ara les interfícies del DirectShow, tancar la connexió sèrie amb el rodet, l'alliberament de la memòria reservada, ...

En referència als botons, s'ha de comentar que hi ha una sèrie de condicions d'habilitació:

- Play: sempre estarà habilitat excepte quan la ruta està en fase de realització. També s'habilita quan la ruta està pausada per tal de continuar.
- Pause: s'habilita quan la ruta es troba en fase de realització (quan s'ha pitjat la opció de play).
- Stop: estarà habilitat quan la ruta es trobi pausada o en reproducció.
- Reset: es troba habilitat sempre que el simulador estigui en fase de realització o pausada.
- Exit: sempre es troba habilitat.

L'estat inicial és que només hi ha habilitats els botons de play i el d'èxit.

Finalment, hi ha un algorisme que descriu el funcionament del simulador, és a dir, tots els passos que s'han d'anar realitzant per tal de simular la realització de la ruta. En el següent diagrama es poden veure representats:



El funcionament detallat es pot veure en l'Annex del present document, en l'apartat de l'especificació de les

classes.

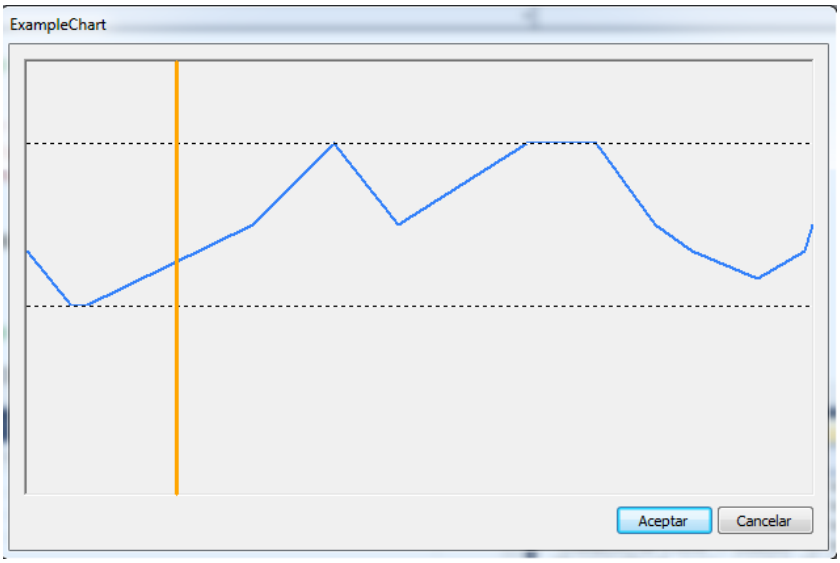
5. Test de l'aplicació

En aquest capítol s'explicaran un seguit d'exemples on es poden observar les dades que s'han obtingut mitjançant la plataforma implementada. Per tal de poder realitzar les comprovacions necessàries, s'ha creat una ruta perfectament geoposicionada i es realitzaran les proves sobre aquesta, tant a nivell teòric com amb la aplicació per tal de poder realitzar les comparacions.

5.1. Descripció de la ruta

La ruta que s'ha utilitzat per tal de realitzar les proves, consisteix en un recorregut breu realitzat per Mataró. A la hora de decidir quin tipus de recorregut calia triar, s'ha tingut en compte que complís els següents requeriments:

- Varietat de pendents: per tal de realitzar les diferents proves, el recorregut ha de constar de pujades, baixades i plans, tots ells en diferents graus d'inclinació. Concretament, el recorregut que s'ha dissenyat consta de 5 baixades, 5 pujades i dos plans. Com es pot observar, les inclinacions tenen diferents pendents per tal de posar a prova el grau de resistència.



- Breu: ha de tractar-se d'un recorregut de mida breu per tal de poder realitzar els càlculs tant a nivell teòric com mitjançant el simulador. El recorregut té un longitud de 1308 metres.

En la següent taula es poden veure les diferents coordenades que determinen el recorregut (latitud i longitud), la elevació, la distància entre nodes, la distància acumulada, la pendent i el corresponent grau de resistència (les dades es calculen del punt actual respecte l'anterior). La resistència s'obté a partir de la taula de conversió de pendent a grau de resistència, vista en l'apartat de requeriments de la present memòria.

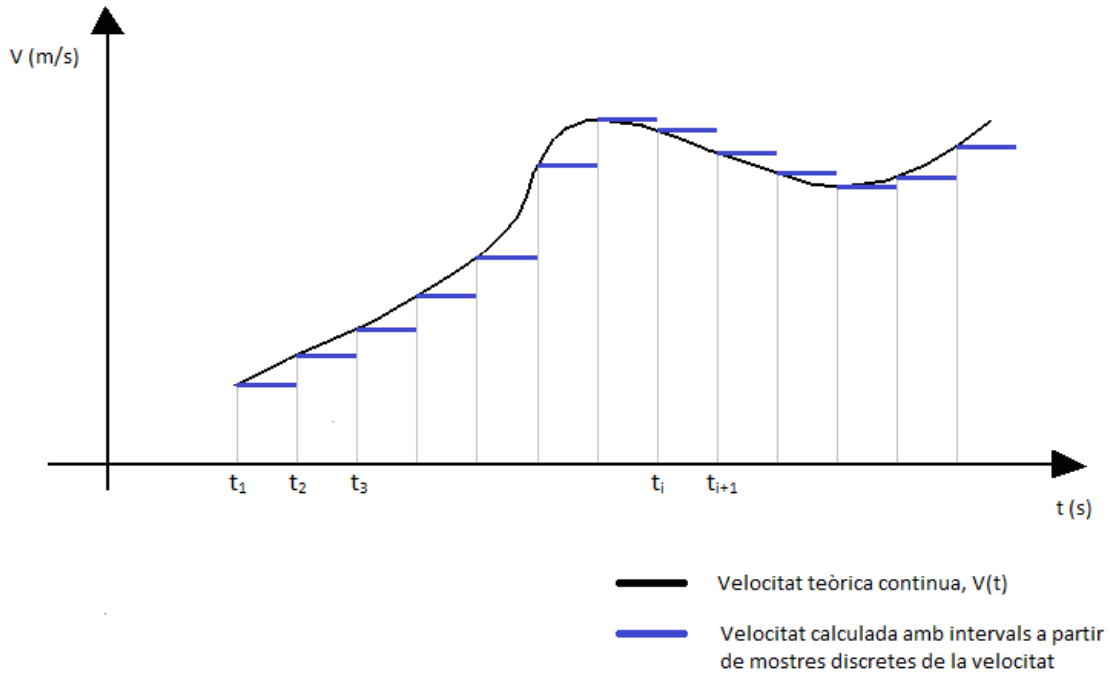
latitud	longitud	elevació	distància	distància total	pendent	resistència
41.535852	2.447076	9	0	0	-	-
41.535358	2.447715	7	76.48	76.48	-2.61574	2
41.535164	2.447669	7	21.90	98.39	0	5
41.533432	2.445259	10	278.09	376.49	1.07881	7
41.534338	2.444167	13	135.71	512.21	2.21098	8
41.534969	2.445163	10	108.64	620.86	-2.76221	2
41.536489	2.443614	13	212.59	833.46	1.41125	7
41.537075	2.444194	13	81.09	914.55	0	5
41.537243	2.444540	13	34.32	948.88	0	5
41.536490	2.445158	10	98.31	1047.19	-3.05289	1
41.536762	2.445801	9	61.48	1108.68	-1.62671	2
41.536087	2.446762	8	109.69	1218.37	-0.91167	3
41.535637	2.446042	9	78.07	1296.45	1.28087	7
41.535548	2.445968	10	11.69	1308.15	8.57887	9

En la imatge es pot veure una vista aèria del recorregut:

Les dades bàsiques han estat obtingudes amb un dispositius GPS Garmin i posteriorment visualitat en l'aplicació Google Earth (s'ha aplicat l'algorisme de Douglas-Peucker per simplificar el nombre de punts i quedar-nos amb els més significatius del recorregut), la resta de dades han estat calculades mitjançant les fórmules explicades en el document de requeriments i disseny. Resumint, aquestes son:

- Moviment rectilini uniforme (MRU): com que es va calculant la distància recorreguda dins

d'intervals de temps molt petits, es fa una aproximació i es suposa que la velocitat és constant dins del breu interval (obtenim la velocitat de forma discreta). Per tant, per calcular la distància recorreguda s'aplica la fórmula d = v * t (on v és la velocitat en m/s, la t és el temps en segons i la d la distància recorreguda en el interval en metres) a cada interval.



En la gràfica es pot veure dibuixada la funció de la velocitat respecte el temps, pintada en negre. Es tracta d'una funció continua, però el prototip no disposa d'aquesta informació. En canvi, disposem de la velocitat de forma discreta, rebem mostres cada interval de temps (representades amb t_i a la gràfica). Per tant, durant el interval t_i – t_{i+1} només tenim informació per suposar la velocitat constant (es representa en blau a la gràfica).

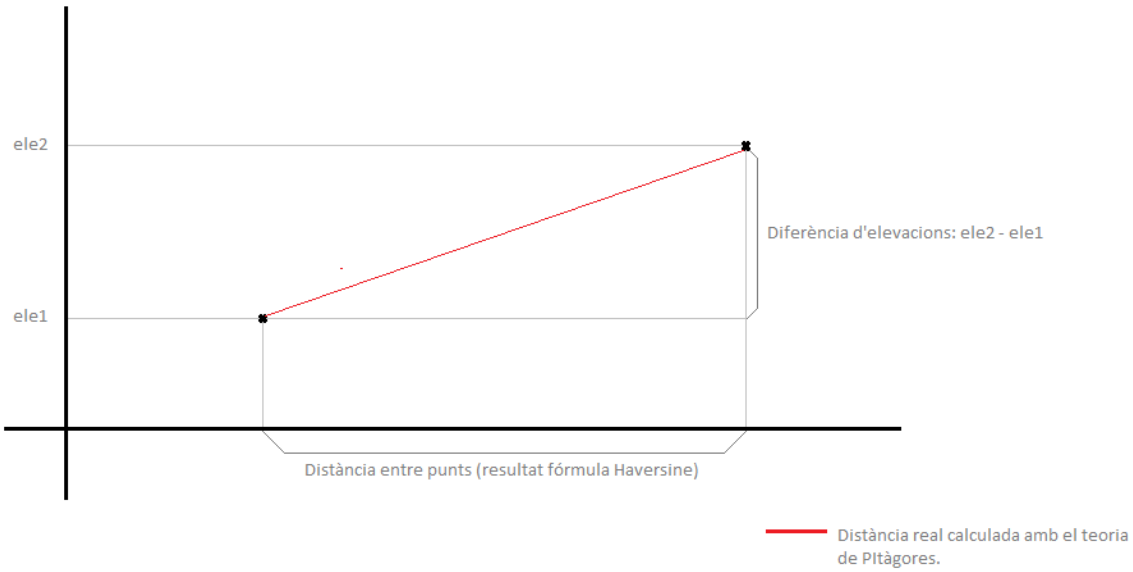
- Distància entre coordenades: per calcular la distància del recorreguts (a partir de les coordenades), s'aplica la fórmula Haversine. Aquesta es defineix com:

R = radi de la terra (6.371 km)
$\Delta lat = lat_2 - lat_1$
$\Delta long = long_2 - long_1$
$a = \sin2 (\Delta lat/2) + \cos (lat_1)*\cos (lat_2)*\sin2(\Delta long/2)$
$c = 2*atan2(sqrt(a),sqrt(1-a))$
$d = R*c$

Afegim la distància segons el pendent, ja que en aquest punt tenim la distància sobre el pla (usem el teorema de Pitàgores):

$D = sqrt(d^2 + (ele_2 - ele_2)^2)$

Tal com es pot veure, s'ha afegit un pas al final dels càlculs. Amb la fórmula de Haversine aconseguim la distància del dos punts sobre el pla. Però les coordenades poden tenir elevacions diferents, per tant, la distància sobre el pla no es la real. Per aquest motiu, s'aplica el teorema de Pitàgores on la distància en el pla i la diferència d'elevacions son els catets. D'aquesta forma obtenim la distància real (tenint en compte el pendent). Aquesta simplificació és possible ja que els punts de les coordenades es troben molt propers entre ells.



- Fórmula de la pendent: pendent = (elev2 – elev1) * 100 / d

Un cop hem vist la ruta utilitzada en les proves realitzades, es realitzarà l'estudi de dos casos diferents:

- Cas A: realització de la ruta a una velocitat constant. Es tracta del cas més senzill que ens podem trobar i ens servirà per analitzar fàcilment les dades obtingudes.
- Cas B: realització de la ruta amb variació de la velocitat. Es tracta del cas més proper a la realitat.

5.2. Exemples pràctics

Cas 1: realització ruta a velocitat constant

Tal com s'ha comentat, aquesta és una primera prova que permet comprovar si l'aplicació està funcionant en el cas més bàsic: és a dir, a una velocitat constant.

Les dades que es poden obtenir en aquesta primer prova son: els graus de resistència en els diferents trams i el temps total que es tarda en realitzar la ruta.

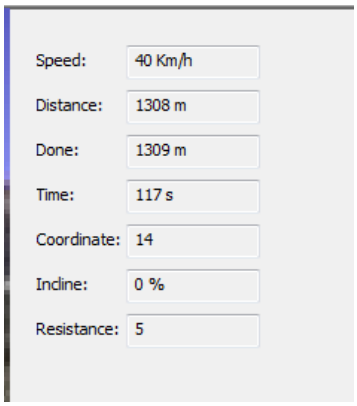
En la prova, s'ha decidit usar la velocitat de 40 km/h (11,111 m/s). A partir d'aquí, tenim que a nivell teòric, son necessaris:

d = v * t
d = d / v

d = 1308,15 m
v = 11,111 m/s

t = 1308,15 / 11,111 = 117,735 s

En la pràctica, obtenim el següent valor: 117 s (117,905s en el log)



Per tant, l'error que s'observa és de: 100 - 117,905 * 100 / 117,735 = -0.144%

Analitzant el percentatge d'error, observem que en la simulació es produeix un retard de 0.170 segons. Per la qual cosa, es considera una diferència tolerada.

Per altra banda, si estudiem els diferents graus de resistència obtinguts en cada tram, observem:

Tram	1	2	3	4	5	6	7	8	9	10	11	12	13
Esperat	2	5	7	8	2	7	5	5	1	2	3	7	9
Obtingut	2	5	7	8	2	7	5	-*	1	2	3	7	9

* el signe – indica que no hi ha hagut variació en el grau de resistència

Per tant, s'observa que les operacions d'assignació del grau de resistència si que es realitza de forma correcta. Tot seguit analitzem els instants en que s'hauria de realitzar el canvi de grau, tant a nivell teòric com experimental (Obs.: només es representen els canvis de grau):

Canvi	Instant esperat	Instant obtingut	Diferència
1	0s	0,296s	-0,296s
2	6,953s	7,498s	-0,544s
3	8,945s	9,299s	-0,353s
4	34,226s	34,5s	-0,274s
5	46,564s	46,501s	0,063s
6	56,442s	56,501s	-0,058s
7	75,769s	75,502s	0,267s
8	86,262s	86,103s	0,159s
9	95,199s	95,103s	0,096s
10	100,789s	100,304s	0,484s
11	110,761s	110,304s	0,456s
12	117,859s	117,305s	0,553s

Tal com es pot observar a la taula, hi ha diferents tipus de desajustaments: per un costat hi ha retards i per l'altre precipitacions. Aquests estan produïts per els intervals: no detectem el canvi de node en el moment exacte, sinó que aquest punt pot estar dins de l' interval i no ser detectat fins que aquest finalitza. Si aquest fet li sumem el temps de càlcul, obtenim la causa dels desajustaments.

Cas B: realització de la ruta amb variació de la velocitat

En aquesta segona prova, el que es pretén comprovar és el funcionament de l'aplicació quan la velocitat de realització de la ruta és variable. En aquest cas, es planifica una sèrie de canvis de velocitat en certs instants i es realitzaran els càlculs a nivell teòric. Posteriorment, es realitzarà la mateixa seqüència de canvis de velocitat a nivell experimental per comprovar els resultats obtinguts en els dos casos.

En la següent taula es poden veure les següents dades per cada nivell (el teòric i l'experimental): el temps

d'inici, el temps final, la velocitat en l' interval i la distància recorreguda. En la darrera columna, veurem diferència entre les distàncies recorregudes (teòrica – experimental) per tal d'obtenir els marges d'error i el percentatge d'error respecte a la distància teòrica. La planificació de la ruta queda definida com:

Velocitat	Cas teòric			Cas pràctic			Comparació	
(m/s)	t inici (s)	t final (s)	d (m)	t inici (s)	t final (s)	d (m)	dif. (m)	%
4,167	0	5	20,835	0	5,337	20,5708	0,2642	1,268
2,778	5	8	8,334	5,337	8,338	8,3361	-0,0021	-0,025
0,000	8	12	0	8,338	12,238	0	0	0
5,556	12	18	33,336	12,238	18,239	33,339	-0,003	-0,009
11,111	18	24	66,666	18,239	24,301	67,355	-0,689	-1,034
16,667	24	32	133,336	24,301	32,401	135	-1,664	-1,248
22,222	32	40	177,776	32,401	40,302	175,578	2,198	1,236
16,667	40	45	83,335	40,302	45,402	85	-1,665	-1,998
11,111	45	53	88,888	45,402	53,255	86,678	2,21	2,486
4,167	53	60	29,169	53,255	60,455	30,217	-1,048	-3,593
8,333	60	70	83,330	60,455	70,356	82,508	0,822	0,986
11,111	70	77	77,777	70,356	77,258	76,667	1,11	1,427
8,333	77	87	83,330	77,258	87,459	85,025	-1,695	-2,034
4,167	87	96	37,503	87,459	96,459	37,5	0,003	0,008
16,667	96	101	83,335	96,459	101,259	79,996	3,339	4,007
5,556	101	112	61,116	101,259	112,46	62,23	-1,114	-1,823
22,222	112	118	133,332	112,46	118,262	133,36	-0,028	-0,021
11,111	118	127.6	106,752	118,262	128,063	108,9	-2,148	-2,012
			1308,150			1308,26		

Cal destacar que les proves s'han fet mitjançat un software que envia les trames segons la planificació del recorregut de test. D'aquesta forma, els temps d'enviament s'ajusten molt al anàlisis teòric. Dit això, s'observa que generalment hi ha una diferència (tant per excés com per carència) en la distància obtinguda de forma experimental. Aquest ve produïda per la durada d'intervals que alteren l' inici i la finalització dels trams a la velocitat indicada. D'altra banda, el percentatge d'error es situa en un màxim de 4% (excepcionalment). Per tractar-se del prototip, el resultat es considera acceptable ja que s'observa una compensació, de forma que el temps total de la ruta, es manté amb una percentatge d'error de:

Error = 100 – 128,063/127,6 * 100 = -0,363%

Després de provar el prototip i comprovar els resultats que s'obtenen, es valora la solució presentada com a vàlida per la realització d'un producte complet.

Finalment, en aquest darrer cas analitzem els frame rates que l'aplicació calcula i que posteriorment aplica al reproductor. En aquest punt, cal recordar que el càlcul es realitzava a partir de la velocitat base indicada; en el cas del test, aquesta es correspon a 50 km/h.

En la taula que apareix a continuació, es poden veure els frame rates corresponents a cada velocitat que hem aplicat en la realització del test:

Velocitat (m/s)	Velocitat (km/h)	Frame rate (Vel/Vel bsae)
4,167	15	0,3
2,778	10	0,2
0,000	0	0,0 → 0,01
5,556	20	0,4
11,111	40	0,8
16,667	60	1,2
22,222	80	1,6
8,333	30	0,6

Les dades de la velocitat les passo a Km/h ja que és la unitat en que es reben des de la unitat de control. Seguidament es divideix el valor que es rep per la velocitat base per obtenir el *frame rate* que s'ha d'aplicar. Aquesta operació es realitza correctament en l'aplicació ja que, tant bon punt es rep una velocitat diferent a l'anterior, es realitza la operació i s'envia el valor al reproductor multimèdia (mitjançant la funció MediaPlayer.setRate).

El *frame rate* no és un valor acceptat per els components del DirectShow; per aquest motiu, es tracta d'una forma especial: s'assigna el *frame rate* mínim acceptat, que es correspon amb el valor 0,01.

6. Conclusions

En aquest capítol es pretén realitzar una valoració de la realització del projecte i dels resultats que finalment s'han obtingut. A continuació, es farà un anàlisi de les possibles vies de futur que presenta el producte.

6.1. Conclusions del projecte

Per poder parlar de les conclusions, cal primerament recordar quins eren els objectius que s'havien plantejat inicialment per el desenvolupament de projecte final de carrera. L'objectiu consistia en el desenvolupament d'un prototip que tenia tres pilars bàsics:

- Programació de la Unitat de control per tal de poder interactuar amb el hardware.
- Disseny i implementació d'un algorisme que permetés la simulació de la ruta geoposicionada.
- Implementació d'un reproductor multimèdia que permetés variar la velocitat de reproducció.

A continuació analitzaré cada un dels pilars per separat i, al final, comentaré les conclusions de forma global.

La programació de la unitat de control depenia en gran part de la electrònica sobre la que es treballava. Aquesta part no depenia de mi exclusivament, per tant, he treballat conjuntament amb un altre tècnic. Durant el desenvolupament s'han hagut de fer ajustaments per tal d'obtenir les dades correctament. El resultat que finalment he obtingut és una versió que ens permet realitzar les accions que s'esperen:

- Tecles de navegació: les tecles que l'usuari pitja en la unitat de control son enviades correctament, per tant, s'ha aconseguit que puguin ser utilitzades per la navegació per la interfície.
- L'enviament de la velocitat: la captura de la velocitat del rodet es realitza correctament d'acord a les especificacions. Per tant, a partir d'aquest valor, es poden realitzar correctament tots els càlculs posteriors (control de la distància i actualització del *frame rate*). En el tema de la velocitat, he tingut el problema de les velocitats baixes (per sota de 1 km/h), ja que en aquest cas es capturen valors incorrectes.
- Canvi del valor de la resistència: aquest operació es realitza correctament.

El següent punt que s'ha de tenir en compte és el del disseny de l'algorisme que permeti la simulació de la ruta a partir del *track* del recorregut i les dades de la velocitat que es capturen del rodet. Per comprovar el funcionament d'aquest algorisme, s'han realitzat les proves reflectides en el capítol 5. En ell es pot veure que totes les accions es realitzen correctament:

- Càlcul de la distància recorreguda.
- Assignació correcte dels nodes actuals en cada instant.

- Assignació correcte del grau de resistència segons el tram que s'està realitzant.

Totes aquestes operacions es realitzen de forma sincronitzada mentre l'usuari realitza la ruta. Analitzant els resultats globals, el temps que es tarda en realitzar la ruta, he vist que entre el temps teòric calculat i el temps real obtingut hi ha un error del 0,144% i del 0,363%. Per tant, la conclusió a la que arribo és que l'error es troba dins d'uns marges tolerables; per tant, la realització d'aquest punt ha estat un èxit.

Finalment queda el darrer aspecte que s'havia de realitzar: el reproductor multimèdia amb velocitat de reproducció variable. El resultat obtingut en satisfactori quan parlem d'ajustar la velocitat de reproducció del vídeo a la velocitat que es rep del rodet, però a nivell funcional no acaba de ser convincent ja que a velocitats baixes, el vídeo es reprodueix fent salts (no és una reproducció fluïda). Aquest problema és intrínsec al mètode amb vídeo; per tant, la solució passaria per canviar el mètode de visualització. Queda en estudi per la propera versió.

En una valoració global, crec que he assolit els objectius que s'havien plantejat en el desenvolupament del prototip. El sistema hardware és viable per el propòsit que es vol realitzar però, a l'hora de desenvolupar el producte complet, caldria revisar el mode de visualització.

6.2. Experiència professional

Després de realitzar aquest projecte final de carrera, he de dir que ha estat molt interessant i ha suposat tot un repte si es té en compte tots els nivells del procés de desenvolupament de software que han hagut de ser tractats. D'altra banda, la aplicació ha suposat un desenvolupament a diferents nivells: a baix nivell per la programació de la unitat de control, alt nivell amb la interfície, comunicació entre sistemes, bases de dades, ...

Cal afegir que la elaboració d'aquest projecte ha permès posar en pràctica molts dels coneixements adquirits al llarg de la carrera, especialment aquells relacionats amb la enginyeria del software, a la hora de capturar el requeriments i dissenyar l'aplicació, i les assignatures referents a algorismes i programació per tal de obtenir una solució viable al problema plantejat.

Per tant, es pot veure que el projecte representa un exemple pràctic, dins el món empresarial, de desenvolupament d'un producte software. Aquesta memòria intenta reflectir tots els passos seguits per tal de portar-lo a bon terme; aquests es corresponen amb el cicle de vida del software: requeriments, anàlisis i disseny, desenvolupament, proves, ... Per tant, crec que s'ha realitzat tot els passos que es poden esperar en la realització del projecte final de carrera d'una enginyeria en informàtica.

En el capítol previ es poden veure els resultats que s'han obtingut en les proves finals. La satisfacció per la realització d'una tasca es pot veure en el resultat final, i en aquest cas concret, crec que puc afirmar que he

assolit la meta amb èxit.

A nivell personal, aquest projecte ha estat molt enriquidor per la meva experiència en el món laboral. M'ha permès assolir els diferents rols que es troben al llarg del procés de desenvolupament de software: des del gestor de projectes, l'analista de software, el desenvolupador, al *tester*, ... sense oblidar l'estudiant, al redactar la present memòria. Durant aquest temps, he après la dificultat que representa el desenvolupament d'un software, on no hi ha un camí fàcil i on els problemes sorgeixen mentre el projecte va evolucionant. He après a analitzar les diferents opcions, seleccionar les que considero més adequades, aplicar-les al projecte, a tractar amb el client, ... Tots aquests aspectes m'han donat un coneixement addicional als obtinguts en la carrera; per tant, puc dir sense por a equivocar-me, que he crescut con a enginyer gràcies al projecte.

6.3. Vies de futur

El producte resultant del desenvolupament realitzat porta la etiqueta de prototip, no es tracta d'un producte final ni tancat. La seva utilitat ha estat de demostrar la viabilitat o no de la idea que es vol implementar. S'ha desenvolupat el que es podria considerar el nucli del projecte que realitza les accions més bàsiques que requeriria el projecte complet.

Un cop s'han vist aquests aspectes, no ha de ser una sorpresa la multitud de vies de futur que el projecte presenta. A continuació es farà una descripció de les diferents possibilitats.

La primera via de futur més evident que es podria aplicar, seria desenvolupar la solució completa que s'ha definit en l'anàlisi de requeriments que apareix en punts anteriors del present document. Si es desenvolupen tots els punts descrits, es podria dir que hi ha un producte complet i funcional. Cal observar que la plataforma web també s'inclou en el projecte global.

El problema que presenta aquesta solució és que la solució en un PC queda una mica provisional. Si la plataforma s'utilitza davant el televisor, no és pràctic vendre el producte software per ordinador. Una nova via de futur seria traslladar el software a un *Set-top* box, de forma que es pogués connectar al televisor directament sense la necessitat d'un ordinador. La dificultat d'aquesta via es troba en la implementació del software per la plataforma seleccionada. Cal observar que en aquest cas, hauria de disposar de connexió a la xarxa per tal de jugar en mode *on-line* i obtenir noves rutes. Actualment, aquesta és la via en la que es pretén dirigir el projecte.

Una altra possibilitat que es pot mirar és el tema d'aprofitar les noves televisions com ara la *Google TV*. Aquesta tecnologia encara es troba massa immadura per tal de considerar-la de forma immediata, però es podria resumir en: hi ha entitat que crea un *Set-top box* on ofereix els seus serveis, la idea seria aprofitar la pròpia plataforma per oferir la nostra aplicació (obs.: seria realitzar l'aplicació per el sistema operatiu que usi

el *Set-top* box, Google usa *Android*).

A part de la plataforma sobre la que s'integrarà l'aplicació, hi ha altres vies de futur que permetrien millor la solució actual:

- Ampliació del hardware: es podria aplicar més sensors al hardware per detectar altres successos com ara si l'usuari està assegut a la bicicleta, si té els peus al pedals, si mou el manillar, ... Totes aquestes dades podrien produir diferents accions en la reproducció de la ruta per completar el realisme.
- Móns 3-D: s'estudia la possibilitat de canviar la reproducció del vídeo per el desenvolupament de mons en tres dimensions. La avantatge que presentaria aquesta vida seria la de permetre una major llibertat de moviments (moviment del manillar), però el preu que es pagaria seria el realisme. Es deixa com una possibilitat.
- Múltiples monitors: usant múltiples monitors es podria guanyar realisme ja que es cobriria la vista del laterals (aquesta millora estaria més destinada a fires i promocions, no a usuaris domèstics).

Finalment, cal destacar un altre aspecte important que presenta el projecte: els àmbits d'aplicació. S'ha construït un mecanisme hardware i una software que permet aplicar el sistema a diferents àmbits. Alguns exemples son:

- Simulació de rutes ciclistes: àmpliament descrit en el present document.
- Promoció turística: es podria usar per promocionar regions, de forma que els interessats poguessin realitzar un recorregut virtual, ja sigui en bicicleta o mitjançant una cinta caminadora.
- Exercici per la gent gran: es podria usar el mateix sistema de la cinta caminadora per estimular l'exercici de la gent gran a les residències (mitjançant cintes caminadores).
- Gimnasos: es podria usar el sistema descrit per l'exercici de la gent gran en els gimnasos, tant en bicicleta com en cinta. Podria servir per fer menys monòton la realització de l'exercici.
- Etc.

Tal i com es pot veure, es tracta d'un projecte que ofereix diferents opcions de cara el futur.

ANNEX

Especificació de les classes del prototip

SW UNITAT DE CONTROL

Nom: SW unitat de control
Descripció: implementa les funcions requerides per el correcte funcionament de la unitat de control.
Data especificació: 05/10/2010
Versió: 1.0

ATRIBUTS I CONSTANTS

El codi ha de contenir constants per tots els valors de comanda (és a dir, les combinacions de bytes que indiquen l'ordre que es rep o les capçaleres de les respostes).

A més, ha de tenir constants que indiquin el nombre de graus de resistència que pot tenir el rodet i el valor del grau corresponent al pla (resistència neutra).

Finalment, cal que tingui definida la constant del token (no cal la constant de la resposta ja que és equivalent a la negació del token).

MÈTODES

Nom	**sendToken**
Descripció	Aquesta funció és l'encarregada de l'enviament del token al iniciar la aplicació segons les indicacions del document. També s'encarrega de escoltar i validar la resposta rebuda. Es tracta d'un bucle que ha de: - Enviar el token. - Esperar la resposta un temps determinat. - Validar la resposta si s'ha rebut o repetir els passos (també cal repetir en cas de rebre una resposta incorrecta).
Visibilitat	public
Modificadors	None
Paràmetres	Void
Retorn	0 – si hem establert una connexió. 1 – indica que s'ha produït algun error. No es pot continuar amb la execució del programa.

Nom	**initDevice**
Descripció	Aquesta funció és la encarregada de inicialitzar totes les estructures de dades i els valors inicials que han de ser usats al llarg del funcionament de l'aplicació. A més, té dos accions principals: - Posar el valor de la resistència al valor neutral. - Posar la unitat de control en modalitat de resistència.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	0 – la operació ha finalitzat correctament. 1 – s'ha produït un error en l'execució de les ordres. L'aplicació no pot continuar.

Nom	**incResistance**
Descripció	Aquesta funció ha de gestionar les operacions necessàries per tal que la unitat de control enviï la senyal al rodet i s'incrementi el grau de resistència en una unitat.

Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**decResistance**
Descripció	Aquesta funció ha de gestionar les operacions necessàries per tal que la unitat de control enviï la senyal al rodet i decreixi el grau de resistència en una unitat.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**sendSpeed**
Descripció	Es tracta de programar una interrupció per tal que executi aquesta funció cada cop que es rebin dades per el port que usa el rodet per comunicar-se. Aquesta funció ha de llegir les dades que es reben i muntar la trama que finalment s'envia per el port cap a l'ordinador.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**sendKey**
Descripció	Es tracta de programar una interrupció per tal que executi aquesta funció cada cop que es rebin dades per el port que usa el teclat de la UC per comunicar-se amb la unitat de control. Quan es rep una pulsació, s'ha de detectar la tecla polsada, muntar la trama i enviar-la per el port cap a l'ordinador.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**Main**
Descripció	Es tracta del codi principal del programa que s'executa a la unitat de control. He d'executar les següent accions: - Obrir el port de comunicació amb l'ordinador. - **sendToken** - **initDevice** - Escoltar el port a la espera de comades. - Respondre comandes. - Saltar a escoltar comandes.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

components globals

Nom: components globals
Descripció: fa referència a un conjunt de components que s'utilitzen de forma compartida al llarg del funcionament de l'aplicació i que permeten facilitar les tasques repetitives.
Data especificació: 03/10/2010
Versió: 1.0

Rodet

Nom: Rodet
Descripció: es tracta d'una classe que permet la gestió d'un port sèrie de forma encapsulada; és a dir, facilita les tasques comunes d'obertura i tancament del port sèrie, a més de gestionar la escriptura i la lectura de les dades que s'envien o es reben a través d'aquest.
Visibilitat: public
Modificadors: void
Herència: void
Data especificació: 03/10/2010
Versió: 1.0

ATRIBUTS

Nom	**inicialitzat**
Visibilitat	private
Modificadors	Void
Tipus	bool
Descripció	Es tracta d'un flag que s'utilitza per indicar si el port sèrie està obert o no.

Nom	**hPort**
Visibilitat	Private
Modificadors	Void
Tipus	HANDLE
Descripció	Port sèrie.

CONSTRUCTORS I DESTRUCTORS

Nom	**Rodet**
Descripció	Aquest mètode és usat per inicialitzar les variables de l'aplicació. Concretament inicialitza el flag de inicialitzat a fals.
Visibilitat	public
Paràmetres	Void
Retorn	Rodet: el constructor retorna una instància de la classe.

Nom	**~Rodet**
Descripció	Aquest mètode és usat per tancar el port en cas que el flag de inicialitzat indiqui que el port ha estat obert però que no s'ha tancat. Internament, crida el mètode close, que es comentarà posteriorment, en cas que el flag inicialitzat estigui al valor verdader.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Nom	**open**
Descripció	Aquest mètode és l'encarregat d'obrir una connexió amb el port sèrie segons la configuració que s'indica mitjançant els valors que es reben com a paràmetres. El mètode realitza les següents accions: - Obre el port sèrie que s'indica en el paràmetre *port*. - Posa el flag inicialitzat a verdader. - Configura el timeout per tal que les lectures del port sèrie siguin no bloquejants i l'aplicació pugui continuar executant-se tot i que no hi hagi valors per llegir (usa el paràmetre *timeout*). - Configura el port segons els paràmetres de configuració que es reben.
Visibilitat	public
Modificadors	void
Paràmetres	- Port (char*): conté la cadena indicant el port en format COMX, per exemple: COM3. - Baudrate (int): enter que indica la velocitat de transmissió. - Datasize (int): enter que indica el número de bits que conté una dada. - Parity (int): indica el nombre de bits de paritat. - Stopbits (int): indica el nombre de bits d'stop. - Timeout (int): indica el número de milisegons que espera la lectura, si passa aquest límit i no s'ha rebut cap valor, es desbloqueja l'execució.
Retorn	HRESULT: - S_OK: s'ha obert i configurat el port correctament. - Error code: codi que indica quin error s'ha produït.

Nom	**close**
Descripció	Aquest mètode ha de tancar la connexió amb el port sèrie. Per tal de realitzar aquesta tasca, comprova si el port sèrie es troba inicialitzat (mitjançant el flag); en cas d'estar-hi, crida la funció per alliberar el recurs i posa el flag inicialitzat a fals.
Visibilitat	Public
Modificadors	Void
Paràmetres	Void
Retorn	HRESULT: - S_OK: s'ha tancar el port correctament. - Error code: indicant la causa de la fallida.

Nom	**read**
Descripció	Aquest mètode rep com a paràmetre un vector de bytes i la longitud del vector. Primerament controla si el port sèrie ha estat inicialitzat. En cas de no estar-ho, surt del mètode indicant la causa. Si el port està inicialitzat, l'aplicació llegeix les dades del port sèrie i les desa en el vector (té en compte l'espai del vector). A més, retorna, mitjançant referència, el nombre de bytes llegits.
Visibilitat	public
Modificadors	Void
Paràmetres	- Buffer (char*): buffer de bytes on es desa el contingut llegit. - Len (int): indica la longitud del buffer. - readBytes (DWORD&): variable on es desa el valor corresponent al nombre de bytes que s'han llegit.
Retorn	HRESULT: - S_OK: la lectura de bytes ha finalitzat correctament. - Error code: codi d'error que indica l'error produït.

Nom	**write**
Descripció	Aquest mètode rep com a paràmetre un vector de bytes que conté el contingut que s'ha d'enviar a través del port sèrie i un nombre que indica els bytes que s'han d'enviar.

	Primerament controla si el port sèrie ha estat inicialitzat. En cas de no estar-ho, surt del mètode indicant la causa. Si el port està inicialitzat, l'aplicació envia les dades desitjades a través del port sèrie. A més, retorna, mitjançant referència, el nombre de bytes enviats.
Visibilitat	public
Modificadors	Void
Paràmetres	- Buffer (char*): buffer de bytes on hi ha el contingut que es vol enviar. - Len (int): nombre de bytes que es vol enviar. - sendBytes (DWORD&): variable on es desa el valor corresponent al nombre de bytes que s'han enviat.
Retorn	HRESULT: - S_OK: la escriptura dels bytes ha finalitzat correctament. - Error code: codi d'error que indica l'error produït.

CSBDialog

Nom: CSBDialog
Descripció: aquesta classe conté aquells atributs i mètodes que tots els formularis necessiten per tal de respondre als events produïts per la unitat de control enlloc dels produïts pels dispositius d'entrada tradicionals (teclat i ratolí).
Visibilitat: public
Modificadors: none
Herència: none
Data especificació: 04/10/2010
Versió: 1.0

ATRIBUTS

Nom	**keys**
Visibilitat	Private
Modificadors	void
Tipus	Vector: vector d'apuntadors a estructures Key_st
Descripció	Aquest vector és usat per contenir els apuntadors a totes les estructures que es creïn mitjanant memòria dinàmica. D'aquesta forma, en el destructor es pot alliberar tota la memòria reservada.

Nom	**ckey**
Visibilitat	Protected
Modificadors	None
Tipus	Apuntador a estructura Key_st
Descripció	Aquest apuntador s'utilitza per contenir una referència a l'element seleccionat al llarg de tota l'estona en que l'aplicació està en ús.

Nom	**hThread**
Visibilitat	Protected
Modificadors	void
Tipus	Apuntador a CWinThread
Descripció	Element usat per contenir una referència al thread que ha d'anar llegint les tecles que pitja l'usuari.

Nom	**width**
Visibilitat	Protected
Modificadors	Void
Tipus	Int

Descripció	Conté el valor de l'amplada total de la finestra o CDialog.

Nom	**height**
Visibilitat	Protected
Modificadors	Void
Tipus	Int
Descripció	Conté el valor de l'alçada total de la finestra o CDialog.

CONSTRUCTORS I DESTRUCTORS

Nom	**CSBDialog**
Descripció	Crea una instància de la classe. No ha de realitzar cap acció.
Visibilitat	Public
Paràmetres	Void
Retorn	Instància del CSBDialog

Nom	**~CSBDialog**
Descripció	Mètode que crida la funció deleteKeys en cas que el vector keys contingui algun element per tal de poder garantir que s'allibera tota la memòria.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Nom	**commRodet**
Descripció	És el mètode que executa constantment el thread. Cal tenir en compte el flag active_rodet que s'ha descrit prèviament. Mentre el flag es trobi actiu, el mètode crida la funció read del rodet. Si aquesta retorna S_OK (lectura correcta), es comprova si el nombre de bytes és més gran que 0. En cas de no haver llegit bytes, es realitza una pausa. Si es llegeixen bytes, es mira el primer per comprovar si és una tecla. En cas contrari, es salta a la pausa. Si és una tecla, es dispara un event al que s'ha li passa la tecla pitjada de forma que pugui ser atesa adequadament (el mètode que s'executa és el OnApplyKey comú en tots els diàlegs i que serà descrit a posteriorment).
Visibilitat	Protected
Modificadors	static
Paràmetres	LPVOID lpParam: un tipus de referència al formulari actual.
Retorn	UINT: - S_OK: finalització correcta.

Nom	**startCOMM**
Descripció	Aquest mètode és utilitzat tal de llençar el thread que anirà llegint les tecles que pitja l'usuari. El thread ha d'estar associar al diàleg actual per tal de permetre l'accés a els components d'aquest. Cal indicar al thread que el mètode que s'executa és el commRodet.
Visibilitat	Protected
Modificadors	Void
Paràmetres	- Apuntador al diàleg actual (CDialog*)
Retorn	HRESULT: - S_OK: s'ha llençat el thread correctament. - Error code: indicant la causa de l'error.

Nom	**unselectField**
Descripció	Aquest mètode rep una referència al diàleg o finestra actual. És l'encarregat de desmarcar l'element que actualment està seleccionat per tal de refrescar la interfície. Aquí s'usa el camp tipus de la estructura Key_st per tal de saber quina acció s'ha de realitzar: - Camp de text: no s'ha de realitzar cap acció. - Botó: s'ha de canviar l'estil a l'estil per defecte.
Visibilitat	Protected
Modificadors	Void
Paràmetres	- Apuntador al diàleg actual (CDialog*)
Retorn	void

Nom	**selectField**
Descripció	Aquest mètode rep una referència al diàleg o finestra actual. És l'encarregat de marcar d'alguna forma l'element que actualment està seleccionat per tal de que ressalti en la interfície. Aquí s'usa el camp tipus de la estructura Key_st per tal de saber quina acció s'ha de realitzar: - Camp de text: cal assignar-li el focus. - Botó: s'ha de assignar el focus i canviar l'estil del botó.
Visibilitat	Protected
Modificadors	Void
Paràmetres	- Apuntador al diàleg actual (CDialog*).
Retorn	Void

Nom	**arrowKey**
Descripció	Aquest mètode rep dos paràmetres: una referència al diàleg o finestra actual i l'identificador de la tecla que s'ha pitjat. Segons el valor de la tecla pitjada, ha de mirar si l'element actual (estructura a la que apunta l'apuntador ckey) té un camí en la direcció indicada. Si n'hi ha, ha de: 1. Cridar el mètode unselectField per treure el focus a l'element actual. 2. Actualitzar el node actual per el node que ve a continuació segons la direcció. 3. Cridar el mètode selectField per tal de marca el nou node actual en la interfície. En cas que la direcció indicada no apunti a cap estructura, el mètode no realitza cap acció.
Visibilitat	Protected
Modificadors	Void
Paràmetres	- Apuntador al diàleg actual (CDialog*) - Value (char): identificador de la tecla pitjada.
Retorn	void

Nom	**deleteKeys**
Descripció	Aquest mètode ha de recórrer tots els elements del vector keys (son apuntadors) i anar alliberant la memòria que aquests tenen reservada de forma dinàmica.
Visibilitat	Protected
Modificadors	Void
Paràmetres	Void
Retorn	Void

Nom	**addKey**
Descripció	Aquest mètode rep com a paràmetre un apuntador a una estructura de tipus key_st. Ha de comprovar que el valor que es rep no sigui nul i, si no ho és, afegeix l'element en el vector keys.
Visibilitat	Protected
Modificadors	Void
Paràmetres	Apuntador a estructura Key_st
Retorn	void

Formularis genèrics

Nom: components comuns als formularis
Descripció: en aquest apartat s'especificaran aquells components i mètodes que han de tenir les classes que representen els diàlegs de l'aplicació per tal que es puguin controlar mitjançant l'ús de les tecles de la unitat de control.
Visibilitat: public
Modificadors: void
Herència: CDialogEx, CSBDialog
Data especificació: 05/10/2010
Versió: 1.0

EVENTS

Nom	WM_UPDATE_INTERFACE
Visibilitat	Public
Modificadors	None
Tipus	ON_MESSAGE
Descripció	Es tracta d'indicar a la interfície que quan es rebi el missatge WM_UPDATE_INTERFACE, l'aplicació ha de executar el mètode OnApplyKey. Cal afegir la línia: ON_MESSAGE(WM_UPDATE_INTERFACE, &CLoginDlg::OnApplyKey)

CONSTRUCTORS I DESTRUCTORS

Nom	**~CnomDlg (on nom és el nom del diàleg que es crea)**
Descripció	Una de les accions que ha de realitzar el destructor és la de posar el flag global "active_rodet" a fals per tal d'interrompre el thread que va llegint les dades del rodet. A més, ha de cridar el mètode "deleteKeys" (que heretem de CSBDialog) per tal d'alliberar memòria.
Visibilitat	public
Paràmetres	Void
Retorn	void

MÈTODES

Nom	**OnPaint**
Descripció	Una de les accions que s'han de realitzar en aquest mètode és la de configurar el diàleg, és a dir, assignar el fons que es vol utilitzar. A més, cal cridar el mètode "selectField" (que s'hereda de CSBDialog) per tal de que aparegui marcat a la interfície l'element actualment seleccionat.
Visibilitat	Public
Modificadors	Afx_msg
Paràmetres	Void
Retorn	void

Nom	**OnInitDialog**
Descripció	En aquest mètode s'ha d'indicar que la interfície s'obri en mode maximitzat (per tal d'ocupar tota la pantalla). Una altra acció important que es realitza en aquest mètode és la obtenció de les mesures de la pantalla mitjançant l'alçada i amplada del diàleg (està obert de forma que ocupa tota la pantalla). A continuació, i a partir de les mesures de la pantalla, en aquest mètode es col·loquen tots els elements que apareixen a la interfície mitjançant les mesures de top, bottom, left i right. A més, també es crida el mètode "initKeyboard" que es descriurà a continuació. Finalment, es crida el mètode "startCOMM" (que s'hereta de CSBDialog) per tal de iniciar el thread que llegeix les dades de la unitat de control.
Visibilitat	Public
Modificadors	Virtual
Paràmetres	Void
Retorn	BOOL: - TRUE: s'ha inicialitzat correctament. - FALSE: ha fallat la inicialització

Nom	**initKeyboard**
Descripció	Aquest mètode és usat per crear les estructures Key_st per cada element de la interfície al l'usuari pot accedir. El procés a seguir consisteix en: - Reservar memòria per crear una estructura per cada element que l'usuari podrà activar (es treballa amb apuntadors a estructures). - Marcar les relacions per cada element (on s'anirà en cas de pitjar la tecla amunt, avall, dreta o esquerra). - Desar els apuntadors a les estructures al vector mitjançant el mètode "addKey" (que s'hereta de CSBDialog). - Assignar quin serà l'element inicialment seleccionat mitjançant l'assignació d'un apuntador a l'atribut "ckey" que s'hereta de CSBDialog.
Visibilitat	Private
Modificadors	Void
Paràmetres	Void
Retorn	Void

Nom	**OnApplyKey**
Descripció	Es tracta del mètode que disparará el thread cada cop que es rebi els bytes que indiquen la pulsació d'una tecla.

	Aquest mètode ha de distingir dos casos: si es tracta de la pulsació del botó de confirmació o si es tracta d'una de les quatre direccions. En cas de tractar-se d'una de les quatre direccions, es crida el mètode "arrowKey" (que s'hereta de CSBDialog) passant com a paràmetre l'identificador de la tecla pitjada per tal de donar la resposta corresponent. En cas de tractar-se del botó de confirmació, l'aplicació conté una estructura switch on es compara el identificador de l'element actualment seleccionat (camp id de la estructura que apunta l'atribut ckey). Per cada id, s'ha de realitzar l'acció corresponent.
Visibilitat	Private
Modificadors	Void
Paràmetres	- wParam (WPARAM) - lParam (LPARAM)
Retorn	HRESULT: - S_OK: si tot ha anat bé.

CKeyboardDlg

Nom: CKeyboardDlg (teclat alfanumèric)
Descripció: conté tots els caràcters d'un teclat QWERTY i permet la navegació i la selecció de les tecles mitjançant l'ús de la unitat de control.
Visibilitat: public
Modificadors: none
Herència: public CDialogEx, public CSBDialog
Data especificació: 06/10/2010
Versió: 1.0

ATRIBUTS

Nom	**Field**
Visibilitat	Private
Modificadors	None
Tipus	CWnd* (apuntador a l'element de la interfície)
Descripció	Aquest atribut serveix per contenir una referència a l'element de la interfície que es vol modificar. En aquest cas, consisteix en el camp de text.

Nom	**x**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Aquest valor conté la posició on s'ha de col·locar la finestra que conté el teclat. Concretament conté la coordenada X.

Nom	**y**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Aquest valor conté la posició on s'ha de col·locar la finestra que conté el teclat. Concretament conté la coordenada Y.

Nom	**Mayus**
Visibilitat	Private

Modificadors	None
Tipus	Bool
Descripció	Aquest valor booleà conté un indicador per distingir el cas en que l'usuari ha habilitat les majúscules o les ha inhabilitat; és a dir, conté l'estat en que es troben tecles (Majúscules o minúscules).

Nom	**Letters**
Visibilitat	Private
Modificadors	None
Tipus	Std::vector<int> (vector d'enters)
Descripció	Serveix per contenir tots els identificadors (enters) que identifiquen els diferents botons que formen el teclat i que es corresponen a les lletres (serveix per indicar quines tecles han de canviar la etiqueta que mostren quan l'usuari habilita o inhabilita les majúscules).

CONSTRUCTORS I DESTRUCTORS

Nom	**CKeyboardDlg**
Descripció	En el constructor de la classe és on es realitzen les inicialitzacions dels atributs de la classe. Cal: - L'apuntador field és nul. - Les majúscules es troben habilitades. - La posició X i Y és per defecte 0.
Visibilitat	public
Paràmetres	CWnd* parent (fa referència al formulari pare)
Retorn	Void

Nom	**~CKeyboardDlg**
Descripció	En el destructor es realitza la neteja de la memòria que s'ha reservat dinàmicament. Cal posar a fals el flag global active_rodet per aturar el thread que va llegint el port sèrie. A continuació és crida el mètode deleteKeys per tal d'alliberar la memòria reservada.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Observació: aquest diàleg, al igual que tots els altres de l'aplicació, hereta de la classe CSBDialog; per tant, tots els mètodes prèviament descrits per aquesta es troben disponibles en l'actual classe.

Nom	**setField**
Descripció	Aquest mètode rep una referència a un element del formulari pare (normalment es tractarà d'un camp de text). La seva tasca consisteix en assignar la referència que es rep a l'atribut field de l'objecte.
Visibilitat	Public
Modificadors	none
Paràmetres	CWnd* (apuntador a un element del formulari).
Retorn	void

Nom	**setPosition**

Descripció	Aquest mètode rep dues coordenades que indiquen el punt en que ha d'aparèixer el teclat que es vol mostrar. La tasca a realitzar consisteix en assignar aquests valors als atributs X i Y de l'objecte (posteriorment el mètode OnInitDialog els usarà per tal de mostrar el formulari en el punt indicat).
Visibilitat	public
Modificadors	None
Paràmetres	Const int X Const int Y
Retorn	void

Nom	**OnApplyKey**
Descripció	La finalitat d'aquesta funció ja ha estat descrita en l'apartat de mètodes comuns de tots els formularis. Aquí simplement queda descriure els casos particulars que es poden donar segons la tecla (del formulari) que l'usuari confirmi: - Tecla majúscula: s'ha de negar el valor actual de flag mayus i fer una crida el mètode changeCaptionsMayus. - Tecla retrocedir: s'ha de eliminar el darrer caràcter introduït i retrocedir una posició en el camp de text. - Tecla espai: s'ha d'imprimir un espai en blanc. - Tecla exit: s'ha de tancar el diàleg que conté el teclat. - Per defecte: s'agafa el text de la tecla actual i s'encadena al final de la cadena que l'usuari va escrivint.
Visibilitat	Public
Modificadors	None
Paràmetres	WPARAM wParam LPARAM lParam
Retorn	LRESULT: - S_OK si tot ha finalitzat correctament.

Nom	**changeCaptionsMayus**
Descripció	Aquest mètode és utilitzat per canviar les etiquetes de les tecles segons el valor de l'atribut booleà. Si aquest darrer està a verdader, les etiquetes han de ser mostrades en majúscules. En cas contrari, en minúscules.
Visibilitat	Private
Modificadors	None
Paràmetres	Void
Retorn	Void

CNumKeyboardDlg

Nom: CNumKeyboardDlg
Descripció: aquest formulari conté un teclat numèric virtual per tal que l'usuari pugui completar els camps numèrics mitjançant la unitat de control.
Visibilitat: public
Modificadors: none
Herència: public CDialogEx, public CSBDialog
Data especificació: 13/10/2010
Versió: 1.0

ATRIBUTS

Nom	**Field**
Visibilitat	Private
Modificadors	None

Tipus	CWnd* (apuntador a l'element de la interfície)
Descripció	Aquest atribut serveix per contenir una referència a l'element de la interfície que es vol modificar. En aquest cas, consisteix en el camp de text.

Nom	**x**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Aquest valor conté la posició on s'ha de col·locar la finestra que conté el teclat. Concretament conté la coordenada X.

Nom	**y**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Aquest valor conté la posició on s'ha de col·locar la finestra que conté el teclat. Concretament conté la coordenada Y.

Nom	**decimal**
Visibilitat	Private
Modificadors	None
Tipus	Bool
Descripció	Aquest flag és per determinar si el camp que l'usuari ha de completar admet nombres amb decimals o requereix nombres enters (habilita o inhabilita la tecla de punt decimal). La inhabilitació del botó es realitza en el mètode initKeyboard.

CONSTRUCTORS I DESTRUCTORS

Nom	**CNumKeyboardDlg**
Descripció	En el constructor de la classe és on es realitzen les inicialitzacions dels atributs de la classe. Cal: - L'apuntador field és nul. - El decimal es troba habilitat. - La posició X i Y és per defecte 0.
Visibilitat	public
Paràmetres	CWnd* parent (fa referència al formulari pare)
Retorn	Void

Nom	**~CNumKeyboardDlg**
Descripció	En el destructor es realitza la neteja de la memòria que s'ha reservat dinàmicament. Cal posar a fals el flag global active_rodet per aturar el thread que va llegint el port sèrie. A continuació és crida el mètode deleteKeys per tal d'alliberar la memòria reservada.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Observació: aquest diàleg, al igual que tots els altres de l'aplicació, hereta de la classe CSBDialog; per tant, tots els mètodes prèviament descrits per aquesta es troben disponibles en l'actual classe.

Nom	**setField**
Descripció	Aquest mètode rep una referència a un element del formulari pare (normalment es tractarà d'un camp de text). La seva tasca consisteix en assignar la referència que es rep a l'atribut field de l'objecte.
Visibilitat	Public
Modificadors	none
Paràmetres	CWnd* (apuntador a un element del formulari).
Retorn	void

Nom	**setPosition**
Descripció	Aquest mètode rep dues coordenades que indiquen el punt en que ha d'aparèixer el teclat que es vol mostrar. La tasca a realitzar consisteix en assignar aquests valors als atributs X i Y de l'objecte (posteriorment el mètode OnInitDialog els usarà per tal de mostrar el formulari en el punt indicat).
Visibilitat	public
Modificadors	None
Paràmetres	Const int X Const int Y
Retorn	void

Nom	**OnApplyKey**
Descripció	La finalitat d'aquesta funció ja ha estat descrita en l'apartat de mètodes comuns de tots els formularis. Aquí simplement queda descriure els casos particulars que es poden donar segons la tecla (del formulari) que l'usuari confirmi: - Tecla punt: si el decimal està habilitat, s'afegeix el punt decimal al nombre que s'està escrivint (tenint en compte que si el camp està buit, es posa el 0 abans, i si el camp ja té un punt decimal, s'ignora). - Tecla retrocedir: s'ha de eliminar el darrer caràcter introduït i retrocedir una posició en el camp de text. - Tecla exit: s'ha de tancar el diàleg que conté el teclat. - Per defecte: s'agafa el text de la tecla actual i s'encadena al final de la cadena que l'usuari va escrivint (en aquest cas son números).
Visibilitat	Public
Modificadors	None
Paràmetres	WPARAM wParam LPARAM lParam
Retorn	LRESULT: - S_OK si tot ha finalitzat correctament.

Nom	**setDecimal**
Descripció	Aquest mètode rep un indicador (verdader o fals) per saber si el camp numèric que es vol omplir accepta números enters o decimals. La seva funció consisteix en desar aquest valor en l'atribut decimal.
Visibilitat	Public
Modificadors	None
Paràmetres	Bool: value
Retorn	Void

Media player

Nom: Reproductor Multimèdia.
Descripció: es tracta del sistema que conté tota el codi necessari que permet la reproducció d'una ruta (vídeo) mitjançant l'ús de les llibreries DirectShow.
Data especificació: 24/11/2010
Versió: 1.0

MediaPlayer

Nom: MediaPlayer
Descripció: conté totes les funcions que necessita l'aplicació per la reproducció d'un vídeo, creant una interfície senzilla que permeti l'ús del reproductor, ocultant la complexitat de la implementació.
Visibilitat: public
Modificadors: none
Herència: none
Data especificació: 24/11/2010
Versió: 1.0

ATRIBUTS

Nom	**Inicialitzat**
Visibilitat	Private
Modificadors	None
Tipus	Bool
Descripció	Aquest flag és utilitzat per tal d'indicar si s'han inicialitzat els components necessaris del DirectShow abans de començar a fer-ne ús.

Nom	**Fitxer**
Visibilitat	Private
Modificadors	None
Tipus	Bool
Descripció	Aquest flag ens indica si s'ha assignat un fitxer de vídeo al reproductor abans de començar a treballar amb ell.

Nom	**Assignat**
Visibilitat	Private
Modificadors	None
Tipus	Bool
Descripció	Aquest flag ens indica si s'ha assignat un marc o contenidor sobre el que reproduir el vídeo (l'espai on es mostrarà el vídeo).

Nom	**State**
Visibilitat	Private
Modificadors	None
Tipus	StatesDS, es tracta d'una enumeració que pot tenir els valors STOP, PLAY i PAUSE.
Descripció	Aquest atribut ens indica l'estat del reproductor, és a dir, el que està fent.

A més, també requereix un seguit d'apuntadors a les interfícies del DirectShow. Aquests son: IGraphBuilder, IMediaControl, IMediaEvent, IMediaEventEx, IMediaSeeking i IVideoWindow.

CONSTRUCTORS I DESTRUCTOR

Nom	**MediaPlayer**
Descripció	S'ocupa d'inicialitzar tots els flags a fals i assignar l'estat STOP a l'atribut d'estat.
Visibilitat	Public
Paràmetres	None

Retorn	Una instància de la classe MediaPlayer.

Nom	**~MediaPlayer**
Descripció	La seva funcionalitat consisteix en l'alliberació dels recursos. Per aquest motiu, ha de cridar el mètode clean, que es descriurà posteriorment.
Visibilitat	Public
Paràmetres	None
Retorn	None

MÈTODES

Nom	**init**
Descripció	El seu objectiu consisteix en inicialitzar tots els components i interfícies que requereix el DirectShow (IGraphBuilder, IMediaControl, IMediaEvent, IMediaEventEx, IMediaSeeking i IVideoWindow). És a dir, ha de construir els Graph per el que circularà el flux del vídeo fins a la seva reproducció. Cal notar que si tot s'ha inicialitzat correctament, s'ha de posar a true el flag de inicialitzat.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	HRESULT: - S_OK: si tot s'ha inicialitzat correctament. - Error code: codi que indica l'error que s'ha produït.

Nom	**openFile**
Descripció	Primer comprova si s'ha inicialitzat correctament totes les interfícies mitjançant el flag corresponent. A continuació mira l'estat del reproductor per veure si es troba en PLAY o PAUSE (en mig d'una reproducció), per la qual cosa, no es pot canviar el fitxer. Si no succeeix cap dels dos casos, s'intenta obrir el vídeo que s'ha indicat per paràmetre.
Visibilitat	Public
Modificadors	None
Paràmetres	char*: cadena de text que conté la ruta i el nom del fitxer que s'ha de reproduir.
Retorn	HRESULT: - S_OK: el vídeo indicat s'ha pogut obrir correctament. - Error code: valor que indica la naturalesa de l'error produït.

Nom	**link**
Descripció	Aquest mètode realitza una comprovació inicial: mira si s'han inicialitzat els diferents components del DirectShow. Si s'ha inicialitzat, es realitzen les operacions necessàries per col·locar el vídeo i assignar-li les dimensions correctes d'acord amb els paràmetres que es reben.
Visibilitat	Public
Modificadors	None
Paràmetres	- HWND: finestra o formulari principal de l'aplicació on volen afegir el vídeo. - OAHWND: element contenidor on s'afegirà el vídeo. - int width, int height: dimensions de l'espai que ocuparà el vídeo. - int left, int top: posició on s'ha de col·locar el vídeo.

Retorn	HRESULT: - S_OK: si no s'ha produït cap error en les operacions. - Error code: indicant la causa de l'error.

Nom	**clean**
Descripció	Aquest mètode mira si el flag inicialitzat és true. En aquest cas, allibera tots els recursos que les interfícies del DirectShow tenen reservats i assigna false a tots els flag de l'objecte.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**play**
Descripció	Aquest mètode ha de comprovar 4 condicions: - S'han inicialitzat els components del DirectShow (flag inicialitzat). - S'ha assignat el reproductor a un component de la interfície (flag assignat). - S'ha assignat un fitxer a reproduir (flag fitxer). - L'estat del reproducció no és PLAY. Si tot es compleix, l'aplicació inicia la reproducció del vídeo i canvia l'estat de l'objecte a PLAY.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	HRESULT: - S_OK: no hi ha hagut cap problema. - Error code: indicant la naturalesa de l'error.

Nom	**stop**
Descripció	Primerament comprova que s'hagin inicialitzat les interfícies i que hi hagi un fitxer assignat. A continuació es realitza la parada de la reproducció i es canvia l'estat a STOP. Finalment, s'ha de cridar el mètode reset per tal de col·locar el vídeo al principi.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	HRESULT: - S_OK: la operació s'ha finalitzat correctament. - Error code: indicant la naturalesa de l'error

Nom	**pause**
Descripció	Aquest mètode realitza 3 comprovacions: - Les interfícies estan inicialitzades. - S'ha assignat un fitxer. - L'estat actual del reproductor és PLAY. En aquest cas, pausa la reproducció i canvia l'estat actual a PAUSE.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	HRESULT:

	- S_OK: la operació s'ha finalitzat correctament. - Error code: indicant la naturalesa de l'error.

Nom	**reset**
Descripció	Primerament, es comprova que s'hagin inicialitzat els components i s'hagi assignat un fitxer. A continuació col·loca el vídeo en la posició inicial.
Visibilitat	public
Modificadors	None
Paràmetres	Void
Retorn	HRESULT: - S_OK: la operació s'ha finalitzat correctament. - Error code: indicant la naturalesa de l'error.

Nom	**setRate**
Descripció	Primerament comprova que s'hagin inicialitzat les interfícies. A continuació, assigna el nou frame rate que es rep com a paràmetre al reproductor.
Visibilitat	Public
Modificadors	None
Paràmetres	- double: el valor del nou frame rate.
Retorn	HRESULT: - S_OK: si la operació finalitza correctament. - Error code: indicant la naturalesa de l'error.

prototip

Nom: Prototip Simulbike
Descripció: aquest subsistema conté tots els components necessaris per a completar la implementació del prototip de la plataforma Simulbike.
Data especificació: 02/12/2010
Versió: 1.0

Coordenada

Nom: CCoord
Descripció: classe que conté els atributs que emmagatzemen els valors que fan referència a una coordenada geogràfica.
Visibilitat: public
Modificadors: none
Herència: none
Data especificació: 02/12/2010
Versió: 1.0

ATRIBUTS

Nom	**latitude**
Visibilitat	Private
Modificadors	None
Tipus	Double
Descripció	Conté el valor que fa referència a la latitud (component de la coordenada).

Nom	**longitude**

Visibilitat	Private
Modificadors	None
Tipus	Double
Descripció	Conté el valor que fa referència a la longitud (component de la coordenada).

Nom	**elevation**
Visibilitat	Private
Modificadors	none
Tipus	Int
Descripció	Conté la elevació de la coordenada actual.

Nom	**distance**
Visibilitat	Private
Modificadors	None
Tipus	Float
Descripció	Conté la distància que hi ha entre la coordenada anterior i la actual.

Nom	**distance_absolute**
Visibilitat	Private
Modificadors	None
Tipus	Float
Descripció	Conté la distància que hi ha des del principi a la ruta fins a la coordenada actual.

Nom	**incline**
Visibilitat	Private
Modificadors	None
Tipus	Float
Descripció	Conté el percentatge d'inclinació que hi ha en el tram que va des del punt anterior a l'actual.

CONSTRUCTORS

Nom	**CCoord**
Descripció	Constructor per defecte, la seva funció consisteix en inicialitzar tots els atributs amb el valor 0.
Visibilitat	Public
Paràmetres	Void
Retorn	Una instància de la classe.

Nom	**CCoord**
Descripció	Constructor que permet inicialitzar els atributs bàsics: longitud, latitud i elevació.
Visibilitat	Public
Paràmetres	- Double: latitude - Double: longitude - Int: elevation
Retorn	Una instància de la classe.

Nom	**~CCoord**
Descripció	Destructor de la classe. En Aquesta classe, no ha de realitzar cap acció.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Aquesta classe conté tots els mètodes getters & setters per tal de que es pugui accedir a tots els atributs.

Track

Nom: CTrack
Descripció: es tracta d'una classe que conté els mètodes i atributs necessaris per tal de que el simulador pugui tractar una ruta. En essència, es tracta d'un vector d'objectes de tipus CCoord.
Visibilitat: public
Modificadors: none
Herència: none
Data especificació: 02/12/2010
Versió: 1.0

ATRIBUTS

Nom	**vector**
Visibilitat	Private
Modificadors	<CCoord*>: es tracta d'un vector d'apuntadors a objectes de tipus CCoord.
Tipus	std::vector
Descripció	Conté les coordenades (objectes CCoord) ordenats segons la seva posició dins del recorregut.

CONSTRUCTORS

Nom	**CTrack**
Descripció	Constructor per defecte. No realitza cap acció
Visibilitat	public
Paràmetres	Void
Retorn	Una instància de la classe.

Nom	**~CTrack**
Descripció	Destructor de la classe. Crida la funció clean que es descriurà a continuació.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Nom	**addCoord**
Descripció	Aquest mètode és usat per afegir una nova coordenada a la ruta. Crea un nou objecte CCoord (a partir d'un apuntador) i desa els valors de latitud, longitud i elevació que rep com a paràmetres. Finalment, desa la nova coordenada en el vector.
Visibilitat	Public
Modificadors	None
Paràmetres	- Double: lat - Double: long - Int: ele
Retorn	Void

Nom	**getCoord**
Descripció	El primer que realitza el mètode és comprovar que el valor que es rep com a paràmetre índex és correcte i es troba dins del rang vàlid. A continuació, recupera l'element indicat i crea una rèplica de la coordenada.

	Aquesta rèplica és la que finalment es retorna. Si l'índex no és vàlid, es retorna un objecte CCoord on els atributs tenen el valor per defecte (0).
Visibilitat	Public
Modificadors	None
Paràmetres	- Int: índex
Retorn	CCoord

Nom	**loadTrack**
Descripció	Aquest mètode rep la ruta i el nom d'un fitxer que conté les coordenades que formen la ruta. El mètode l'intenta obrir i, si no sorgeixen problemes, llegeix les coordenades del fitxer i les va desant en el vector.
Visibilitat	Public
Modificadors	None
Paràmetres	- Char*: filename
Retorn	Void

Nom	**size**
Descripció	Retorna el nombre d'elements que conté l'atribut vector.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	- Int: número de coordenades de la ruta.

Nom	**clean**
Descripció	Aquest mètode va accedint a tots els apuntadors que conté el vector i allibera la memòria que tenen reservada. Finalment, neteja tots els elements que conté el vector, és a dir, el deixa buit.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**calculateDistances**
Descripció	Aquest mètode és el responsable de realitzar els últims càlculs que requereix realitzar l'aplicació amb les dades de la ruta (les coordenades): càlcul de la distància, la distància total i la pendent. Ha d'anar recorrent totes les coordenades de la ruta realitzant els càlculs del punt actual respecte de l'anterior i desant els valors en el vector. Per el càlcul de les distàncies, s'utilitza la fórmula del Haversine (veure el document de disseny per obtenir més informació).
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Mapa d'alçades

Nom: CChart
Descripció: es tracta de la implementació d'un nou component en el entorn MFC. Concretament permet la visualització de mapes d'alçades i l'evolució del corredor d'una forma gràfica. És un nou component que modifica el comportament d'un Picture Box.
Visibilitat: public
Modificadors: none

Herència: CStatic
Data especificació: 04/12/2010
Versió: 1.0

ATRIBUTS

Nom	**distance**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Aquest atribut és utilitzat per indicar quina distància de la ruta s'ha realitzat. Serveix per anar marcant la part realitza (evolució) del recorregut.

Nom	**distances**
Visibilitat	Private
Modificadors	<int>: els valors son de tipus enter.
Tipus	std::vector
Descripció	Conté els components X dels punts que s'han d'imprimir per pintar el mapa d'alçades (sobre l'eix horitzontal).

Nom	**elevations**
Visibilitat	Private
Modificadors	<int>: els valors son de tipus enter.
Tipus	std::vector
Descripció	Conté els components Y dels punts que s'han d'imprimir per pintar el mapa d'alçades (sobre l'eix vertical).

Nom	**vector**
Visibilitat	Private
Modificadors	None
Tipus	- CPoint*
Descripció	Es tracta d'un component que permet dibuixar una sèrie de punts encadenats. Ens serveix per dibuixar el mapa d'alçades. Simplement consisteix en afegir les distàncies i les elevacions com a punts en aquest vector.

Nom	**max_distance**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Conté la distància total del recorregut. S'utilitza per determinar el final de la ruta i per encabir i escalar el mapa d'alçades en l'espai disponible.

Nom	**max_elevation**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Conté la elevació màxima del recorregut. S'utilitza per encabir i escalar el mapa d'alçades en l'espai disponible.

Nom	**min_elevation**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Conté la elevació mínim del recorregut. S'utilitza per encabir i escalar el mapa

	d'alçades en l'espai disponible.

Nom	**initialized**
Visibilitat	Private
Modificadors	None
Tipus	Bool
Descripció	Indica si s'han assignat els valors abans de començar a realitzar les operacions i dibuixar els gràfics.

CONSTRUCTORS

Nom	**CChart**
Descripció	Constructor per defecte. Ha d'inicialitzar tots els atributs d'acord al següent criteri: - Distance = 0 - Min_elevation = INT_MAX (constant del C++) - Max_elevation = INT_MIN (constant del C++) - Max_distance = INT_MIN (constant del C++) - Vector = NULL - Initialized = false (inicialment no està res inicialitzat)
Visibilitat	Public
Paràmetres	Void
Retorn	Una instància de la classe.

Nom	**~CChart**
Descripció	Destructor de la classe. Ha de cridar a la funció clean que es descriurà a continuació.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Nom	**paint**
Descripció	Aquest mètode serveix per actualitzar la distància que ja s'ha recorregut i ha de cridar el mètode OnPaint per refrescar el gràfic.
Visibilitat	Public
Modificadors	None
Paràmetres	- Int: distància
Retorn	Void

Nom	**addPoint**
Descripció	Aquest mètode serveix per afegir elements en els vectors elevations i distances. A més, comprova si es tracta d'una elevació màxima o mínima o una distància màxima. En cas de ser un dels tres casos, actualitza l'atribut corresponent.
Visibilitat	Public
Modificadors	None
Paràmetres	- Int: elevation - Int: distance
Retorn	Void

Nom	**clean**
Descripció	Aquest mètode serveix per reiniciar el gràfic. Si l'atribut vector és diferent de NULL, allibera la memòria que aquest té reservat i li assigna NULL.

	A més, buida tots els elements dels vector de distances i elevations. Finalment, posa l'atribut initialized a fals.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**init**
Descripció	Aquest mètode s'utilitza per inicialitzar l'atribut vector. Primerament, s'ha de reservar espai per cada coordenada que contenen els vectors elevations i distances. A continuació, s'han de desar els valors d'elevacions i distàncies al vector creat. Segons les dimensions de l'àrea on s'ha de pintar, aquest mètode escala el gràfic per tal d'encabir-lo en l'espai. Finalment, es posa l'atribut initialized a verdader.
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**OnPaint**
Descripció	Aquest mètode és el responsable de pintar el mapa d'alçades. La seva missió consisteix en: - Pintar la gràfica que representa el mapa d'alçades. - Pintar la línia que indica l'alçada mínima. - Pintar la línia que indica l'alçada màxima. - Pintar l'àrea que indica la distància ja recorreguda (indicada per l'atribut distance de l'objecte). Cal indicar que només es pinta si l'atribut initialized és verdader.
Visibilitat	Protected
Modificadors	Afx_msg (respon a events del MFC).
Paràmetres	Void
Retorn	Void

CSimulbikeDlg

Nom: CSimulbikeDlg
Descripció: aquesta classe implementa el funcionament del prototip que s'està desenvolupant: des de la interfície fins al seu comportament. A més, sincronitza tots els elements per tal de poder realitzar una ruta mitjançant el simulador.
Visibilitat: public
Modificadors: none
Herència: CDialogEx, CSBDialog
Data especificació: 05/12/2010
Versió: 1.0

ATRIBUTS

Nom	**track**
Visibilitat	private

Modificadors	None
Tipus	CTrack
Descripció	Conté les dades que fan referència a la ruta a realitzar.

Nom	**mp**
Visibilitat	Private
Modificadors	None
Tipus	MediaPlayer
Descripció	Es tracta de l'objecte que gestiona la reproducció del vídeo de la ruta.

Nom	**current_node**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Indica a quin node de la ruta es troba l'usuari actualment.

Nom	**max_nodes**
Visibilitat	Private
Modificadors	None
Tipus	Int
Descripció	Ens indica el nombre de nodes que té el recorregut. Usat per determinar el final del recorregut.

Nom	**menuvisible**
Visibilitat	Private
Modificadors	None
Tipus	Bool
Descripció	Indica si la interfície té visible el menú o no.

Nom	**register_values**
Visibilitat	Private
Modificadors	None
Tipus	Bool
Descripció	Aquest flag indica si els valors que es llegeixen del rodet s'han de tenir en compte en el càlculs numèrics de la realització de la ruta o si han de ser ignorats.

CONSTRUCTORS

Nom	**CSimulbikeDlg**
Descripció	Constructor per defecte de la classe. No realitza cap acció ja que les inicialitzacions es deleguen als events MFC.
Visibilitat	Public
Paràmetres	- CWnd* pParent (paràmetre opcional, per defecte és NULL)
Retorn	Una instància de la classe

Nom	**~CSimulbikeDlg**
Descripció	Destructor de la classe. Fa una crida el mètode OnDestroy que es descriurà a continuació.
Visibilitat	Public
Paràmetres	Void
Retorn	Void

MÈTODES

Nom	**OnInitDialog**
Descripció	Quan s'inicia el prototip, cal realitzar una sèrie de tasques: - Inicialitzar i obrir el vídeo de la ruta, mitjançant l'atribut mp (Media-Player). - Muntar la interfície col·locant tots els components gràfics. - Iniciar la connexió al port sèrie amb el rodet i el thread que va tractant les ordres que es reben des del rodet. - Llegir les coordenades de la ruta (atribut track) i realitzar les inicialitzacions requerides. - Inicialitzar tots els labels que mostren valors (funció cleanValues que s'explicarà tot seguit).
Visibilitat	Protected
Modificadors	Virtual
Paràmetres	Void
Retorn	BOOL

Nom	**hideButtons**
Descripció	Segons els valor del flag, mostra o oculta el menú d'accions. És a dir, els botons de Play, Pause, Stop, Reset i exit. Si el flag és verdader, mostra els botons. Sinó, oculta els botons.
Visibilitat	Private
Modificadors	None
Paràmetres	- Bool: flag
Retorn	Void

Nom	**degreeResistance**
Descripció	Aquesta funció rep com a paràmetre un valor que indica la pendent actual del tram que està realitzant l'usuari. A partir d'aquest valor, determina en quin rang es troba segons la taula del document de requeriments. Finalment retorna un enter que indica el grau de resistència que s'ha d'aplicar al rodet.
Visibilitat	Private
Modificadors	None
Paràmetres	- float: pendent
Retorn	Int

Nom	**cleanValues**
Descripció	Aquest mètode és usat per assignar els valors per defecte en els labels de l'aplicació. Aquests son: - Velocitat: 0 km/h - Distancia recorreguda: 0m - Temps: 0s - Distància: longitud de la ruta (distancia absoluta de la darrera coordenada). - Coordenada/node: 1 - Pendent: pendent del primer tram (mirar node de la posició current_node + 1) - Grau de resistència: el valor neutre, 5.
Visibilitat	Private
Modificadors	None

Paràmetres	Void
Retorn	Void

Nom	**OnDestroy**
Descripció	Aquest funció és la encarregada d'alliberar els recursos de l'aplicació: - Posa a CLOSE el flag que controla el bucle del thread que atén les comandes del rodet. - Allibera els recursos de l'atribut mp (MediaPlayer). - Allibera els recursos de l'atribut track (CTrack).
Visibilitat	Public
Modificadors	None
Paràmetres	Void
Retorn	Void

Nom	**OnBnClickedPlay**
Descripció	Aquest mètode ha d'activar la reproducció del vídeo. A més, ha d'habilitar que els valors que es llegeixin a partir d'aquest moment, es tenen en compte a l'hora d'actualitzar els valors de la realització de la ruta.
Visibilitat	Public
Modificadors	Afx_msg (respon a events MFC)
Paràmetres	Void
Retorn	Void

Nom	**OnBnClickedStop**
Descripció	Aquest mètode atura la realització de la ruta. Ha d'aturar la reproducció del vídeo i netejar tots els camps amb els valors de la reproducció de la ruta (funció cleanValues).
Visibilitat	Public
Modificadors	Afx_msg (respon a events MFC)
Paràmetres	Void
Retorn	Void

Nom	**OnBnClickedPause**
Descripció	Aquest mètode realitza una pause en la reproducció de la ruta. A més, posa que la velocitat de reproducció és 0. D'altra banda, atura el registre de noves dades.
Visibilitat	Public
Modificadors	Afx_msg (respon a events MFC)
Paràmetres	Void
Retorn	Void

Nom	**OnBnClickedReset**
Descripció	Aquest mètode és usat per recomençar la ruta sense que l'usuari s'aturi. Posa el vídeo al principi i crida a la funció cleanValues.
Visibilitat	Public
Modificadors	Afx_msg (respon a events MFC)
Paràmetres	Void
Retorn	Void

Nom	**OnUpdateSpeed**
Descripció	Aquest mètode és el que crida el thread per refrescar la velocitat actual del ciclista. El paràmetre WPARAM és un enter que conté la velocitat del rodet. S'ha de dividir

	entre 10 per obtenir la velocitat en Km/h. A continuació s'han de realitzar els càlculs per obtenir el frame rate amb el qual s'ha de reproduir el vídeo (el document de requeriments descriu el funcionament). S'assigna el frame rate al reproductor MediaPlayer mp i s'assigna la velocitat al camp de text de la interfície.
Visibilitat	Public
Modificadors	None
Paràmetres	- WPARAM: wparam (valor que es rep, no té estructura) - LPARAM: lparam (valor que es rep, no té estructura)
Retorn	LRESULT

Nom	**OnUpdateDistance**
Descripció	Aquest mètode és cridat per el thread que atén al rodet. Serveix per actualitzar els valors dels camps de text de la interfície, sempre i quan el valor de register_values sigui verdader. En aquest cas, obté dels paràmetres: - WPARAM: double que conté la distància. - LPARAM: double que conté el temps. La distància es mostra en el camp de text de distància recorreguda. A més, s'actualitza el gràfics d'alçades passant la nova distància a l'objecte CChart. El temps es converteix en segons i es mostra en el camp de text corresponent al temps. A partir de les dades de l'atribut track i la distància recorreguda, es comprova el node actual. El número corresponent a la posició del node es mostra en el camp de text node actual. D'aquest node s'obté el pendent del tram, que es mosta en el camp de text corresponent. Finalment, es crida a la funció de degreeResistance per determinar el grau de resistència que s'ha d'enviar al rodet. Aquest es mostra en el camp de text i s'envia al rodet. Es realitza una darrera comprovació per determinar si l'usuari ha arribat al darrer node. En aquest cas, es finalitza el recorregut mostrant un missatge.
Visibilitat	Public
Modificadors	None
Paràmetres	- WPARAM: wparam (valor que es rep, no té estructura) - LPARAM: lparam (valor que es rep, no té estructura)
Retorn	LRESULT

Bibliografia

En la redacció del present document, s'ha requerit la següent documentació:

- Sistema d'informació geogràfica:
  - http://en.wikipedia.org/wiki/Geographic_information_system

- Estàndard GPX:
  - http://en.wikipedia.org/wiki/GPS_eXchange_Format

- Sistema de Coordenades Universal Transverse Mercator (UTM):
  - http://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system

- Documentació de C/C++, Python i QT:
  - http://www.python.org/
  - http://qt.nokia.com/products
  - http://psyco.sourceforge.net/
  - http://en.wikipedia.org/wiki/Psyco
  - http://qt.nokia.com/products/
  - http://www.cplusplus.com/

- DirectShow:
  - http://msdn.microsoft.com/en-us/library/dd375454(v=VS.85).aspx
  - http://msdn.microsoft.com/en-us/library/dd407299(v=VS.85).aspx
  - http://msdn.microsoft.com/en-us/library/dd318197(v=VS.85).aspx

- Comunicació per port sèrie:
  - http://www.todopic.com.ar/foros/index.php?topic=26052.0

- Estat de l'art dels simuladors de ciclisme actuals:
  - http://www.fitnesssolutions.com.gt/index.php?option=com_content&view=article&id=91&Itemid=88
  - http://www.tacx.com/
  - http://simulation.mirage3d.nl/index.php?option=com_content&task=view&id=15&Itemid=30
  - http://www.sextonivel.com/general/cyberbike-la-bicicleta-estatica-para-wii

- Algoritmos Geométricos em SIG:
  - http://www.dpi.inpe.br/gilberto/livro/geocomp/geometria.pdf