Personatges animats i dinàmica de grups per a videojocs

Adrià Pérez i Rovira

Projecte fi de carrera
Setembre de 2006
Director: Juan Abadía
Enginyeria en informàtica
Universitat Pompeu Fabra

Agraïments

Aquest projecte ha estat possible gràcies a totes aquelles persones que m'han ofert la seva ajuda durant aquests mesos. És per això que vull agrair aquest suport als meus companys de feina, especialment al meu cap Xavier, i també a Sira, Brais i Lluís, a la meva germana, Laia, als meus pares, Josep Maria i Maria, al meu veí etern, Arnau, i als meus actuals companys de pis, Hèctor i Pol.

També m'agradaria dedicar aquest projecte a en Tomeu Penya, ídol inamovible de l'imaginari col·lectiu català.

Resum

La simulació del comportament dinàmic d'individus agrupats no és un camp excessivament nou ni desconegut. De fet, tant el món de la investigació com el de la industria de l'entreteniment l'han abordat amb èxit durant els últims anys. Cal destacar els treballs de recerca publicats per *Craig W. Reynolds* durant els últims 20 anys, a partir de la seva primera publicació al respecte de 1987 *Flocks, Herds, and Schools: A Distributed Behavioral Model*, que es pot considerar pionera a l'hora d'abordar científicament els problemes derivats de la dinàmica de grups.

Cal destacar que l'industria de l'entreteniment ha utilitzat algoritmes d'aquest àmbit tant per a videojocs com per al món del cinema. En aquest últim cas, el problema acostuma a ser molt tancat i es busquen solucions concretes per a simular comportaments concrets. Difícilment veurem en una pantalla de cinema el resultat de modelitzar un animal amb diferents estats que configurin el seu comportament, ja que segurament només veurem un comportament específic per a una situació concreta, com per exemple una manada de nyus fugint del seu depredador (tal i com es pot veure a la pel·lícula *El rei lleó*). I no un modelatge complert dels diferents tipus de comportament de l'animal.

En aquest projecte es pretén fer una aproximació a aquest problema, enriquint la solució amb funcionalitats diverses. Aquestes funcionalitats comporten una sèrie de restriccions i requeriments que obliguen a que la implementació es mogui en diferents àmbits de l'algorítmica. És per això que no només caldrà modelitzar el comportament dels individus en camp obert i en grups reduïts, sinó que la solució final haurà de ser una composició d'algoritmes de cerca de camins, tècniques de renderitzat, sistemes d'interacció amb l'usuari, control del temps i comportament dinàmic de grups.

Així doncs, la correcta combinació de tots aquests algoritmes són el principal repte a aconseguir, ja que de la seva harmonia dins de la solució en depèn directament el resultat final.

Índex

1. Introducció 9
1.1 - Motivació 9
1.2 - Context 9
1.3 - Objectius del projecte 10

2. Anàlisis de requeriments 11
2.1 - Requeriments generals 11
2.2 - Requeriments funcionals 11
2.3 - Requeriments no funcionals 13

3. Conceptes 17
3.1 - El sistema de locomoció 17
3.2 - El model de moviment en grup 20
3.3 - La cerca de camins 21
3.3.1 - L'algoritme emprat: l'A 22*
3.3.2 - L'heurística emprada 24

4. Eines i llibreries 25
4.1 - Llenguatge de programació 25
4.2 - Llibreries de programació 25
4.2.1 - Llibreries utilitzades 25
4.2.2 - Llibreries descartades 26

5. Anàlisis i Disseny 29
5.1 - L'escenari 31
5.1.1 - Les forces de repulsió i les col·lisions 34
5.1.2 - Els camins i checkpoints 35
5.2 - El grup 37
5.2.1 - Les característiques del grup 37
5.2.2 - Els atributs del grup 37
5.2.3 - Els estats del grup 39
5.3 - L'individu 45
5.3.1 - Les característiques de l'individu 45
5.3.2 - Els atributs de l'individu 47
5.3.3 - Els estats de l'individu 48
5.3.4 - Les 5 forces que es poden aplicar a un individu 53

6. Implementació 59
6.1 - Metodologia utilitzada 59
6.2 - Etapes del projecte 60
6.3 - Estructura i funcionament de l'aplicació 62
6.3.1 - La visualització 66
6.3.2 - El control del temps. Divisió dels blocs de pintat i de càlcul 75
6.3.3 - La integració de l'algoritme de cerca de camins 76
6.3.4 - La selecció interactiva dels grups i l'assignació d'objectius 77
6.4 - Els arxius d'escenari 77

7. Conclusions i treball futur 81
7.1 - Validació dels requeriments funcionals 81
7.2 - Validació dels requeriments no funcionals 83
7.3 - Limitacions i propietats del model aplicat 84
7.3.1 - Limitacions de l'escenari 84
7.3.2 - Limitacions de l'algoritme de cerca de camins 84
7.3.3 - Limitacions de la composició dels grups 85

7.3.4 - Limitacions de les propietats dels individus 85
7.3.5 - L'estabilitat dels grup 86
7.3.6 - L'homogeneïtat en els grups 87
7.4 – Ampliacions possibles 87

8. Annexos 89
8.1 - Pseudocodi de l'A* 89
8.2 - Pseudocodi del Diagonal Shortcut 90
8.3 - Exemple d'un fitxer d'escenari 90

9. Bibliografia 95

1. Introducció

Avui dia l'animació informàtica la podem trobar en qualsevol racó de la nostra societat. La trobem en anuncis publicitaris, pel·lícules d'èxit, videojocs, presentacions arquitectòniques i en un llarguíssim etcètera d'àmbits d'oci i professionals.

És per això que la simulació del comportament d'un grup d'individus (persones, animals, vehicles motoritzats, etc.) pot ser aplicat en infinitat de camps amb objectius tan dispars com fer més creïble una escena d'una pel·lícula on surt una manada de cérvols en desbandada o enriquint un entorn 3D on es mostren habitatges per vendre per part d'una immobiliària.

Aquest treball de final de carrera intenta aprofundir en el tema de la dinàmica de grups basant-se en un model determinat, per tal de crear una aplicació on es pugui simular aquest comportament dinàmic de grups relativament petits d'individus.

1.1 - Motivació

El creixent realisme gràfic de les escenes virtuals generades per ordinador corre el risc de donar una sensació de pobresa si aquestes no aporten una impressió de fluïdesa en les seves simulacions. És inqüestionable que una imatge estàtica aporta molta informació a la persona que l'estigui visualitzant, però resulta encara més indiscutible que si a aquesta es converteix en una animació que evoluciona al llarg del temps aporta molta més informació i sensació de realisme.

Així doncs, una simulació en temps real del comportament dinàmic d'individus pot resultar útil en una gran varietat de projectes diferents. Per exemple, un grup de persones desplaçant-se per una ciutat pot afegir sensació de realisme en la presentació animada d'un edifici. També cal destacar que la simulació del comportament de grups és útil per a entendre el comportament de certes situacions, de manera que una bona simulació permet treure conclusions de com construir un escenari.

Per altra banda, la dinàmica de grups es pot enfocar des de molts punts de vista diferents. Una aproximació possible podria ser l'ús de la disciplina de la dinàmica de fluids, àmpliament estesa per entendre el comportament de líquids i gasos en fricció amb les parets del conducte pel que viatgen. Aquest enfoc no té cabuda en aquest projecte ja que estudia el comportament d'un conjunt, obviant per complet els elements que el formen.

1.2 - Context

Gairebé tothom ha vist en algun moment de la seva vida la utilització de la dinàmica de grups, tot i que segurament no s'hagi aturat a pensar com s'havia generat aquella animació. Alguns exemples podrien ser els nyus corrent en desbandada en la pel·lícula *El Rei Lleó* o el

comportament d'alguns personatges en coneguts videojocs com el *Call of Duty* o l'*Operation Flashpoint*, només per citar-ne alguns.

Però no només existeixen animacions merament decoratives fruit de l'ús d'algoritmes de dinàmica de grups. L'ús d'aquestes simulacions s'ha aplicat a multitud de camps per tal de poder estudiar certs comportaments d'objectes o individus desplaçant-se per un escenari virtual. D'aquesta manera es poden prendre decisions sobre l'estructura de l'escenari abans de ser construït i evitar així possibles problemes d'aglomeracions en carreteres, passadissos, etc.

Un exemple prou il·lustratiu podrien ser els entorns de simulació on s'estudia com situar sortides d'emergència en espais tancats. De fet, ja existeixen programes capaços de simular allaus humanes intentant escapar d'un perill a través de les sortides d'emergències d'un local. Aquests estudis es poden convertir en un increment de la seguretat d'espais públics o privats, ja que al basar-se en les dades extretes d'aquestes simulacions aquests recintes tancats poden comptar amb sortides d'emergència molt més eficients, capaces de desallotjar una sala en flames en pocs segons.

1.3 - Objectius del projecte

L'objectiu d'aquest projecte és la creació d'una aplicació que simuli en temps real el comportament de diferents individus que es mouen agrupats en conjunts relativament petits (entre 2 i 10 individus, aproximadament) de forma fluida i sense una jerarquització del grup. És a dir, evitant crear un líder i que la resta de membres es limiti a seguir-lo.

L'escenari de l'aplicació serà un món tridimensional finit amb obstacles i parets on hi apareixeran, a part dels grups, enemics que afectaran directament al comportament dels grups fent-los fugir o apartar-se, de manera que els grups es pugin desmembrar i els individus s'hagin de reagrupar posteriorment, afegint realisme al projecte.

Aquest escenari serà una superfície plana on es simularà el comportament més comú que coneixem: el desplaçament per damunt d'una superfície, desestimant simular el comportament d'objectes voladors o que es desplacin per un fluid. D'aquesta manera es pretén no desviar els esforços del projecte en la transformació d'energia potencial en cinètica o en un sistema de locomoció complex. De fet, és força comú en el món dels videojocs que l'escenari es presenti a l'usuari en tres dimensions, mentre que la lògica de l'aplicació treballi només en dues.

Així doncs, un sistema de locomoció relativament senzill i un marc de visualització sense una excessiva riquesa visual serviran de marc per tal de poder aplicar el model estudiat de comportament dinàmic de grups.

2. Anàlisis de requeriments

El primer pas que s'ha de fer a l'hora d'iniciar qualsevol projecte mínimament complex és emmarcar i acotar el terreny on volem fer néixer, créixer i evolucionar el projecte en si mateix. Per a aquest fi resulta imprescindible detallar els requeriments del projecte i definir de forma clara quin és l'objectiu final.

Per a tota aplicació podem distingir tres tipus de requeriments:

- *Requeriments funcionals*: Són aquells que descriuen les funcionalitats que ha d'aportar el sistema i com aquest reaccionarà davant d'entrades concretes i situacions particulars.

- *Requeriments no funcionals*: No detallen el sistema sinó que defineixen restriccions que tindran les funcionalitats degut a elements externs (temps, eficiència, ...)

- *Requeriments de domini*: Són conseqüència del domini en el que s'emmarca el projecte.

Seguidament passem a detallar els requeriments de la nostra aplicació per tal de definir clarament què ha de fer el nostre projecte i com ho ha de fer.

2.1 - Requeriments generals

El sistema té un objectiu final clar: Crear una aplicació capaç de simular el comportament dinàmic de diferents conjunts d'individus (animals, persones, vehicles, etc.) movent-se de forma coherent dins d'un escenari complex (amb parets i obstacles). Aquesta simulació ha de ser tot lo amplia possible, és a dir, ha de permetre recrear diferents situacions que, tot hi basar-se en una algorítmica similar, produeixin situacions absolutament dispars com ho serien una família passejant per un carrer transitat o una manada d'animals salvatges formada per adults i cadells (que evidentment corren menys i tenen més por als enemics), desplaçant-se per trobar aliment i fugint dels seus depredadors.

Evidentment aquest objectiu principal resulta massa ambigu i per a això es fa imprescindible definir de forma concreta i pragmàtica tot un conjunt de requeriments molt més específics i concrets. De tal manera que, a l'hora d'implementar l'aplicació, el codi segueixi un esquema perfectament definit.

2.2 - Requeriments funcionals

Seguidament es descriuen les funcionalitats que s'espera que tingui la simulació, detallant les característiques que hauran de tenir els diferents elements que formen part de l'escenari simulat.

- Grups formats per un mínim d'entre 2 i 10 individus

Per a poder considerar un grup cal com a mínim treballar amb dos individus que tinguin una estreta relació a l'hora de moure's per l'escenari. Així doncs, tot i que sembli una obvietat, cal poder definir grups compostos amb dos o més individus. Evidentment una simulació amb grups només de dos individus quedaria molt pobre, i és per això que per tal de poder simular una major varietat de situacions l'aplicació haurà de poder treballar amb grups més grans. Més concretament, haurà de poder simular el comportament de grups formats per un mínim d'entre 2 i 10 individus.

- Interacció de l'usuari: Modificació d'objectius en temps real

Per tal de poder crear infinitat de situacions diverses resulta imprescindible que l'usuari de l'aplicació pugui alterar l'estat de l'escenari i els individus. Per a tal fi, s'oferirà la possibilitat de seleccionar i modificar els objectius de cada grup amb el ratolí, creant una situació diferent i irrepetible a cada execució.

- Varietat en les propietats dels individus

Cada individu es podrà comportar de manera molt diferent a la resta i no tindran tots idèntiques propietats. Alguns dels paràmetres configurables més importants seran l'acceleració, la velocitat màxima, la velocitat de gir o la resistència al pànic provocat pels enemics. Aquesta varietat resultarà imprescindible per tal d'aconseguir simular situacions on, per exemple, un grup hagi d'esperar a un individu molt més lent que la resta, escenes on grups aniran molt més ràpids que d'altres, etc.

- Escenaris complexos amb parets, passadissos, habitacions, etc.

No només els individus podran tenir diferents aspectes àmpliament configurables sinó que l'escenari en sí haurà de poder simular gran varietat d'entorns. Des d'escenaris molt oberts amb pocs obstacles fins a complicats laberints que posin a prova la capacitat del grup de mantenir-se unit en espais estrets i enrevessats.

- Escenaris definits en arxius per a carregar directament

La forma més ràpida i senzilla per a treballar amb diferents escenaris complexos és la creació de diferents arxius fàcilment modificables per tal de que la creació d'una situació concreta en l'aplicació sigui quelcom tan senzill com escollir un arxiu a carregar. D'aquesta manera es podrà generar un conjunt d'arxius amb diferents propietats per tal de simular les diferents situacions possibles. Cal remarcar que aquests arxius han de ser àmpliament configurables i no només limitar-se a guardar posicions i informació geomètrica de l'escenari. En altres paraules, han de contenir tota la informació d'un escenari en un moment concret, des del número d'enemics fins a la velocitat de rotació de tots els individus en un instant determinat i totes les seves característiques (acceleració, velocitat màxima, nivell de pànic, etc.)

- Distribució i alineament dels individus en assolir l'objectiu

Els grups es mouran lliurement per l'escenari de forma que l'àrea que ocupin s'adapti als diferents obstacles que hi puguin trobar. Tot i això, cal considerar la possibilitat de que un cop aturats en el seu objectiu hauran de situar-se de forma coherent, tots mirant cap a la mateixa direcció (que l'usuari podrà definir) i no només aturar-se descoordinadament.

- Detecció de col·lisions i evitar obstacles

Per tal de crear un moviment creïble dels individus, caldrà implementar algoritmes de detecció i evitació d'obstacles. D'aquesta forma els individus no xocaran directament contra la paret, sinó que abans d'arribar-hi giraran i/o frenaran per tal de crear un moviment suau i realista que evitarà aquests possibles xocs amb les parets.

- Interacció amb enemics

Un escenari on només hi hagi parets i grups d'individus que es limitin a desplaçar-se evitant els obstacles crearia una situació poc realista que no permetria mostrar els diferents estats que es volen simular, com per exemple el de fugides descontrolades i les seves posteriors reagrupacions. Buscant aquesta major varietat de situacions s'ha decidit afegir enemics que es moguin per l'escena de forma que els diferents grups i individus puguin entrar en estat de pànic i conseqüentment els intentin evitar.

- Possibilitat de fugides massives

Resultarà interessant implementar la possibilitat de provocar fugides en massa degudes a que si un individu entra en estat de pànic (provocat per la proximitat d'un enemic) aquest influenciï als seus companys, fent-los augmentar el seu nivell de pànic i fins i tot provocant la fugida d'aquests. Això permetrà simular, de forma encara més real, el comportament de diferents grups d'animals que estan fortament condicionats per la resta del grup.

2.3 - Requeriments no funcionals

- Eficiència en temps i 'frames' per segon:

Al tractar-se d'una aplicació en temps real on es mostren per pantalla els resultats de la simulació amb gràfics tridimensionals, resulta evident que aquesta visualització ha de tenir un mínim de qualitat gràfica. Obviant la qualitat gràfica entesa com a individus realistes o amb gran nombre de polígons que queda fora dels objectius d'aquest projecte. Així doncs, resta quantificar la qualitat considerant el número de *frames* per segon. En aquest aspecte, i considerant que l'ull humà difícilment és capaç de percebre el parpelleig per sobre de **25 *frames* per segon (*fps*)**, s'ha estimat aquesta xifra com el llindar mínim per considerar la simulació com a acceptable en termes d'eficiència.

Cal considerar també que el numero d'*fps* depèn de la complexitat de l'escenari i de la situació en cada instant, a part de que el *hardware* també influeix dràsticament a l'hora

d'obtenir uns bons resultats. És per això que cal especificar amb major detall en quins casos aquest nivell de 25 *fps* és realment el llindar a tenir en consideració.

Primerament caldrà especificar amb quin tipus d'escenaris ens trobarem a l'hora de realitzar simulacions. Si tenim en compte que el principal factor que comporta un augment lineal de consum de processador és la quantitat d'individus que hi ha dins l'escena, el factor a determinar és justament aquest número.

Estimem que per tal de simular gairebé la totalitat de situacions possibles, exceptuant aquelles realment massificades, calen uns 100 individus (cal apuntar que la repartició equitativa o no en grups similars és gairebé indiferent ja que la gran majoria de càlculs es realitzen de forma individual i no en grup).

Tot i que en principi el número d'enemics serà molt inferior al número d'individus, caldrà especificar una xifra per acotar encara més les condicions d'avaluació del rendiment. Al igual que la grandària de l'escenari, tot i que aquesta no afecti gairebé per a res al rendiment.

Així doncs, la situació a avaluar serà la següent:

Situació mínima a avaluar	
Número d'individus	**100**
Número d'enemics	**5**
Mida de l'escenari (en cel·les)	**10.000**

Figura 1. Condicions de la situació a simular.

Finalment definirem el hardware amb que s'ha de provar aquesta eficiència, intentant escollir una configuració que s'apropi el màxim possible al que es consideraria un ordinador de sobretaula actual, no gaire desfasat (a data de 2006), però tampoc sent aquest d'última generació.

De forma arbitraria s'ha definit:

Configuració de Hardware aproximada	
Generació d'ordinador	Intel **Pentium IV** (FSB 800)
Velocitat del processador	**2.600 MHz**
Memòria RAM	**1.024 MB** (DDR2 a 400 MHz)
Targeta gràfica	Nvidia **GeForce 5900** (128MB RAM DDR2)

Figura 2. Condicions per provar els requeriment no funcionals.

- Consum de Recursos (Memòria):

Un altre aspecte a considerar és el consum de memòria RAM que pot fer l'aplicació. Per a tal consideració cal tenir en compte el número d'escenes que s'estan simulant de forma simultània. Evidentment, tot i que l'aplicació permetrà simular diferents escenaris a la vegada, l'ús normal serà d'una sola escena i per tant els criteris d'avaluació suposaran que només hi ha un escenari executant-se simultàniament.

Així doncs, considerant que la limitació es deguda a la memòria RAM i estimant que la configuració definida en la taula de la figura 2 és l'estàndard per a fer el conjunt de proves de rendiment sembla lògic i coherent afirmar que l'aplicació no hauria de consumir més de la meitat d'aquesta memòria per tal de garantir el correcte funcionament d'altres aplicacions i del propi sistema operatiu.

Així doncs, el límit de memòria el situem en **512 MBytes** de RAM. S'ha de reconèixer que aquest límit és una xifra molt gran i seria recomanable no abusar de la memòria RAM per tal de facilitar l'execució de l'aplicació en ordinadors menys potents, però en tot cas mai ha de superar aquest llindar, inclús en escenaris molt més grans i complexos que els definits anteriorment a la figura 1.

3. Conceptes

En aquest capítol es detallen els algoritmes i les idees consultades i escollides més rellevants del projecte. Una primera idea extreta de la bibliografia consultada ha estat la del sistema de locomoció presentat per ***Craig Reynolds*** en l'article ***Steering Behaviors For Autonomous Characters***[1] a partir de la qual s'ha anat construint tot el sistema de moviment que impregna el projecte. D'aquesta mateixa publicació s'ha extret un model de moviment per a individus que es desplacen en grup.

Una segona idea important, tot i que en aquest cas ha estat en forma d'algoritme, és la de l'algoritme **A*** per tal de cercar el camí més curt entre dos punts de l' escenari, el qual conté obstacles i parets.

3.1 - El sistema de locomoció

Per tal de simular el moviment d'un objecte de forma que quedi creïble s'ha d'actualitzar la posició del mateix de forma constant i coherent, depenent de tota una sèrie de paràmetres més o menys complexos. En aquest apartat descriurem quin sistema de locomoció hem emprat, basant-nos en una lleugera modificació del model presentat per *Craig Reynolds* en la publicació *Steering Behaviors For Autonomous Characters*[1] el qual es basa en aplicar un conjunt de forces al centre de masses d'aquest objecte.

Primer de res, cal definir què és i què fa el sistema locomotor. Segons aquest publicació el moviment d'un individu es pot dividir en tres nivells o capes, els quals són:

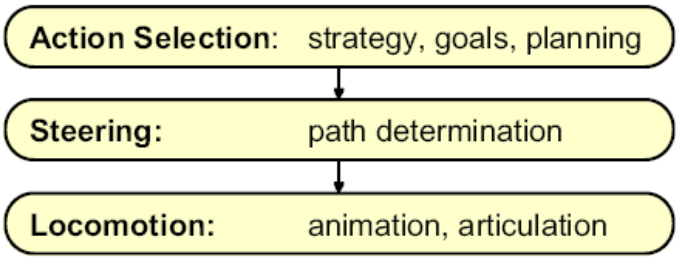


Figura 3. Moviment subdividit en capes, extret de l'article Steering Behaviors For Autonomous Characters *[1].*

- Selecció d'acció (*Action Selection*)

És en aquesta capa on l'individu pren les decisions oportunes depenent de la situació en la que es trobi. Així doncs, si un individu està massa allunyat del grup i decideix reagrupar-se és en aquest nivell on es produeix el canvi d'estat, delegant tota la resta a les dues capes inferiors.

- **Conducció (*Steering*)**

En aquesta capa intermitja és on es defineix el comportament de l'individu, així doncs si a un individu se li assigna un objectiu, és en aquesta capa on es realitzen el diferents càlculs i subdivisió d'objectius per tal de aportar-li la informació oportuna a la capa inferior, encarregada de la locomoció.

- **Locomoció (*Locomotion*)**

És el més baix dels tres nivells, encarregat de realitzar el canvi de posició i orientació del objecte seguint les instruccions de la conducció (capa superior).

Un exemple per tal de visualitzar aquests tres nivells podria ser un grup de *cowboys* intentant conduir un ramat de vaques. Les ordres del cap dels genets són la capa superior, ja que és qui varia els objectius temporals del grup, donant ordres a la resta de companys. La capa de conducció (l'intermitja) es pot interpretar com les diferents accions que realitzen els genets per tal d'assolir l'objectiu assignat (dirigir-se a una vaca aïllada, rodejar obstacles, reagrupar a la vaca, etc.). Aquest genet utilitza diferents sistemes per tal d'indicar que ha de fer el seu cavall (ordres orals, regnes, etc.), el qual converteix en moviment les instruccions rebudes, representant la capa inferior: la locomoció.

Per tal de convertir les ordres en moviment, el sistema de locomoció es basa en un centre de masses al qual se li apliquen unes forces (força cap a l'objectiu, força de repulsió d'una paret, etc.) i recalcula la posició del objecte de forma periòdica, també en funció del temps i l'estat del individu (velocitat, acceleració, massa, etc.).

Aquest model es caracteritza per ser molt poc costós computacionalment comparat en un model físic més realista. Per exemple, la simplificació de tractar l'objecte com a un punt fa que l'individu tingui un moment lineal (velocitat), però no un moment rotacional que sí que poden considerar altres models. En aquest aspecte es pot considerar irreal, ja que qualsevol objecte físic no té un radi de 0 i per tant té un moment d'inèrcia, però aquesta pèrdua de realitat ve compensada per un model molt generalista que pot simular de forma força aproximada el moviment de gran varietat de vehicles diferents, des d'avions fins a submarins.

La idea central és descompondre aquesta força, resultat de la suma ponderada de forces que exerceixen sobre l'individu, en una component paral·lela a la direcció cap on mira l'individu i una altra perpendicular a la primera.

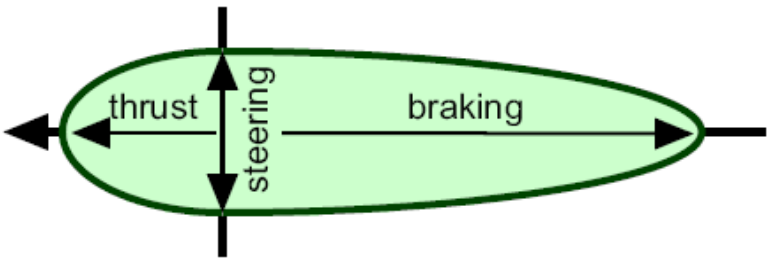


Figura 4. Descomposició de forces segons l'article
Steering Behaviors For Autonomous Characters *[1]*.

La força alineada amb la direcció és l'acceleració (o desacceleració) i la perpendicular és la quantitat de gir que ha de sofrir l'individu. Aquestes forces es poden truncar de forma que cada individu tingui unes propietats de màxima acceleració o màxim gir que limitin la magnitud d'aquests dos vectors, condicionant el moviment resultant.

Aplicant aquests dos vectors (provinents d'una capa superior) l'individu renovarà els 3 paràmetres que defineixen el seu estat de moviment:

- La posició, condicionada per l'escenari.

- La velocitat, limitada per les pròpies característiques de l'individu.

- La direcció, qualsevol angle sobre el pla.

És apropiat indiciar, que per tal d'aconseguir un moviment realista, és molt aconsellable que la màxima força de frenada sigui varies vegades superior a la màxima acceleració, de manera que l'individu requereixi menys espai per a frenar que no pas per a accelerar.

Per a efectuar aquesta locomoció el primer pas és calcular la nova direcció de l'individu segons el vector de gir. Per a fer aquest càlcul es pren el vector de direcció (que està normalitzat) de l'individu i se li suma el vector de gir (perpendiculars entre si). Aquest nou vector es normalitza i es guarda com el nou vector de direcció de l'individu.

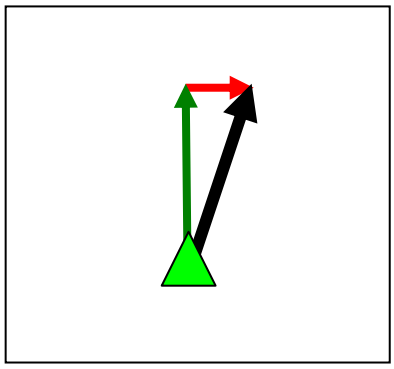


Figura 5. Càlcul del vector de direcció a partir del anterior vector de direcció i la força perpendicular.

Seguidament es calcula la nova velocitat, sumant el mòdul del vector d'acceleració (positiu si accelera, negatiu si frena) a la velocitat de l'individu i truncant-lo entre 0 (parat) i la màxima velocitat de l'individu.

Finalment a la posició actual se li suma el vector de direcció multiplicat per la nova velocitat i els temps recorregut i s'obté la nova posició.

Aquest model de locomoció permet un tipus de moviment en el qual el gir es independent de la tracció (acceleració i frenada), de forma que un individu pot girar sense necessitat de desplaçar-se frontalment.

3.2 - El model de moviment en grup

En el mateix article presentat per *Craig Reynolds, Steering Behaviors For Autonomous Characters*[1], d'on s'ha extret el model de locomoció dels individus exposat en l'apartat anterior també hi podem trobar un model de vehicle que es desplaça en grup. Aquest vehicle es caracteritza per utilitzar 3 conceptes en forma de força que s'apliquen al seu centre de masses per tal de desplaçar-se de forma agrupada. Aquestes 3 característiques que defineixen el concepte de desplaçar-se en grup són:

- Cohesió

- Separació

- Alineament

Evidentment aquests 3 conceptes són perfectament combinables, i de fet han de estar degudament combinats i ponderats per tal de crear un model realista de moviment en grup.

3.2.1 Cohesió

La cohesió és la força que manté units els diferents individus, atraient-los cap al centre del grup al qual pertanyen. Per calcular el centre del grup només cal fer la mitjana de les posicions de tots els individus del grup, incloent la del propi individu.

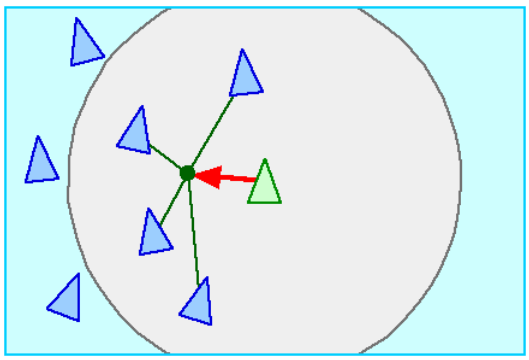


Figura 6. Força de cohesió.

Aquesta força és constant i no depèn de la distància a la que estigui situat el centre del grup, de forma que els individus situats en la perifèria (sobretot en casos on els grups siguin molt nombrosos) no exerceixin una gran pressió a la resta d'individus situats en el centre.

3.2.1 Separació

Per a contrarestar la força de cohesió i evitar que tots els individus es dirigeixin cap al centre del grup sense deixar espais entre ells és imprescindible simular una força de separació que creixi a mesura que s'apropin 2 individus. Quant més gran sigui aquesta força més espai hi haurà entre els diferents individus del grup.

Per calcular aquesta força s'ha d'iterar amb la resta d'individus del grup i anar sumant la força de repulsió amb cada un de forma ponderada, depenent de la distància.

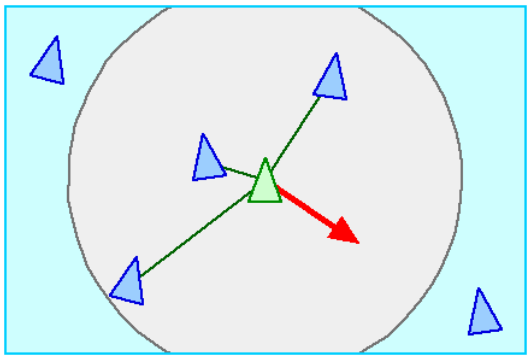


Figura 7. Força de separació.

A diferencia de la força de cohesió, aquesta força sí que varia en la seva magnitud. Concretament ho fa de forma inversament proporcional a la distància amb els altres individus del grup. D'aquesta forma, quan més a prop estiguin dos individus, més gran serà aquesta força.

3.2.1 Alineament

Quan es parla d'alineament tothom pensa en que tots els individus tendeixin a mirar en la mateixa direcció, i tot i que és cert, el concepte d'alineament també fa referència a que els diferents individus tinguin tendència a homogeneïtzar la seva velocitat, el qual impedeix que els individus més lents quedin despenjats del grup i els més ràpids s'escapin.

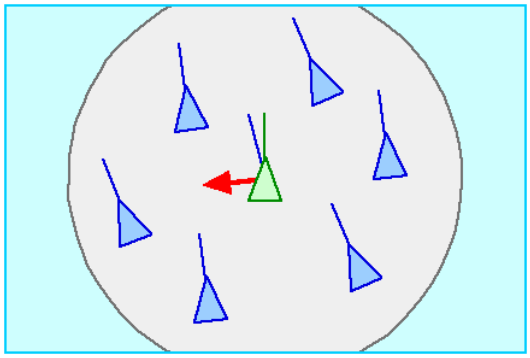


Figura 8. Força d'alineament.

3.3 - La cerca de camins

Si es vol simular el comportament d'individus mínimament complexes amb capacitat de prendre decisions, i que no només es limitin a rebre estímuls i avançar despreocupadament sense considerar l'escenari (passadissos, parets, concavitats, etc), cal integrar a l'aplicació algun algoritme capaç d'evitar i/o rodejar tots els possibles obstacles que es puguin trobar dispersos per l'escenari.

Evidentment, una possible opció era restringir el tipus i la complexitat dels escenaris per tal d'evitar afegir algun algoritme de cerca de camins, els quals es caracteritzen per ser força costosos computacionalment. Però si s'aplicaven aquestes restriccions l'escenari quedava molt empobrit ja que només podríem treballar amb terrenys sense obstacles o amb obstacles circulars (ja que l'ús de forces de repulsió de les parets permetria rodejar-los).

És per això que s'ha considerat oportú i imprescindible afegir un algoritme d'aquest tipus en l'aplicació el qual, un cop integrat, permet despreocupar-se de les limitacions a l'hora de crear escenaris. Només s'ha de tenir en consideració que l'escenari haurà de poder-se representar en un graf, de forma que l'espai quedarà subdividit en cel·les uniformes.

Inclús es podrien afegir costos a les diferents cel·les que subdivideixen el terreny per tal de simular diferents tipus de superfícies, per tal de simular diferents tipus de terreny amb característiques heterogènies depenent de la seva dificultat de ser travessades. Un possible exemple seria poder crear cel·les amb vegetació (difícils de travessar) o sense (molt més fàcils de creuar). Aquests costos només són una possible aproximació, ja que un algoritme d'aquest tipus permet afegir a cada cel·la tota la informació que es desitgi, de tal manera que es podria calcular un cost per a cada cel·la depenent de la quantitat d'individus que hi hagi, del temps que fa que hi ha passat per ella un enemic, etc.

3.3.1 - L'algoritme emprat: l'A*

L'algoritme escollit per tal de fer la cerca de camins ha estat l'A* (*A estrella*). Presentat per *Peter E. Hart, Nils J. Nilsson i Bertram Raphael* l'any 1968, aquest algoritme, combinació de cerca en amplada i cerca en profunditat, es classifica dins del grup dels *algoritmes de cerca en grafs* ja que treballa directament sobre estructures de dades en forma de graf.

Característiques i propietats

L'A* es caracteritza perquè permet trobar el camí de menor cost entre dos nodes d'un graf amb costos, sempre hi quant es compleixi una sola restricció. Aquesta és:

- L'heurística ha de ser admissible: el cost per arribar a un node objectiu estimat no pot ser superior al cost real.

Si no es compleix aquesta condició l'algoritme no serà capaç de trobar el camí de menor cost, tot i que si que assegura trobar-ne un, si és que aquest existeix. En aquest cas l'algoritme passa a anomenar-se simplement A.

Així doncs, les propietats d'aquest algoritme són:

- Es un algoritme complert: (en cas d'existir una solució, sempre la trobarà).

- Si per tot node n, es compleix g(n) = 0, l'algoritme efectua una cerca voraç.

- Si per tot node n, es compleix f(n) = 0, l'algoritme efectua una cerca de cost uniforme no informada.

Funcionament

El funcionament de l'algoritme és força senzill i es caracteritza per fer una cerca basada tant en la funció heurística, com en el cost real del recorregut.

Així doncs, utilitza una funció d'avaluació *f(n) = g(n) + h(n)*, on *g(n)* representa el cost del camí recorregut per arribar a aquest punt i *h(n)* el valor heurístic del node a avaluar.

L'A* manté dues estructures contenidores de nodes que podem denominar *open* i *closed*.

- *Open*: Es tracta d'una cua de prioritats ordenada pel valor f(n) de cada node.

- *Closed*: Una llista desordenada on es guarden els nodes visitats.

A cada iteració de l'algoritme, s'expandeix el primer node que estigui en la llista de nodes oberts, i en el cas de no ser un node objectiu, calcula la f(n) de tots els seus fills, els afegeix a la llista d'oberts i mou el node avaluat a la llista de nodes visitats.

Complexitat computacional i en memòria

La complexitat computacional de l'algoritme depèn en gran mesura de la qualitat de l'heurística emprada. En el pitjor dels casos (amb una heurística triada de molt baixa qualitat) la complexitat serà exponencial. En el cas de treballar amb una heurística optima, l'algoritme s'executarà en temps lineal.

Per a que això succeeixi s'ha de complir que,

$$h(x) \leq g(y) - g(x) + h(y)$$

Figura 9. Condició per treballar amb una heurística òptima.

el qual implica que l'heurística és optima, ja que no sobreestima el cost real que tindrà el recorregut.

Per altra banda, un dels grans defectes de l'A* és l'espai en memòria que requereix, ja que ha d'emmagatzemar tots els possibles nodes futurs de cada estat, de manera que la quantitat de memòria que utilitza es exponencial amb relació amb la mida del graf (l'escenari dividit en cel·les). Afortunadament, en cap dels possibles escenaris amb que ens trobarem sobrepassa les 10.000 cel·les, el qual es perfectament emmagatzemable amb la quantitat de memòria que disposa un PC de sobretaula actual.

Aquest projecte treballa amb un tipus de problema basat en espais relativament reduïts, i per tant no representa cap inconvenient aquesta possible limitació de l'A* en quant a requeriments de hardware. Però tot i això, es important apuntar que existeixen diferents variacions d'aquest algoritme com poden ser el *RTA*, l'IDA** o *l'SMA**. Tots ells caracteritzats per oferir diferents solucions al problema de la memòria.

3.3.2 - L'heurística emprada

L'heurística utilitzada és simplement el cost que s'estima que tindrà el camí real fins l'objectiu. Ja que l'expansió de cada node es pot fer en les 8 direccions (2 verticals, 2 horitzontals i 4 diagonals) el cost de canviar de cel·la depèn dels 2 tipus de moviment que hi ha, un per als desplaçaments verticals i horitzontals, i un altre per a les diagonals amb un cost aproximat d' (arrel de 2) vegades el cost d'un desplaçament no diagonal.

Per tal de calcular aquesta heurística s'utilitza el codi detallat en l'annex, el qual estima la distància entre dues cel·les com el camí més curt possible si es combinen els dos tipus de moviment possibles. Anant primer en línia recta i posteriorment en diagonal, tal i com es pot veure en la següent imatge:

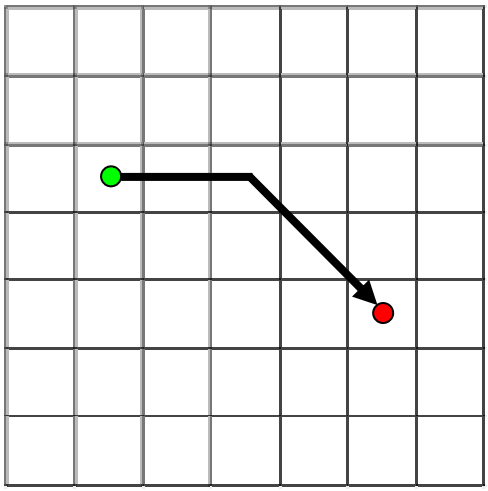


Figura 10. Exemple de l'heurística emprada.

Aquest sistema per calcular l'heurística s'anomena *Diagonal Shortcut (Drecera Diagonal)*. I és la millor heurística possible ja que és la mes ajustada al cost real sense sobreestimar-lo. El qual faria perdre la completitud al algoritme. De fet, aquesta heurística és exactament el cost real que tindria el camí en el cas de que no hi haguessin obstacles.

4. Eines i llibreries

El primer pas per a realitzar el software d'aquest projecte final de carrera ha estat definir el llenguatge de programació i les llibreries de suport a utilitzar tenint en compte els requeriments tant funcionals com no funcionals descrits anteriorment.

Tot i semblar un procés senzillament i merament teòric ha resultat molt més complex de l'esperat degut a que s'han fet diverses proves per tal de poder avaluar de forma objectiva quina era la millor combinació per tal d'atacar l'objectiu de realitzar un software capaç de respectar tots els requeriments funcionals i no funcionals del projecte.

4.1 - Llenguatge de programació

El llenguatge emprat finalment ha estat **C++** ja que permet generar un codi estructurat en classes i orientat a objectes, el qual facilita tant l'escriptura com la comprensió del mateix. Evidentment hi ha molts més llenguatges orientats a objectes com per exemple Java, però aquest ha estat descartat degut a que no és tan eficient com C++ en temps d'execució i la seva difícil integració amb llibreries gràfiques potents. Un altre punt a favor de la utilització de C++ ha estat la possibilitat d'incloure la llibreria **MFC** (*Microsoft Foundation Classes*) que ofereix un esquelet de finestres, diàlegs i menús interactius molt intuïtiu per a qualsevol usuari de *Windows* i molt fàcil d'integrar i adaptar en qualsevol aplicació un cop s'hi ha treballat prèviament i s'entén el seu funcionament.

4.2 - Llibreries de programació

A l'hora d'implementar el projecte s'ha considerat convenient utilitzar un parell de llibreries, aquestes són les *MFC* i *OpenGL*. Tot seguit es justifica aquesta decisió i s'explica també les altres llibreries considerades que finalment han estat descartades.

4.2.1 - Llibreries utilitzades

- MFC (Microsoft Foundation Classes)

Les Microsoft Foundation Classes son l'API proporcionada per *Microsoft* per tal de crear aplicacions sobre el sistema operatiu *Windows* en C++. Aquest conjunt de classes facilita i estandarditza la feina de programació per a qualsevol programador que les domini mínimament. En elles s'hi inclou un gran nombre de funcionalitats, entre les que destaquen la facilitat de crear una barra de menú personalitzada, un potent gestor de finestres i diàlegs, estructures per a un precís control del temps i una fàcil programació per a controlar la interacció de l'usuari a través del ratolí i el teclat.

Les aplicacions creades amb les MFC es basen en l'estructura **Document - Vista**, la qual es caracteritza per separar i encapsular la part de dades de la de visualització, permetent tenir diferents vistes d'un conjunt de dades (en el nostre cas una escena) i tenir diferents conjunts de dades en una sola aplicació corrent de forma simultània.

És per tot això que ens ha semblat una elecció molt útil per tal de crear una interfície de l'aplicació amigable i fàcil de fer servir i programar.

- OpenGL

OpenGL és una llibreria de gràfics 3D àmpliament utilitzada per a infinitat d'aplicacions diferents i compatible amb un gran nombre de plataformes, des de jocs a programes d'arquitectura o generació d'efectes especials per a pel·lícules. Desenvolupada per *Silicon Graphics, Inc.* es caracteritza per la seva qualitat visual propera a l'obtinguda amb tècniques de Ray-Tracing però amb el gran avantatge de ser molt més ràpida.

La combinació de rapidesa i mal·leabilitat han estat els argument per escollir-la enfront d'altres biblioteques gràfiques com per exemple *VTK (Visual ToolKit)*, molt més orientada a visualitzacions estàtiques (representacions mèdiques, per exemple) que no pas a escenes dinàmiques que és en el que es centra aquest projecte.

4.2.2 - Llibreries descartades

Tot i que pot semblar que des d'un principi s'ha tingut molt clar el conjunt de llibreries a utilitzar no ha estat així i s'han requerit diverses proves per tal de trobar la configuració més apropiada per desenvolupar el projecte. Una bona forma de justificar l'elecció final sempre és comentar què s'ha descartat i per quins motius. Per això mateix seguidament llistem les llibreries provades i descartades durant les primeres fases de l'elaboració del projecte.

- Irrlicht

Aquesta potent llibreria de visualització es caracteritza per la seva eficiència i rapidesa. A part d'aquestes excel·lents prestacions de rendiment també aporta mètodes interessants com per exemple la detecció de col·lisions o la possibilitat de carregar models animats amb suma facilitat.

Però tot i aquests importants avantatges que en teoria haurien de reduir les hores de programació destinades a crear l'aplicació ha estat descartada per un sol motiu: És molt difícil de manipular i modificar per tal d'assolir els objectius del projecte, on la qualitat gràfica, com ja hem apuntat, no és una prioritat i si que ho és la interactivitat.

- OpenSteer

Quan es parla de llibreries destinades a realitzar simulacions de comportaments dinàmics *OpenSteer* és la més indicada per a tal objectiu. Facilita enormement la codificació d'agents dinàmics i compta amb nombrosos exemples amb diferents models de moviment implementats. A la mateixa vegada aporta un entorn de visualització molt

complert amb detalls visuals remarcables com el de dibuixar la trajectòria dels diferents individus de forma que es vagi difuminant progressivament o mètodes per a afegir text a qualsevol part de l'escena tridimensional per tal de debugar l'aplicació de forma intuïtiva.

Però tot aquest ventall de facilitats tenen un costat negatiu que ha resultat ser el mateix que el de la llibreria *Irrlicht*: És tan complerta que dificulta molt afegir-hi funcionalitats teòricament simples, en aquest cas ha resultat impossible implementar la interacció de l'usuari amb el ratolí.

El seu problema ha estat, doncs, que no es tracta només d'una llibreria que faciliti la tasca d'implementar models dinàmics sinó que va més enllà i trepitja el territori de la visualització i la interactivitat, on clarament s'ha demostrat insuficient en quant a mal·leabilitat i adaptació als nostres requeriments funcionals.

5. Anàlisis i Disseny

En aquest capítol es pretén fer una aproximació a la solució proposada per tal de simular el comportament de varis conjunts d'individus que es mouen de forma coordinada i en grup per escenaris complexos, els quals inclouen diferents obstacles i enemics que també es desplacen evitant els obstacles.

Aquesta primera aproximació pretén explicar de forma més o menys genèrica la solució proposada, exposant les idees principals que hi ha implícites en el codi que efectua la simulació del comportament dinàmic i obviant els aspectes més tècnics i concrets de la solució implementada finalment.

A l'hora d'abarcar un problema d'aquesta magnitud és molt important tenir molt clar quins son els grans bloc que el formen, per tal de poder atacar cada una d'aquestes parts de forma individual. Com es pot veure en el esquema de la figura 11, l'estructura contenidora de tots els elements (grups, individus, parets, etc.) és l'estructura anomenada *World*, la qual es caracteritza per tenir 3 elements que formen la totalitat de l'entorn a simular.

- L'escenari

- Una llista de grups d'individus (controlables per l'usuari)

- Una llista de grups d'enemics (autònoms completament)

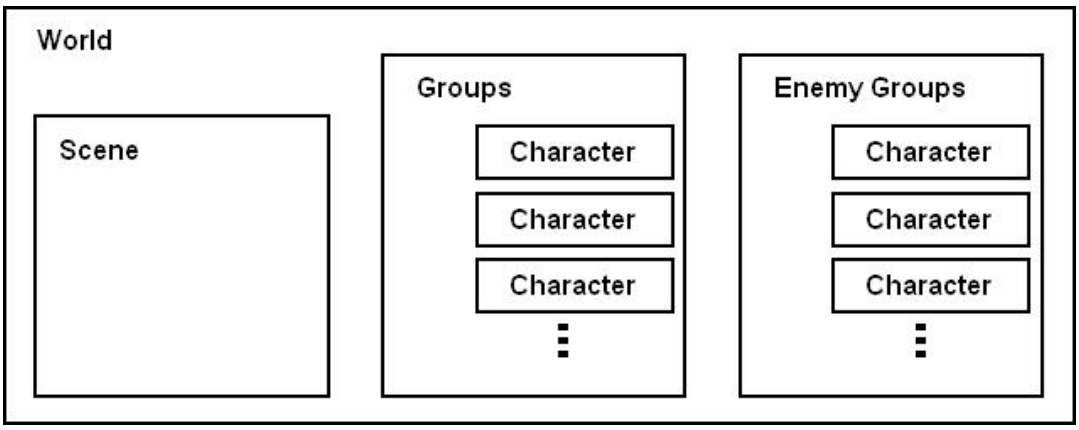


Figura 11. Estructura de l'entorn de simulació.

Tal i com es pot veure a la figura 11 aquests grups contenen una llista dels individus que els formen, però això ja resulta irrellevant per a l'estructura que representa la totalitat de la simulació (*World*) ja que són els grups els únics que poden treballar amb els seus individus. Al igual que l'escenari és qui encapsula les diferents cel·les en les quals es subdivideix.

Així doncs, podem dir que el món de la nostra simulació és un món format per un escenari (dividit en cel·les), un conjunt de grups (formats per diferents individus) i un altre conjunt de grups (evidentment, també formats per individus), aquest cop identificats com a enemics. Aquests tres blocs interactuen entre sí per tal de generar una simulació en temps real dels diferents individus desplaçant-se a través de l'escenari, evitant obstacles i fugint dels enemics.

De cadascun d'aquest blocs s'espera un comportament determinat, el qual tots junts formaran el comportament esperat de la simulació. Aquests diferents rols són:

- *Grups d'individus* – Es desplaçaran entre dos punts de l'escenari escollits per l'usuari, evitant obstacles (parets i altres individus) i fugint dels enemics en cas de que s'aproximin massa.

- *Grups d'enemics* – Es desplaçaran de forma constant entre diferents punts de l'escenari escollits periòdicament de forma aleatòria, evitant tots els obstacles que hi hagi.

- *Escenari* – Obstaculitzarà el desplaçament de tots els individus i les seves parets generaran una força de repulsió en cas que un individu se li aproximi. També proporcionarà una llista de punts intermedis (*chekpoints*) que definiran el camí mes curt per anar d'un punt A a un punt B. En altres paraules: oferirà un algoritme de cerca de camins que retorni el camí més curt entre dos punts.

Resulta obvi veure una estreta relació funcional entre els dos tipus de grups, el d'individus (controlats per l'usuari) i el d'enemics. De fet justament aquesta similitud de comportament ha estat la principal motivació per a unificar els 2 tipus de grups (i d'individus) en una sola classe que s'inicialitza en un estat diferent i que té petites variacions en el comportament. De fet d'ara en endavant quan es parli d'un grup, es podrà considerar que parlem d'un grup tant d'individus com d'enemics, de forma indistinta.

Per altra banda, tot aquest sistema d'interacció entre l'escenari, els individus controlats per l'usuari i els enemics dóna com a resultat una direcció desitjada i no tot un moviment d'un model complex, amb forma i volum. Per a això, cal un sistema de locomoció capaç de convertir el vector resultant en un desplaçament de l'objecte, de manera que quedi natural i realista.

Per fer aquesta conversió s'ha decidit utilitzar el sistema de locomoció proposat per *Craig Reynolds* en l'article *Steering Behaviors For Autonomous Characters*[1] exposat en el capítol de Conceptes. Aquest model es basa en considerar els objectes com un punt infinitesimal en el qual s'aplica la força. El model també té un vector de direcció que indica la seva orientació i a partir del qual es pot limitar la capacitat de gir per tal de simular l'efecte de la inèrcia. Un dels aspectes que no considera és la capacitat de fricció del moviment, el qual emula limitant la velocitat màxima. Tampoc té en consideració el moment angular ja que treballa sobre un punt i no sobre un volum.

Però tot i aquestes limitacions, que fan que el sistema de locomoció es consideri irreal, aquest model permet calcular les actualitzacions de posicions dels individus amb un cost computacional molt baix i de forma molt senzilla, de manera que un cop elaborat, les energies del projecte es poden destinar a aprofundir més en l'anàlisi a més alt nivell del comportament dels individus desplaçant-se en grup i interactuant amb les parets de l'escenari, sense que el sistema de locomoció alteri els experiments.

Un cop ja hem fet aquesta visió general, entrarem a estudiar l'estructura dels diferents elements esmentats de forma individual.

5.1 - L'escenari

L'escenari és la representació de la superfície per on es desplacen els diferents individus. Bàsicament es pot considerar que es tracta d'un rectangle situat en un món tridimensional, dividit en diferents cel·les quadrades, les quals poden tenir diferents propietats (cost de creuar-les, tipus de terreny, etc.).

Com es tracta d'una superfície plana, ens limitem a fer un moviment en dues dimensions, obviant per complet la vertical, que només s'utilitza en la visualització amb una finalitat purament estètica.

La decisió de treballar sobre una superfície plana és deguda a que la majoria de moviments que poden ser interessants d'estudi es produeixen sobre una superfície, com per exemple les persones, els cotxes o animals terrestres. Descartarem doncs, la simulació de vehicles que volin o que viatgin a través d'un medi líquid.

Per altra banda, la utilització de superfícies no planes aportaria major realisme a la simulació, però a un elevat preu en complexitat del sistema locomotor, de la cerca de camins òptims i d'una gran varietat d'aspectes. Aquest preu s'ha considerat massa elevat per tenir cabuda en aquest projecte, però pensant en una possible adaptació i ampliació s'ha preparat l'estructura de manera que els càlculs interns de manipulació de vectors i coordenades es fa en tres dimensions, el que permetria fer variacions en punts aïllats de l'aplicació sense necessitat de fer una completa readaptació.

Per altra banda, la subdivisió en cel·les homogènies permet crear parets, habitacions, passadissos i tota una complexa geometria de forma realment simple, intuïtiva i senzilla, marcant unes caselles com a transitables (buides) i altres com a obstacles (plenes). Però les avantatges d'usar una graella uniforme per a l'escenari són moltes més, per exemple permet emprar algoritmes de cerca de camins dissenyats per a ser usats amb grafs per tal de trobar el camí més curt entre dos punts, ja que un rectangle subdividit en cel·les d'igual mida es pot concebre com un graf on cada cel·la és un node que està connectat amb un cert cost amb les cel·les veïnes, que a la vegada són també nodes.

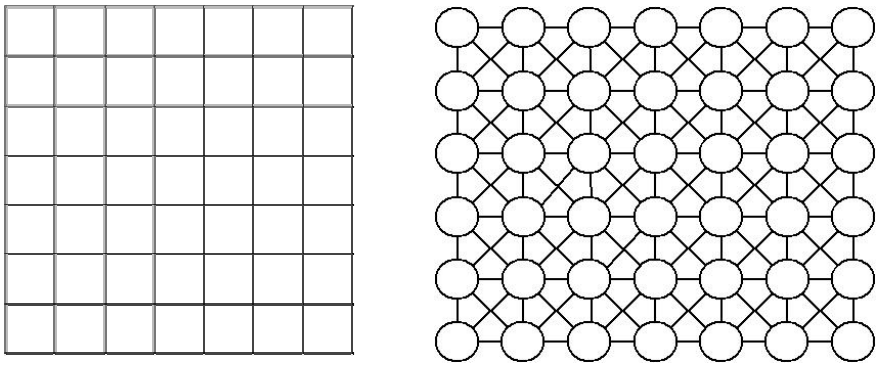


Figura 12. Un escenari interpretat com a graella (esquerra) o com a graf (dreta).

Una altra virtut d'aquest model per representar l'escenari és el baix cost computacional que comporta calcular la força de repulsió amb les parets en un punt determinat o la detecció de col·lisions en un segment de recta, ja que només cal mirar el tipus de cel·la on estiguem situats i

les veïnes. Amb una cost gairebé nul de convertir un vector de posicions absolutes en coordenades dins l'espai de cel·les, i a l'inrevés.

A la figura 13 es pot apreciar que en cas de que no hi hagués obstacles entre el punt A i B, només caldria mirar en unes poques cel·les si estan buides (quelcom força habitual) per tal de detectar les col·lisions Per fer aquest càlcul l'aplicació utilitza un algoritme recursiu que primerament mira quin és el punt de sortida (quin costat i en quin punt exacte) de la recta i la cel·la. Un cop calculat aquest punt es crida al mateix mètode passant-li per paràmetre el punt d'entrada de la recta amb la nova cel·la. Com a condició de sortida, es mira si la cel·la que estem mirant és la final (la cel·la final és aquella que conté el punt de destí), en cas de ser així és que no hi ha cap col·lisió i per tant sabem que no cal aplicar un algoritme de cerca de camins per tal d'assolir l'objectiu.

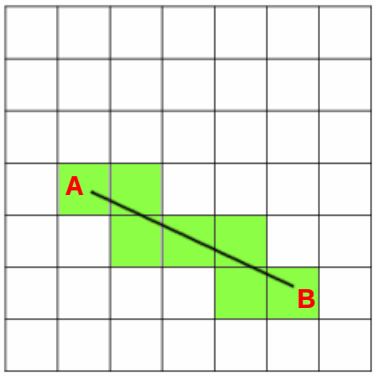


Figura 13. Cel·les visitades per mirar si hi ha obstacles entre el punt A i el B.

En el cas de voler calcular la força de repulsió que exerceixen les parets sobre els individus en un punt de l'escenari lliure d'obstacles els avantatges són exactament els mateixos, només cal mirar les 8 cel·les veïnes i només es calcula la força de repulsió en cas de que alguna de les cel·les veïnes no estigui buida, tal i com s'aprecia a la figura 14.

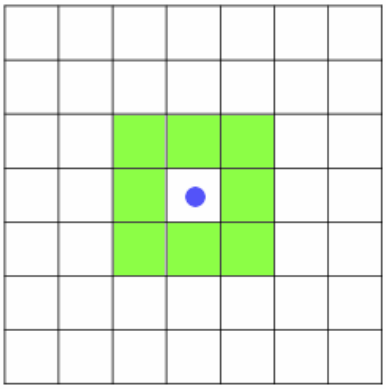


Figura 14. Cel·les visitades per obtenir la força de repulsió sobre un individu.

Difícilment es pot arribar a una solució on tot siguin avantatges i l'elecció de concebre l'escenari com una graella formada per cel·les uniformes té certes limitacions i contrapartides. La principal limitació d'aquest model és que la geometria es basa en una unitat mínima, la cel·la, que a la vegada ha de ser suficientment gran per a que hi càpiga un individu ja que si no

fos així un individu estaria en més de 4 cel·les a la vegada i els beneficis d'usar aquesta graella uniforme desapareixerien immediatament ja que un individu no podria passar per un passadís format per una cel·la i l'algoritme de cerca de camins seria molt més complex i costós. Així doncs, l'escenari consta d'una severa restricció: l'ample mínim d'un obstacle serà de la mida d'una cel·la, que a la vegada és més gran que un individu

A la figura 15 es pot apreciar una paret d'amplada mínima en comparació amb l'individu que té al costat, segons la relació escollida. L'individu fa dues unitats de diàmetre mentre que la paret té 4 unitats d'amplada.

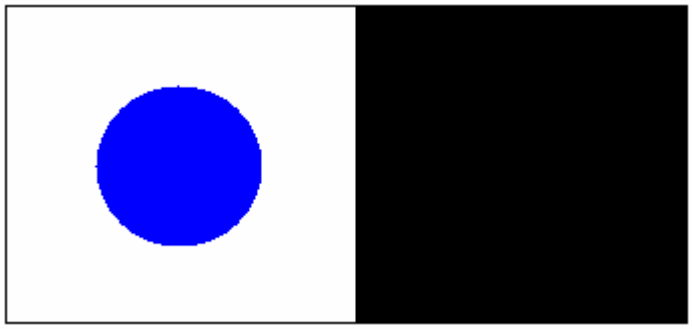


Figura 15. Mida d'un individu (blau) comparada amb una paret d'amplada mínima (negre).

L'altra limitació important d'aquest model és que es treballarà amb un nombre finit de tipus de cel·les, ja que cada tipus de cel·la s'ha de tractar de forma independent per tal de calcular les forces de repulsió, la cerca de camins i les col·lisions. I per tant, la varietat de peces amb les que compondre l'escenari sol ser força reduïda.

En la nostra simulació hi ha 6 tipus de cel·les diferents, les quals es poden classificar en 3 grups diferents, segons les seves característiques més importants.

- *Cel·les buides:* Son plenament transitables ja que representen un espai buit.

- *Cel·les completament plenes:* Al contrari que les buides, no tenen cap espai per on transitar, representen una paret sòlida en tot l'espai de la cel·la.

- *Cel·les rodones:* Aquest tipus de cel·la s'usa per tal d'evitar angles rectes en els extrems dels objectes. Com poden haver-hi quatre posicions d'una cantonada n'hi ha 4 diferents dins d'aquest grup. Cada una es considera que té el centre de la circumferència que dibuixen en un dels 4 angles de la cel·la. Aquest tipus de cel·la es considera plena quan s'utilitza l'algoritme de cerca de camins, però els individus poden travessar sense problemes l'espai que queda fora de la paret.

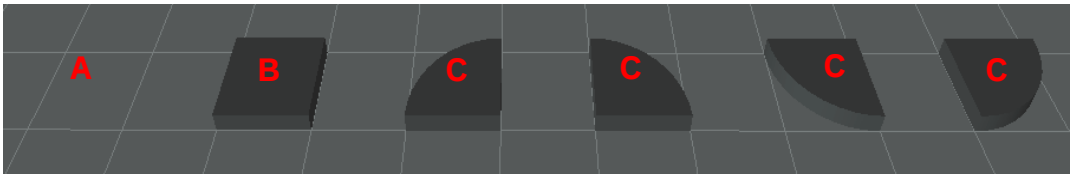


Figura 16. Diferents tipus de cel·les.

A la figura 16 es poden veure els diferents tipus de cel·la que pot contenir un escenari. La cel·la etiquetada com ‘A’ és una cel·la buida, la ‘B’ és una cel·la plena i la ‘C’ és una cel·la rodona, en les 4 orientacions possibles.

5.1.1 - Les forces de repulsió i les col·lisions

Com ja hem apuntat, una de les propietats de l’escenari és la d’exercir una força de repulsió, segons la seva composició, sobre els individus. Aquest vector de força s’aplicarà (junt a altres forces) sobre el centre de masses de l’individu, de forma que tendeixin a apartar-se de les parets i deixin un petit espai, simulant un comportament realista, ja que gairebé la totalitat de vehicles tendeix a separar-se lleugerament dels objectes amb els que poden col·lidir.

La direcció d’aquesta força de repulsió sempre serà perpendicular a la paret propera a l’individu, i la seva magnitud dependrà de la distància que hi hagi entre el centre de masses de l’individu i el punt més proper de la paret. D’aquesta forma si un individu és molt proper d’una paret el seu moviment estarà molt condicionat per aquesta repulsió, a diferencia d’un altre individu més apartat, tendint a situar-se a una distància natural on les diferents forces que actuïn sobre l’individu s’anul·lin a excepció de la component paral·lela al propi vector de direcció (cap on està orientat l’individu).

Aquest vector de força és independent de cada cel·la, per tant cada individu que actualitza la seva posició ha de testejar les seves 8 cel·les veïnes i anar sumant els vectors de repulsió de cadascuna.

Tot i que realment hi ha 6 tipus de cel·les, ja hem dit que es poden agrupar en 3 tipus, cel·les buides, cel·les completament plenes i cel·les rodones. Així doncs hi ha dues formes de calcular la repulsió (direcció i magnitud del vector), ja que les cel·les buides no exerceixen repulsió.

Com ja hem dit, la direcció del vector és perpendicular a la paret, de forma que en les cel·les plenes (que són quadrades) només cal mirar si és la superior, inferior, dreta o esquerra. En cas que no sigui cap d’aquestes, la direcció de la força de repulsió és el vector diagonal entre el centre de la cel·la i el centre de masses de l’individu, a la figura 17 es poden apreciar aquests vectors en un exemple.

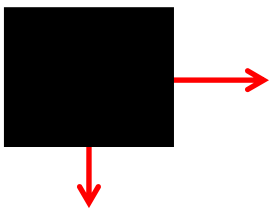


Figura 17. Forces de repulsió perpendiculars a les parets d’una cel·la quadrada.

En cas que es tracti de cel·les amb parets arrodonides, només cal buscar el centre de l’arc que dibuixa la paret i des d’allí calcular la direcció del vector fins al centre de masses de l’objecte tal i com s’aprecia a la figura 18.

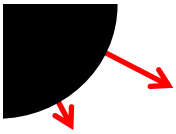


Figura 18. Forces de repulsió perpendiculars a la paret d'una cel·la rodona.

A part d'aquestes forces de repulsió de l'escenari també s'hauran de computar les forces de repulsió entre individus, per tal que no xoquin de forma indiscriminada, però això s'explica amb més detall a l'apartat d'individus dins d'aquest mateix capítol, on es llisten les diferents forces que afecten a un individu.

En alguns casos, aquesta força de repulsió no és suficient per a evitar un xoc entre un individu i una paret, de forma que s'ha de controlar que la distància entre la paret i l'individu no sigui mai inferior al radi de l'individu (que es considera un cercle) i en cas de detectar una violació d'aquesta norma caldrà reajustar la posició, separant-lo la distància suficient, en la direcció perpendicular a la paret,.

5.1.2 - Els camins i *checkpoints*

Tot i la gran utilitat i els avantatges d'usar les forces de repulsió per tal que els individus evitin col·lidir amb les parets dels escenaris, cal alguna cosa més en casos que la geometria de l'escenari comporti obstacles mínimament complexos, com per exemple passadissos sense sortida o parets en zig-zag.

Així doncs, un individu implementat de forma que reaccioni amb l'escenari només tenint en compte les forces de repulsió es veurà constantment incapacitat per assolir el seu punt de destí si aquest es troba darrere d'alguna paret que cal rodejar. Per tal de dotar els grups (i conseqüentment els individus) d'aquesta major capacitat per a poder arribar al seu punt de destí independentment de la geometria de l'escenari, s'ha utilitzat un algoritme de cerca de camins, el qual retorna una llista amb totes les cel·les ordenades per on passa el camí més curt. Aquest algoritme és l'A*, explicat en el capítol de Conceptes.

Evidentment aquesta llista conté informació redundant, ja que no és necessari conèixer totes i cadascuna de les cel·les per on passa el camí més curt, sinó que és suficient només emmagatzemant el centre de les cel·les on el camí fa un git. Aquest punts són els anomenats chekpoints (o punts intermedis de control). Així doncs, l'algoritme de cerca de camins està complementat per un petit algoritme que transforma la llista de cel·les en una llista de posicions tridimensionals. Aquest algoritme és força simple i es caracteritza per anar recorrent el camí més curt en sentit invers (des del destí fins a la posició inicial). Cada cop que detecta que el camí gira, emmagatzema la posició del centre de la cel·la.

Tal i com es pot veure a la figura 18, els chekpoints són els objectius del grup de forma temporal, de manera que si un individu es dirigeix de forma ordenada d'un chekpoint a un altre arribarà al seu destí final a través del camí més curt.

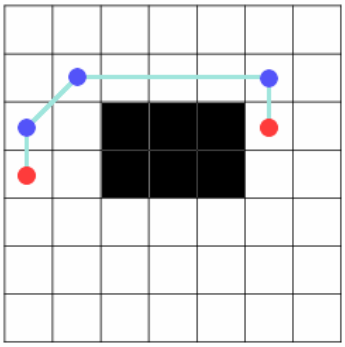


Figura 18. Camí més curt entre dos punts (vermells), representat per 3 checkpoints *(punts blaus).*

La successió de *checkpoints* són doncs el que defineix el camí a seguir per tal d'arribar des d'un punt d'origen a un de destí minimitzant la distància a recórrer.

Aquest ús dels camins facilitats per l'escenari són útils tant per a un grup (quan ha d'assolir un objectiu situat darrere d'algun obstacle) com per a individus (en situacions on no avancen en grup, reagrupant-se per exemple).

5.2 - El grup

En un projecte titulat 'Personatges animats i dinàmica de grups' resulta evident que la pedra angular són els grups. A parir d'aquesta idea s'ha desenvolupat la resta del projecte, i per tant, no és descabellat dir que en aquest apartat hi ha l'essència del treball realitzat.

5.2.1 - Les característiques del grup

Abans d'entrar en detall sobre els atributs del grup, els estats que defineixen el comportament i altres aspectes més tècnics, cal descriure quin és el concepte de grup que s'ha utilitzat en aquest projecte.

Els grups que es pretenen simular són grups no jerarquitzats, sense un líder ni una entitat que prengui decisions conjuntes. Sinó que un grup és, simplement, un conjunt d'individus que intenten desplaçar-se de forma unida i cohesionada. Cada individu pren les seves pròpies decisions tenint en compte la informació conjunta del grup.

Dit en altres paraules, no hi ha res ni ningú que dirigeixi el grup, sinó que cada individu actua sent conscient que forma part d'un grup. Per tant, tenen mecanismes de desplaçar-se cap al centre del grup, detectar si estan aïllats, etc.

Tot i aquesta individualitat els individus comparteixen informació i per tant, poden realitzar accions coordinades. Un exemple: quan la dispersió augmenta massa el grup canvia d'estat a reagrupar-se i, immediatament, tots els individus canvien el seu propi estat a reagrupar-se. En canviar tots d'estat de forma sincronitzada, tots els individus inicien el reagrupament a la vegada, i fins que el grup no compleixi unes certes condicions (que la dispersió sigui acceptable) tots els individus mantindran aquest estat.

5.2.2 - Els atributs del grup

Hi ha dos tipus de grups, per una banda tenim els grups d'individus que controla l'usuari i per una altra banda tenim els grups d'enemics. Tot i que tenen un comportament diferent (els primers fugen dels segons). Les seves similituds són majors que les diferencies. És per això que a l'hora d explicar-los s'expliquen com un de sol, ja que realment és una única classe on depenent de l'estat es pot saber quin dels dos rols juga un grup.

Un grup es caracteritza per contenir un conjunt d'individus (amb les seves propietats) i un conjunt de propietats comunes (centre de masses del grup, velocitat mitjana dels individus, etc.). Seguidament, es detallen els principals atributs d'un grup:

- *state* (Enter): Variable on s'emmagatzema l'estat del grup, pot tenir 5 valors diferents. 4 en el cas de tractar-se d'un grup d'individus controlats per l'usuari (avançar, formar, reagrupar o fugir) o 1 de sol si es tracta d'un grup d'enemics.

- *center* (Vector): Es tracta de la mitjana de les posicions de tots els individus que no estan en estat de pànic. En cas que tots els individus entrin en estat de pànic, s'emmagatzemarà l'últim centre de masses vàlid.

- *weighedCenter* (Vector): Aquest centre és el resultat de calcular el centre de masses de forma ponderada, de manera que els individus més lents (amb menor velocitat màxima) tindran major pes a l'hora de calcular aquest centre de masses virtual.

- *dispersion* (Coma flotant): És la distància mitjana entre el centre del grup i cada individu. En aquest cas tampoc es té en compte els individus que estan en estat de pànic.

- *checkPoints* (Llista de Vectors): En aquesta llista s'emmagatzemen els punts intermedis retornats per l'algoritme de cerca de camins, per tal de poder arribar a l'objectiu evitant i rodejant els obstacles.

- *nCheckPoints* (Enter): És la quantitat de punts intermedis que li falten al grup per tal d'arribar a l'objectiu.

- *leaderCheckPoint* (Enter): A cada actualització es mira quin dels individus està més avançat (li queden menys *checkpoints* per a arribar a l'objectiu) i es guarda el mínim número de punts intermedis que li quedin a qualsevol individu del grup.

- *finalTarget* (booleà): Cert en cas que el camí descrit per la llista de *checkpoints* arribi a l'objectiu, o fals en el cas contrari. Si és fals voldrà dir que un cop arribat al final del camí s'haurà de recalcular el camí fins l'objectiu, des de la nova posició.

- *waitingTarget* (Vector): En aquesta variable s'emmagatzema l'objectiu final designat per l'usuari, per tal de saber on s'ha de dirigir el grup si el camí calculat no hi arriba.

- *forward* (Vector): Aquest vector és la mitjana de tots els vectors de direcció de l'individu. Es pot concebre com la direcció cap on avança el grup.

- *speed* (coma flotant): És la velocitat mitjana dels individus que formen el grup (exceptuant els que estan fugint).

- *selected* (booleà): Per saber si un grup està seleccionat, aquest conté una variable que ho controla. Cert si el grup està seleccionat (l'usuari pot modificar-ne l'objectiu) o fals en cas contrari.

- *formAlignment* (Vector): A un grup se li pot assignar un objectiu, i una direcció sobre el pla per a que s'alineïn tots els individus quan hagin arribat al punt de destí. Aquesta direcció és modificable per l'usuari en temps real i s'emmagatzema en aquesta variable anomenada *formAlignment*.

- *characters* (Llista de *Characters*): En aquesta llista hi trobem, de forma desordenada, els diferents individus que formen el grup.

De tot aquest llistat d'atributs només hi ha 2 que l'usuari pugui modificar directament fent servir d'interfície de l'aplicació: el punt de destí (*target*) i el vector d'alineament (*formAlignment*). La resta d'atributs són propietats internes del grup que, evidentment, es veuen afectades per les accions de l'usuari en temps real però de forma indirecta, com per exemple la posició, la velocitat o l'estat del grup.

5.2.3 - Els estats del grup

De tots aquests atributs, el més important és l'*state* (estat). Aquesta variable conté la informació sobre en quin dels 5 estats possibles es troba el grup. Aquesta importància és deguda a que el grup ha estat pensat per a que tingui 4 tipus de comportament diferents: *avançar*, *formar*, *reagrupar-se* i *escapar*. El cinquè estat possible és el dels enemics, *intimidar*, el qual està pensat exclusivament per als enemics, i un grup normal (controlat per l'usuari) no hi pot accedir mai.

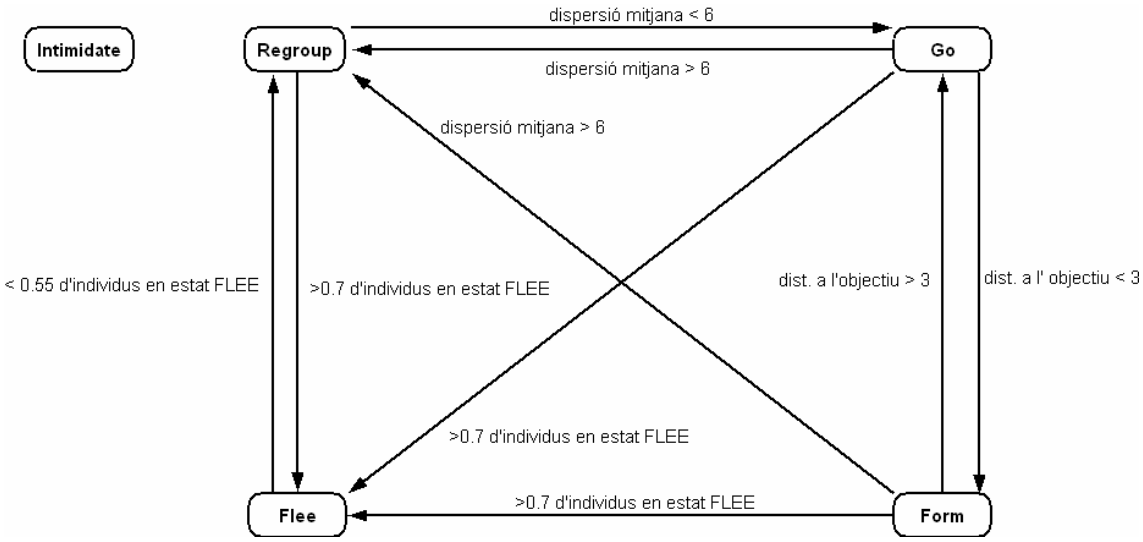


Figura 19. Diagrama d'estat d'un grup.

Com es pot veure a la figura 19, hi ha un estat aïllat (*Intimdate*) reservat per als enemics, mentre que un grup controlat per l'usuari pot anar canviant entre els altres 4 segons les condicions de sortida de cada estat. Seguidament detallem què fa, com s'hi arriba i com es surt de cada un dels estats.

- Avançar (Go)

Descripció:

El grup està unit i es desplaça de forma cohesionada cap a l'objectiu assignat. Un cop arribi al seu destí el grup passarà a l'estat **Formar**.

Condicions de sortida:

- Si la proporció d'individus en estat de pànic és superior al llindar assignat (0.7) el grup canvia d'estat a **Fugir**.

- Si la distància mitjana dels individus amb el nucli supera el llindar assignat (6 cops el diàmetre d'un individu) el grup passa a l'estat **Reagrupar**.

- Si el centre del grup està a menys distància de l'objectiu final que el llindar assignat (1.5 vegades el diàmetre d'un individu), aquest passa a l'estat **Formar**.

Accions:

Entrada:

- Res.

Bucle:

- Es recalculen les propietats del grup (centre, dispersió, etc.).
- S'actualitzen els individus que el formen.
- Es comproven les condicions de sortida de l'estat.

- Formar (Form)

Descripció:

El grup està unit i proper a l'objectiu. Tots els individus que el formen s'aturen i s'alineen segons la direcció preestablerta per l'usuari.

Condicions de sortida:

- Si la proporció d'individus en estat de pànic és superior al llindar assignat (0.7) el grup canvia d'estat a **Fugir**.
- Si la distància mitjana dels individus amb el nucli supera el llindar assignat (6 vegades el diàmetre d'un individu) el grup passa a l'estat **Reagrupar**.
- Si el centre del grup està a major distància del objectiu final que el llindar assignat (1.5 vegades el diàmetre d'un individu), aquest passa a l'estat **Avançar**.

Accions:

Entrada:

- Res.

Bucle:

- Es recalculen les propietats del grup (centre, dispersió, etc.).
- S'actualitzen els individus que el formen.
- Es comproven les condicions de sortida de l'estat.

- Reagrupar (Regroup)

Descripció:

El grup es troba desunit. Els individus que el formen es dirigeixen cap al centre del grup i s'esperen que hi siguin tots per a seguir avançant.

Condicions de sortida:

- Si la proporció d'individus en estat de pànic és superior al llindar assignat (0.7) el grup canvia d'estat a **Fugir**.

- Si la distància mitjana dels individus amb el nucli no és superior al llindar assignat (3 vegades el diàmetre d'un individu) i a cap individu li queden punts intermedis entre la seva posició i el centre del grup el grup passa a **Avançar**

Accions:

Entrada:

- Es comprova si el centre del grup és un punt vàlid (no és una paret). En cas que no ho fos s'agafaria com a centre del grup la posició de qualsevol individu.

Bucle:

- Es recalculen les propietats del grup (centre, dispersió, etc.).
- S'actualitzen els individus que el formen.
- Es comproven les condicions de sortida de l'estat.

- Fugir (Flee)

Descripció:

La major part dels individus estan fugint dels enemics. En aquest estat el grup no fa res, només espera que hi hagi prou individus que ja no fugin per tal de reagrupar-se.

Condicions de sortida:

- Si la proporció d'individus en estat de pànic és inferior al llindar assignat (0.55) el grup canvia d'estat a **Reagrupar**.

Accions:

Entrada:

- Es comprova si el centre del grup és un punt vàlid (no és una paret). En cas que no ho fos s'agafaria com a centre del grup la posició de qualsevol individu.

Bucle:

- Es recalculen les propietats del grup (centre, dispersió, etc.).
- S'actualitzen els individus que el formen.
- Es comproven les condicions de sortida de l'estat.

- Intimidar (Intimidate)

Descripció:

Aquest estat és el dels enemics. Simplement es dirigeixen a un punt aleatori del escenari i quan hi arriben, se'ls assigna automàticament un de nou, de manera que estan constantment desplaçant-se per l'escenari.

Condicions de sortida:

- Aquest estat és únic i no té condicions de sortida (ni d'entrada).

Accions:

Entrada:

- Res.

Bucle:

- Es recalculen les propietats del grup (centre, dispersió, etc.).
- S'actualitzen els individus que el formen.
- Es comprova si hem arribat a l'objectiu, en tal cas se'n assigna un de nou.

Tal i com es pot deduir veient les descripcions dels 5 possibles estats d'un grup, el comportament del grup no es veu directament afectat per l'estat, ja que sempre realitza les 3 accions bàsiques a cada actualització, exceptuant l'estat d'intimidar. Aquestes accions són:

- Es recalculen les propietats del grup (centre, dispersió, etc.).
- S'actualitzen els individus que el formen.
- Es comproven les condicions de sortida de l'estat.

Però tot i l'aparent similitud de comportament en tots els estats, l'estat del grup sí que afecta, i molt, a l'estat dels individus que el formen (els quals tenen els 5 mateixos estats possibles). I, evidentment, els individus sí que tenen comportaments molt diferents depenent de en quin estat es trobin. Per tant, l'estat del grup, és el principal atribut que determina el comportament dels individus.

5.2.4 El moviment en grup

Però per a definir què és un grup cal anar més enllà dels seus atributs i dels seus estat. En el capítol de conceptes s'exposa el sistema de locomoció dels individus, el qual es basa en aplicar un conjunt de forces sobre el centre de masses per tal d'aconseguir un desplaçament de l'individu. Doncs bé, dues d'aquestes forces tenen origen en el desplaçament en grup, les quals s'encarreguen de mantenir una certa separació entre els individus, a la vegada que els individus avancen de forma conjunta i ordenada.

En el capítol de conceptes també s'explica breument el model de vehicle que es desplaça en grup proposat per *Craig Reynolds* dins del seu article *Steering Behaviors For Autonomous Characters*[1]. Segons aquest model existeixen 3 forces que condicionen el moviment el grup:

- *Cohesió*

- *Separació*

- *Alineament*

Però aquest model ha estat lleugerament modificat per tal de poder ampliar l'aplicació amb un escenari complex amb forces de repulsió, diferents estats, cerques de camins, etc. Fruit d'aquesta modificació, una d'aquestes 3 forces ha estat pràcticament eliminada i només es fa servir un cop els individus estan parats formant. Aquesta força és la de l'alineament, la qual es fa prescindible ja que tots els individus es dirigeixen al mateix punt, i per tant la força cap a l'objectiu acostuma a fer el paper de força d'alineament.

Aquesta modificació ha estat necessària degut a que treballem amb escenaris amb obstacles, i la força d'alineament es pot convertir en un problema a l'hora de passar a prop dels objectes ja que, per exemple, en passar al costat d'un sortint alguns individus del grup hauran de desviar-se lleugerament del seu camí (conseqüència de les forces de repulsió) i els de l'altre costat no. Així doncs, per evitar que, fruit d'aquesta força, alguns individus puguin veure's empesos cap a una paret, aquesta força ha estat eliminada.

D'altra banda, l'aparició de la força cap a l'objectiu per als individus, fa que tot i no existir una força d'alineament, la força cap a l'objectiu fa aquest paper i tot el grup tendeix a avançar cap al mateix punt de forma gairebé paral·lela. Sense l'ús d'aquesta força hauria estat impossible eliminar la força d'alineament proposada en el model de Reynolds. A més, la força cap a l'objectiu és constant, de la mateixa manera que ho seria la força d'alineament.

Així doncs, una altra forma d'entendre l'adaptació del model de Reynolds és entenent que la força d'alineament ha estat lleugerament adaptada i reanomenada com 'força cap a l'objectiu'.

Però la força d'alineament, a més a més d'obligar els individus a dirigir-se al mateix punt, té un segon objectiu: homogeneïtzar la velocitat dels individus. Per a simular aquest efecte que tot el grup es desplaça al mateix ritme, cada grup és conscient de quina és la velocitat màxima més petita de tots els individus que el formen. D'aquesta manera quan un individu està avançant, si aquest està en la part anterior del grup i té una velocitat superior a la menor velocitat màxima del grup, es trunca la seva acceleració de manera que tendeixi a homogeneïtzar la seva velocitat amb la de l'individu més lent del grup.

Aquest truncament de la velocitat dels individus més ràpids fa que el grup viatgi aproximadament a la velocitat de l'individu més lent. Ja que si es vol un grup unit no pot haver individus que es desplacin molt més ràpids que les màximes possibilitats dels lents.

Finalment, una altra modificació important ha estat la de calcular un centre de masses ponderat. La ponderació s'ha fet de manera que els individus més lents tinguin més pes a l'hora de fer el sumatori de posicions dels individus per trobar la posició mitjana. Aquesta ponderació obliga al centre del grup a desplaçar-se lleugerament cap als individus més lents, de manera que són els ràpids els que més alteren el seu comportament per tal d'apropar-se als lents. Aquest centre ponderat ha estat imprescindible per tal que els grups no homogenis (on cada individu te unes propietats físiques d'acceleració i velocitat màxima) no patissin una excessiva desunió entre ells.

5.3 - L'individu

Si el grup es considera la base sobre la qual s'assenta tot el projecte, l'individu ha de ser considerat el fonament que justifica el grup. Un grup és poc més que un conjunt d'individus i unes propietats comunes, i és per això que els estats, el comportament i les propietats dels individus són més complexos que els del grup que els contenen.

5.3.1 - Les característiques de l'individu

Fina ara, quan en aquest treball s'ha parlat d'un individu, es feia referència a qualsevol tipus de vehicle o personatge animat que es pugui desplaçar en grup per sobre d'una superfície plana. Però si es vol implementar una solució concreta cal definir clarament què entenem per un individu.

I per tal de definir com és un individu, cal explicar com es comporta. El comportament d'un individu és força senzill. El seu objectiu és tan simple com desplaçar-se d'un punt qualsevol de l'escenari a un altre, evitant els obstacles que hi pugui haver sobre l'escenari mentre es desplaça de forma cohesionada amb la resta d'individus del seu propi grup.

Però a part de les dificultats que pugui oposar l'escenari, també hi ha diversos enemics desplaçant-se alhora que poden influir en el comportament d'un individu. Aquests enemics fan que amb la seva presència un individu vagi augmentant el seu nivell de por, fins que en un moment determinat (quan l'individu té massa por) aquest fuig desentenent-se dels seus companys.

Aquestes fugides provoquen que el grup es desuneixi i els individus s'escampin per l'escenari, el que obliga els individus a que siguin capaços de reagrupar-se abans de seguir avançant.

Els 4 comportaments possibles d'un individu controlat per l'usuari són els següents:

- *Avançar*

- *Fugir*

- *Reagrupar-se*

- *Formar*

En aquest llistat es pot veure que a part dels tres comportaments comentats (*Avançar*, *Fugir* i *Reagrupar-se*), existeix un quart tipus de comportament que pot assolir un individu, anomenat *Formar*. Aquest comportament es produeix quan un individu arriba al seu destí (juntament amb la resta del grup), és en aquest moment quan tots els individus del grup s'alineen en la mateixa direcció de manera que s'esperin en una formació compacta.

Així doncs, hem dit que un individu pot assolir 4 comportaments diferents depenent de les condicions de la simulació en un moment determinat, tot això sense comptar el cinquè estat possible, només reservat per als individus enemics, anomenat *Intimidar*, que es tracta simplement d'una petita modificació del comportament d'avançar.

Però per a definir un individu no es suficient saber 'què' fa, també s'ha de conèixer 'com' ho fa. És per tant necessari descriure quin tipus de moviment efectua un individu quan es desplaça, independentment dels seus companys de grup.

El moviment d'un sol individu pot resultar molt diferent depenent del model que es vulgui utilitzar. Resulta evident que una persona caminant pel carrer no es mou igual que un cotxe. No només per la velocitat o l'acceleració, sinó pel tipus de moviment que pot realitzar. Un cotxe, per exemple, no pot fer desplaçaments laterals ni pot girar sobre si mateix sense avançar. Però aquestes restriccions no són comunes a tots els tipus de vehicles motoritzats, ja que un carro de combat, a diferència de la majoria de vehicles amb rodes sí que pot girar sobre si mateix sense avançar ja que el seu sistema de tracció i de gir són independents. Així doncs, és important delimitar quins tipus de moviments contemplarà la nostra aplicació abans d'endinsar-nos en complexos sistemes de moviments coordinats.

El model de moviment de cada individu escollit per a aquest projecte ha estat el que es correspondria al que efectua un carro de combat. En altres paraules, l'individu només pot avançar frontalment però pot girar sobre si mateix estant parat.

Aquesta elecció ha estat fruit de dos factors, primerament s'ha considerat que si l'individu pot desplaçar-se lateralment el moviment pot quedar molt deslluït i artificial. Per altra banda, restringir el gir al fet que només es pugui produir en el cas que l'individu estigui avançant augmentaria enormement la complexitat de les situacions i es crearia una casuística extremadament complexa (degut a que treballarem amb escenaris amb obstacles, passadissos, etc.) que queda fora de l'abast d'aquest projecte.

En la següent taula es mostra el tipus de moviment escollit per al projecte, en una possible classificació simplificada de tipus de moviments:

	Gira sobre si mateix	Necessita avançar per a girar
Pot fer desplaçament lateral	-	-
Només es desplaça frontalment	Sí	-

Figura 20. Tipus de moviment del sistema de locomoció emprat.

També es poden considerar molts altres factors que defineixen el tipus de trajectòria que segueix un objecte en desplaçar-se. Per exemple, un cotxe pot patir petits desplaçaments laterals si gira pronunciadament (una derrapada) o, seguint amb el mateix exemple, un vehicle pot perdre el control i fer moviments difícilment simulables en cas que aquesta derrapada no sigui controlada. Però tots aquests casos requereixen de models físics força complexos que es consideren fora de l'abast d'aquest projecte, ja que per si sols podrien constituir tot un projecte final de carrera.

Per tots els arguments exposats es pot considerar que el comportament dinàmic dels individus aïllats no és real, però tot i això, el resultat que proporciona aquesta elecció simplificada es pot considerar satisfactori, ja que compleix la finalitat d'aportar una base dinàmica individual suficientment realista i manejable per a construir-hi a sobre el model dinàmic dels grups.

Per a simular aquest tipus de moviment on el gir és independent de la tracció s'ha escollit el sistema de forces exposat en el capítol de Conceptes, el qual consisteix en aplicar diferents forces a un punt (centre de masses d'un individu) i extreure'n dues components, un vector paral·lel a la direcció on mira l'individu i un altre vector perpendicular a l'anterior. El primer vector es correspondrà a la quantitat d'acceleració o desacceleració que pateix l'individu, i el segon a la brusquedat del gir que sofrirà.

Sobre aquest sistema de locomoció s'ha afegit una petita variació per tal de simular la desacceleració d'un individu just abans d'arribar a l'objectiu.

$$distancia = velocitat^2 / (2 \times maxBrakeForce)$$

Figura 21. Fórmula utilitzada per calcular la distància de frenada.

Per a simular aquesta frenada per aturar-se, just abans de renovar-se la posició a partir del temps transcorregut i la força aplicada sobre el centre de masses, es calcula l'espai que requeriria l'individu per a frenar, sabent la seva capacitat de frenada i la velocitat actual (utilitzant la fórmula de la figura 21) i si aquest espai és superior a la distància fins l'objectiu, l'individu frena. El resultat es pot apreciar en la figura 22, on un individu s'atura progressivament sobre l'objectiu final.

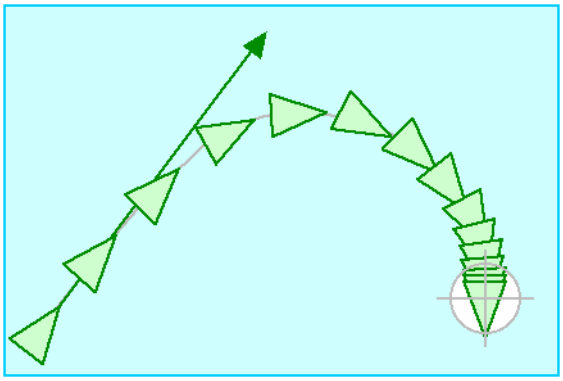


Figura 22. Un individu aturant-se sobre l'objectiu.

5.3.2 - Els atributs de l'individu

De tots els atributs que conté un individu, cap pot ser modificat directament per l'usuari de l'aplicació. Tot i que com a membres d'un grup, tenen accés a la informació compartida amb els altres membres.

Els atributs d'un individu es poden dividir en dos grups clarament diferenciats. Per una banda tenim els atributs que defineixen l'estat temporal d'un individu (posició, velocitat, etc.), mentre que per l'altra banda tenim un conjunt d'atributs que es llegeixen des d'un fitxer d'escenaris (que conté tota la informació per realitzar una simulació) i que no es modifiquen durant tota l'execució.

Aquest segon grup d'atributs simbolitzen les propietats d'un individu. Concretament, defineixen com de resistent a la por pot ser un individu i quines propietats dinàmiques té (acceleració, frenada, velocitat màxima, i velocitat màxima de rotació).

El llistat complet d'atributs és el següent:

- *state* (Enter): Variable on s'emmagatzema l'estat de l'individu. Pot tenir 5 valors diferents. 4 en el cas de tractar-se d'individus controlats per l'usuari (avançar, formar, reagrupar o fugir) o 1 de sol si es tracta d'un individu membre d'un grup d'enemics.

- *fear* (Coma flotant): Quantitat de por acumulada per l'individu, causada per la presència propera d'enemics.

- *position* (Vector): Es tracta d'un vector tridimensional amb la posició absoluta del centre de masses de l'individu.

- *speed* (Coma flotant): És la velocitat instantània de l'individu.

- *forward* (Vector): En aquest vector s'hi guarda la direcció cap on mira l'individu.

- *checkPoint* (Enter): Aquesta variable conté l'index per saber a quin punt intermedi de la llista de *checkPoints* que té el grup s'ha de dirigir l'individu. És un marcador que indica en quin tram del camí fins l'objectiu es troba l'individu, ja que tots els membres del grup comparteixen el mateix conjunt de *checkPoints*).

- *right* (Vector): Es tracta d'un vector perpendicular al *forward* que apunta cap a la dreta de l'individu. De la mateixa manera que el vector *up* s'utilitza per tal de fer les rotacions de l'individu sobre el pla que defineix l'escenari

- *up* (Vector): Normal de l'individu, perpendicular a *forward* i a *right*.

- *toTargetForce* (Vector): Aquesta variable s'utilitza per guardar la força d'atracció cap a l'objectiu que pateix un individu. Juntament amb *toGroupCenterForce* i *collisionForce* s'utilitza per poder mostrar aquests vectors a l'hora de renderitzar.

- *toGroupCenterForce* (Vector): Força de cohesió.

- *collisionForce* (Vector): Força de repulsió amb les parets i els altres individus.

- *regroupCPs* (Llista de Vectors): En aquesta llista s'emmagatzemen els punts intermedis retornats per l'algoritme de cerca de camins, per tal de poder arribar a l'objectiu (evitant i rodejant els obstacles) en el cas que individu ho requereixi per tal de reagrupar-se.

- *regruoupCP* (Enter): És la quantitat de punts intermedis que li falten a l'individu per tal d'arribar al centre del grup.

- *regroupFinal* (booleà): Cert en cas que el camí descrit per la llista *regroupCPs* arribi a l'objectiu (centre del grup) o fals en el cas contrari. Si és fals voldrà dir que un cop arribat al final del camí s'haurà de recalcular el camí fins el centre del grup des de la nova posició.

- *maxSpeed* (Coma flotant): Propietat que limita la velocitat màxima d'un individu.

- *maxAccForce* (Coma flotant): Magnitud màxima de la força d'acceleració (en el sentit de la marxa).

- *maxBrakeForce* (Coma flotant): Magnitud màxima de la força de frenada (en el sentit oposat a la marxa).

- *maxTurnForce* (Coma flotant): Magnitud màxima de la força de gir (perpendicular a la marxa).

- *maxfear* (Coma flotant): Per sobre d'aquest nivell l'individu entra en estat de pànic. Simula la resistència a la por generada per la presència d'enemics.

5.3.3 - Els estats de l'individu

Tal i com succeeix amb els grups, l'atribut que més condiciona el comportament d'un individu és l'estat (*state*). Aqueta variable emmagatzema en quin dels 5 estats possibles es troba

l'individu en un moment determinat. Tots 5 estats són homònims als 5 estats possibles d'un grup, i d'igual forma es poden agrupar en 2 categories. 4 defineixen els 4 possibles comportaments d'un individu controlat per l'usuari (*avançar*, *formar*, *reagrupar-se* i *escapar*), mentre que el cinquè està reservat únicament per als enemics (*intimidar*).

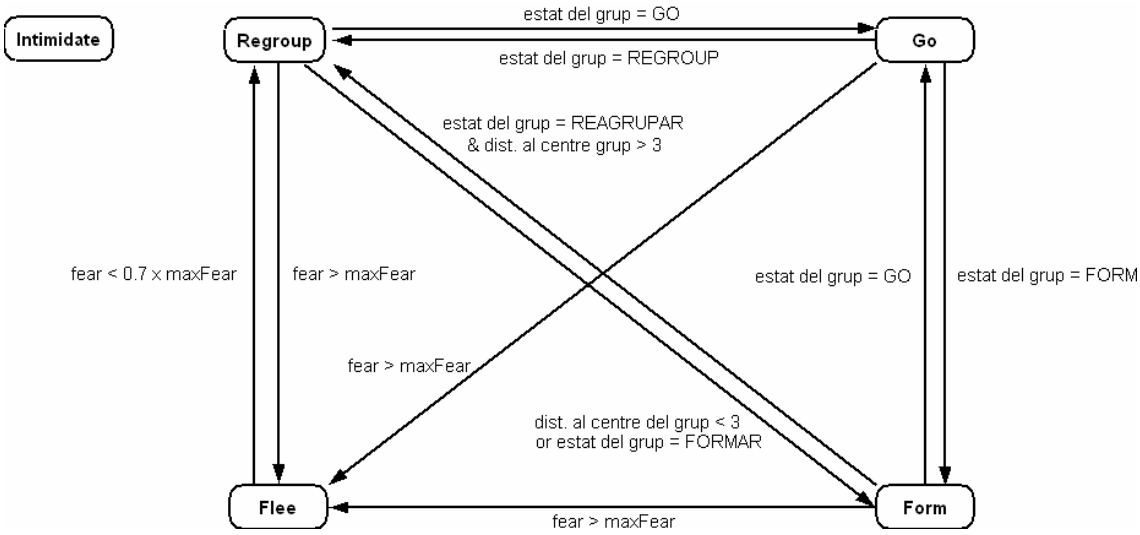


Figura 23. Estats d'un individu.

Com es pot veure a la figura 23, les condicions de sortida dels estats estan molt influenciades per l'estat del grup, ja que els individus, tot i ser independents i no jerarquitzats dins del grup, estan coordinats per tal de simular un comportament força homogeni.

- Avançar (Go)

Descripció:

El grup està unit i els individus es desplacen de forma cohesionada cap a l'objectiu assignat, en cas d'estar a prop de l'objectiu final aniran minorant la marxa progressivament.

Condicions de sortida:

- Si el nivell de pànic (*fear*) supera el màxim suportat per l'individu (*maxFear*) es canvia l'estat a **Fugir**.

- Si el grup ha canviat d'estat a **Reagrupar**, l'individu passa a l'estat **Reagrupar**.

- Si el grup ha canviat d'estat a **Formar**, l' individu passa a l'estat **Formar**.

Accions:

Entrada:

- Res.

Bucle:

- Es recalcula el nivell de por.
- Es comproven les condicions de sortida de l'estat.
- Es sumen les següents forces:
  - Força de cohesió (cap al centre del grup).
  - Força de repulsió (oposada als obstacles veïns).
  - Força constant cap a l'objectiu.
- Si el grup està més endarrerit que l'individu, i no és l'individu més lent, limita la seva velocitat.
- Aplica el sumatori de forces per a realitzar el moviment.
- Si l'objectiu és suficientment a prop, minora la velocitat progressivament.
- En el cas de ser proper al *checkpoint* al que es dirigeix, mou l'índex de punts intermedis i es dirigeix cap al següent.

- Formar (Form)

Descripció:

L'individu ja ha arribat al seu objectiu i s'espera aturat i alineat segons la direcció d'alineació del grup.

Condicions de sortida:

- Si el nivell de pànic (*fear*) supera el màxim suportat per l'individu (*maxFear*) es canvia l'estat a **Fugir**.
- Si el grup està en l'estat **Reagrupar** i la distància amb l'objectiu és major de 3 vegades el diàmetre d'un individu, l'individu passa a l'estat **Reagrupar**.
- Si el grup està en l'estat **Avançar**, l'individu passa a l'estat **Avançar**.

Accions:

Entrada:

- Res.

Bucle:

- Es recalcula el nivell de por.
- Es comproven les condicions de sortida de l'estat.

- Si l'individu és proper al centre del grup i no està gaire apretat, s'atura i s'alinea.

- Si l'individu és proper al centre del grup, però està molt apretat, es dirigeix cap a fora (força de repulsió + força cap a l'objectiu)

- Si l'individu és lluny del centre del grup se li aplica únicament la força de cohesió per a realitzar el moviment. Disminuint la marxa a mesura que s'apropa a l'objectiu.

- Reagrupar (Regroup)

Descripció:

El grup esta desunit i per tant l'individu es dirigeix cap al centre del grup de forma directa (sense usar camins) o seguint una llista de punts intermedis pròpia.

Condicions de sortida:

- Si el nivell de pànic (*fear*) supera el màxim suportat per l'individu (*maxFear*) es canvia l'estat a **Fugir**.

- Si el grup es troba en l'estat **Formar**, l'individu passa a l'estat **Formar**.

- Si la distància entre el centre de l'individu i el centre del grup és inferior a 3 vegades el diàmetre d'un individu, canvia a l'estat **Formar**.

- Si el grup es troba en l'estat **Avançar**, l' individu passa a l'estat **Avançar**.

Accions:

Entrada:

- S'observa si hi ha obstacles entre l'individu i el centre del grup, en cas de ser així es mira si el centre del grup està en una cel·la buida i si no ho està es substitueix el centre del grup per la posició actual de l'individu.

- Si hi ha obstacles, es calcula un camí usant l'A* implementat. En cas que el grup no es trobi en l'estat **Reagrupar**, es posa el booleà *regroupFinal* a cert per tal d'indicar que un cop s'arribi al final del camí aquest s'haurà de recalcular, ja que el centre del grup segurament es desplaçarà.

Bucle:

- Es recalcula el nivell de por.

- Es comproven les condicions de sortida de l'estat.

- Es sumen les següents forces:

  - Força de cohesió (cap al centre del grup).

  - Força de repulsió (oposada als obstacles veïns).

- Aplica el sumatori de forces per a realitzar el moviment.
- Si l'objectiu est suficientment a prop, disminueix la velocitat progressivament.
- En el cas d'estar proper al *checkpoint* al què es dirigeix, mou l'índex de punts intermedis i es dirigeix cap al següent.

- Fugir (Flee)

Descripció:

L'individu ha entrat en estat de pànic i es s'allunya dels enemics que l'envolten, desentenent-se completament de la resta d'individus del grup. Un cop passat el pànic, l'individu s'intentarà reagrupar amb la resta del grup.

Per fomentar les fugides de tot un grup, quan un individu entra en aquest estat propaga cert nivell de por en els seus companys més propers.

Condicions de sortida:

- Si el nivell de pànic (*fear*) baixa per sota del 70% del nivell de por màxim suportat per l'individu (*maxFear*) es canvia l'estat a **Reagrupar**.

Accions:

Entrada:

- Augmenta la por de la resta d'individus del grup.

Bucle:

- Es recalcula el nivell de por.
- Es comproven les condicions de sortida de l'estat.
- Es mira quins enemics hi ha i es suma una força de repulsió per a cada enemic en sentit contrari a la posició de l'enemic.
- Suma la força de repulsió dels enemics i la de repulsió amb l'escenari.
- Aplica la força resultant al centre de masses per tal de realitzar el moviment.
- Si no hi ha cap enemic proper frena progressivament.

- Intimidar (Intimidate)

Descripció:

Si un individu es troba en aquest estat vol dir que es tracta d'un enemic. El comportament d'aquest enemic serà el de dirigir-se constantment al punt de destí (evitant obstacles) marcat pel grup, el qual canvia un cop el grup hi arriba, de forma que els individus estan sempre avançant.

Condicions de sortida:

- Cap.

Accions:

Entrada:

- Res.

Bucle:

- Es sumen les següents forces:
  - Força de cohesió (cap al centre del grup).
  - Força de repulsió (oposada als obstacles veïns).
  - Força constant cap a l'objectiu.
- Si el grup està més endarrerit que l'individu, alinea la força de cohesió amb el vector de direcció del moviment de l'individu.
- Aplica el sumatori de forces per a realitzar el moviment.
- En el cas d'estar proper al *checkpoint* al què es dirigeix, mou l'índex de punts intermedis i es dirigeix cap al següent.

5.3.4 - Les 5 forces que es poden aplicar a un individu

Com ja hem vist, segons l'estat en què es trobi un individu, s'aplicaran diferents forces al centre de masses per tal de simular el desplaçament. De forces possibles a calcular n'hi ha 5:

- Força de repulsió de les parets.
- Força de repulsió d'altres individus.
- Força cap a l'objectiu.
- Força de cohesió (cap al centre del grup).
- Força de repulsió dels enemics.

Aquestes 5 forces es calculen de manera diferent, especialment la magnitud de la força ja que poden ser constants, lineals o exponencials. Però abans de descriure cadascuna d'aquestes

forces. Cal destacar que, depenent de la situació, aquestes forces es ponderen de forma diferent, de manera que depenent de l'estat en què es trobi l'individu, cada una de les magnituds d'aquests vectors es veurà multiplicada per un escalar. A la figura 24, es pot veure una taula amb els pesos de cada força depenent de l'estat de l'individu.

	Go	*Form*	*Regroup*	*Flee*	*Intimidate*
Repulsió parets	1	1	1	1	1
Repulsió individus	1	1	1	1	1
Cap a l'objectiu	0.5	0.2	-	-	0.5
Cohesió	0.1	0.5	2	-	0.1
Repulsió enemics	-	-	-	0.5	-

Figura 24. Taula de ponderació de les forces.

Seguidament es detalla com es calcula cada una d'aquestes 5 forces. La combinació de les quals determina el desplaçament d'un individu.

Força de repulsió de les parets

Aquesta força es calcula mirant la distància entre el centre de masses de l'individu i la paret més propera. Per calcular aquest vector de força es sumen les forces de repulsió amb les 8 cel·les veïnes, que són perpendiculars a la paret tal i com s'explica en l'apartat "L'escenari".

Però a part de la direcció de la força, també és important la magnitud de la força. Aquesta magnitud és exponencial i es calcula segons la fórmula de la figura 25, on *distancia* és la distància entre el centre de masses de l'individu i el punt més proper de la paret.

$$magnitud = (0.6 / distancia^2) - 0.375$$

Figura 25. Fórmula de la magnitud de la força de repulsió amb les parets.

Aquesta magnitud només es calcula en cas que la distància es trobi dins del rang [1, 4], tenint en consideració que el radi d'un individu es d'1. Per tant, un individu que es trobi tocant una paret, estarà a distància 1. El límit superior és de 4 ja que és l'amplada d'una cel·la, i per tant, calcular a major distància requeriria mirar no només les 8 cel·les veïnes, sinó que també s'haurien de mirar cel·les més enllà, però s'ha considerat prescindible.

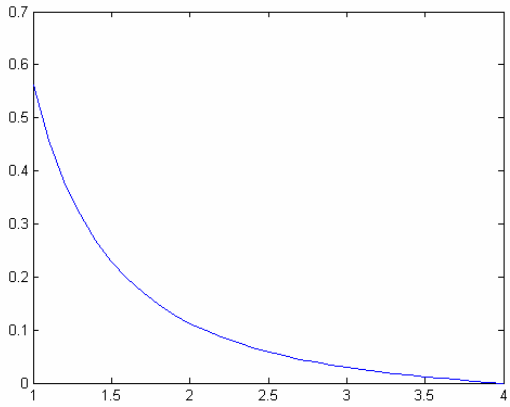


Figura 26. Magnitud de la força de repulsió amb les parets depenent de la distància.

En el cas de produir-se un xoc amb una paret, en l'instant immediatament posterior pot succeir que la distància entre el centre de masses i la paret sigui inferior a 1, el que indicaria un solapament. En tal cas, quan es calcula la força de repulsió amb les parets, la posició de l'individu es reajusta, desplaçant-la perpendicular a la paret fins a una distància equivalent al diàmetre del propi individu, de manera que un lateral de l'individu toqui la paret.

Força de repulsió d'altres individus

De la mateixa manera que les parets, la resta d'individus representen un obstacle per al propi individu. És per això que cal calcular una força de repulsió entre els individus.

A diferència de la força de repulsió generada per les parets, la força que generen els individus per tal de separar-se entre ells és senzilla de calcular. Per calcular aquesta força només cal recórrer tots els individus que es trobin propers a l'individu que estem processant i anar acumulant la força de repulsió generada per cada un d'ells.

$$magnitud = 2 / distancia^2$$

Figura 27. Fórmula de la magnitud de la força de repulsió amb altres individus.

La magnitud d'aquesta força és exponencial i només es calcula (segons la fórmula de la figura 27) dins del rang [2 10], ja que només s'observaran els individus que estiguin a una distància menor o igual a 10 vegades el radi d'un individu. El límit inferior de 2 està justificat perquè 2 és dues vegades el radi d'un individu, de tal manera que dos individus que s'estiguin tocant estan realment a distància de 2, ja que la distància fa referència a l'espai que separa els centres de masses i no els laterals d'un individu.

Més concretament, la magnitud de la força de repulsió entre individus és la següent:

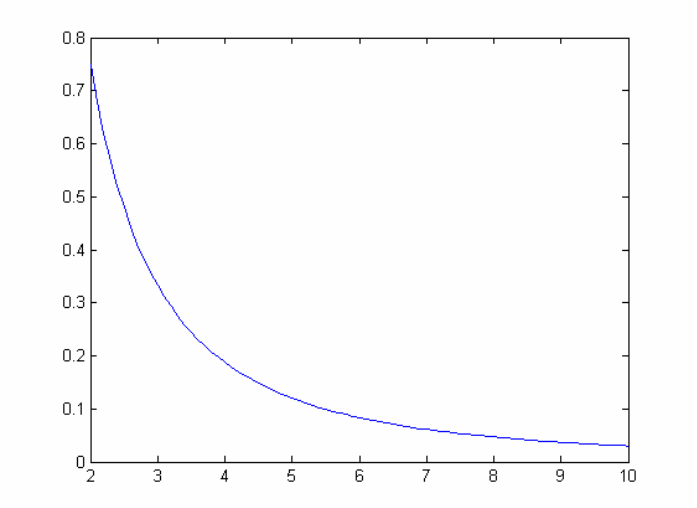


Figura 28. Magnitud de la força de repulsió entre individus depenent de la distància.

Justament quan es calcula aquesta força és quan es comprova que no hi hagi dos individus solapats (amb una distància entre centres de masses inferior a 2). Si es donés aquest cas, tots dos individus es veurien separats la mateixa distància per en sentits oposats, el qual simula un xoc de manera que els individus es poden empènyer entre ells.

Força cap a l'objectiu

La força cap a l'objectiu és la més simple de totes. La seva magnitud és constant (sempre val 1) i la direcció és el vector resultant de restar-li a la posició de l'objectiu la posició del centre de masses de l'individu.

És lògic usar una força constant per tal de simular l'atracció de l'objectiu sobre un individu, ja que aquesta atracció sempre serà igual de forta, independentment de la distància a la que es trobi l'objectiu.

Força de cohesió

La força de cohesió no és més que una força de magnitud lineal que simula l'interès d'un individu per ser el més a prop possible del centre del grup, de fet aquesta és la força responsable que el grup viatgi unit.

La direcció d'aquesta força és el vector resultant de restar-li la posició del individu a la posició ponderada del centre del grup. Quan es parla de centre ponderat d'un grup es fa referència a un virtual centre de masses del grup. Aquesta ponderació és determinada per la velocitat màxima dels individus, de manera que quan més alta sigui aquesta velocitat màxima menys pes tindrà l'individu en la mitjana ponderada dels centres de masses.

Aquest ús d'un centre de masses que tendeix a ser més proper als individus més lents genera una simulació on els individus més ràpids tendeixen a apropar-se als més lents, de manera que els individus més lents hagin de desviar-se menys del camí més curt, i que siguin els ràpids els que condicionin més el seu desplaçament.

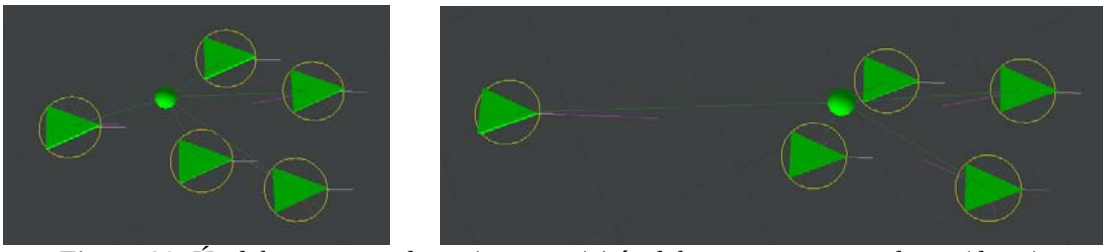


Figura 29. Ús del centre ponderat (esquerra) i ús del centre sense ponderar (dreta).

.

A la figura 29, es poden veure dos grups d'idèntiques característiques, el primer utilitzant un centre de masses ponderat i el segon no. Es pot apreciar clarament que en el primer cas el grup viatja molt més unit ja que el centre es troba més endarrerit (proper a l'individu més lent que es desplaça últim) i els 4 individus més ràpids, tendeixen a esperar al lent. En la simulació fent servir el centre de masses sense ponderar, els grups no homogenis tendeixen a deixar forats entre els individus ràpids i els lents, i tot i que acabin viatjant a la mateixa velocitat mitjana, ho fan amb un major grau de dispersió. Es per això que en l'aplicació s'usa el centre ponderat per velocitats màximes per tal de simular el comportament dinàmic dels grups.

Un cop calculat aquest vector entre el centre ponderat del grup i el centre de masses de l'individu, ja no cal recalcular la magnitud, ja que aquesta magnitud ja és directament proporcional a la distància.

Si només s'utilitzés aquesta força el grup es col·lapsaria en un sol punt, el factor que evita aquesta unió és la força de repulsió entre individus recentment comentada, la qual fa que els individus es desplacin amb una certa distància entre ells, tal i com es pot veure a la figura 30.

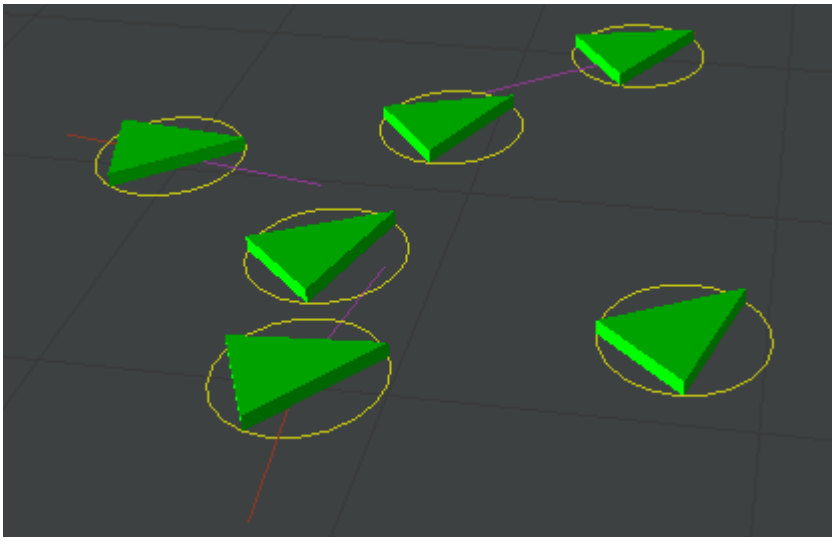


Figura 30. Un grup amb les forces de cohesió i separació.

Força de repulsió dels enemics

Si un individu està fugint, el seu únic interès és anar tany lluny com sigui possible dels enemics que l'envolten. Per simular aquesta repulsió es calculen acumulativament les forces de repulsió de tots els enemic que es trobin a una distància no superior a 15 vegades el diàmetre

d’un individu (que és la distància màxima entre un individu i un enemic en la que l’enemic provoca un augment del nivell de por de l’individu).

La direcció de la força de repulsió d’un enemic és en la direcció oposada a la que es troba l’enemic (vector resultant de restar a la posició de l’individu la posició de l’enemic). Un cop normalitzat aquest vector es multiplica per la magnitud desitjada, la qual es calcula segons la fórmula de la figura 31.

$$magnitud = 1 / distancia^2$$

Figura 31. Fórmula de la magnitud de la força de repulsió dels enemics.

Com es pot veure, aquesta magnitud decreix quadràticament quan la distància augmenta, de manera que l’individu tendirà a allunyar-se més dels enemics més propers i ignorarà els més llunyans.

6. Implementació

Tots els conceptes teòrics exposats fins ara han estat estudiats i revisats per tal de poder crear una aplicació que realitzes aquesta simulació de comportament dinàmic. Aquesta aplicació s'ha creat a partir de les MFC (*Microsoft Foundation Classes*), les quals han aportat un marc predefinit per a encabir-hi les classes encarregades de realitzar la simulació.

El camí fins a obtenir l'aplicació acabada ha estat llarg i complex, ja que la gran quantitat de línies de codi necessàries ha augmentat molt més del planificat en una primera fase d'anàlisi. Tot i patir aquest i altres tipus d'entrebancs, la utilització d'una metodologia d'enginyeria del software ha permès arribar al final i veure'n un resultat satisfactori, el qual reflexa les virtuts i els defectes de la solució proposada, fent que la implementació es limiti a una mera qüestió estètica i pragmàtica que facilita tant la interacció com l'anàlisi de la simulació mitjançant la utilització de codis de colors i de figures mostrats per pantalla.

Així doncs, tot i haver dotat al motor 3D d'una certa vistositat estètica. Totes les formes i colors han estat pensats per a ser fàcilment identificables i facilitar així la comprensió de què està succeint en cada moment. Permetent solapar la fase d'implementació i la d'anàlisis a través d'un model de desenvolupament evolutiu com el que es detalla a continuació.

6.1 - Metodologia utilitzada

El procés fins arribar a obtenir una aplicació que permetés simular el comportament dinàmic dels grups segons el model proposat ha estat un procés complex on la part d'implementació ha aportat un *feedback* al procés de disseny, el qual ha anat variant lleugerament des de l'inici, a mesura que els resultats empírics amb el codi aportaven nous punts de vista complementaris, que han permès una millor comprensió del problema.

Aquesta interacció ha estat fruit d'un model de desenvolupament evolutiu, dividit en varies seqüències d'anàlisis, disseny, testeig i valoració dels resultats. El model evolutiu es fa imprescindible en projectes com aquest, ja que la utilització d'un model de cicle de vida en cascada, basat en un desenvolupament seqüencial d' Anàlisis–Disseny–Implementació, seria massa estricte davant d'una possible variació en l'etapa de disseny, i comportaria que una gran part de la feina realitzada fins al moment resultes inútil.

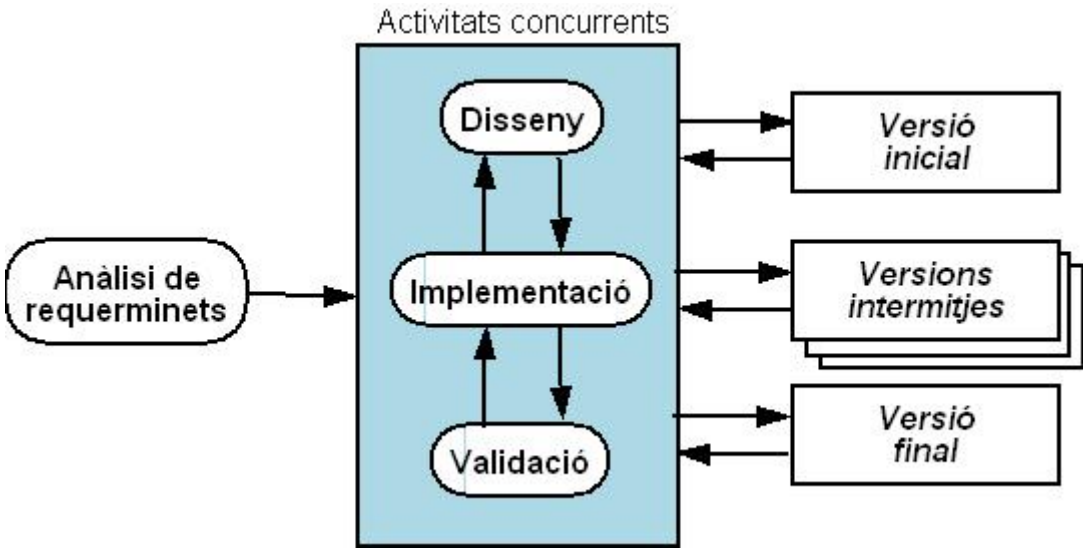


Figura 32. Model de desenvolupament evolutiu aplicat al sistema.

Així doncs, la utilització d'un model de desenvolupament evolutiu com el que es pot veure a la figura 32, està plenament justificat, sobretot si tenim en compte que el sistema es caracteritza per ser de vida curta i que requereix d'un *feedback* constant. Per altra banda, un model de desenvolupament formal que garantís la correctesa de l'aplicació s'ha descartat des de l'inici ja que aquesta metodologia només resulta aconsellable per a sistemes crítics, especialment aquells amb compromisos de seguretat.

Dins d'aquest model de treball, en la primera fase es va crear el marc de visualització, integrat en les llibreries *MFC* (*Microsoft Foundation Classes*) el qual es va convertir en una primera aplicació amb menús configurables i amb capacitat per renderitzar una superfície plana sense obstacles on provar les primeres simulacions de grups d'individus.

Un cop creada aquesta eina es van començar a afegir funcionalitats progressivament, totes elles ben testejades i encapsulades per tal de reescriure la menor quantitat de codi possible en el futur. Aquestes funcionalitats s'han afegit per ordre d'importància, de manera que les més importants s'implementessin primer ja que serien les més comprovades i testejades durant les futures iteracions que encara quedaven per fer.

Una altra virtut d'aquest model incremental de programació és que encara que, en una hipotètica situació, no s'acabés el projecte amb totes les seves funcionalitats, sempre es tindria una versió del sistema executable amb la resta de funcionalitats disponibles.

6.2 - Etapes del projecte

Seguint la metodologia descrita, el projecte ha tingut varies entregues de l'aplicació que s'han testejat de forma independent.

La primera etapa, però, no ha comportat cap implementació, ja que abans de començar a programar res ha calgut fer un anàlisis profund de l'estat de l'art per tal de fer una bona captura de requeriments.

Un cop definits els requeriments, tant funcionals com no funcionals, el següent pas ha estat planificar els diferents estat dels grups i els individus, amb les condicions de sortida i la descripció aproximada de que fa cada individu en aquest estat. Durant aquesta fase d'anàlisis i disseny s'han fet varies proves per tal d'escollir les llibreries utilitzades i descartar-ne les que no eren vàlides, com és el cas d'*Irrlicht* o *OpenSteer*.

Amb aquesta base, ja ha estat possible implementar les diferents entregues fins arribar a la solució final. A continuació es descriuen aquestes entregues amb les funcionalitats afegides a cada iteració.

a) Marc Inicial

En una primera implementació es van crear les funcionalitats principals de visualització (el renderitzat *OpenGL* i una primera versió dels menús amb les *MFC*). Dins d'aquest marc també es van crear les principals classes per tal de definir l'esquelet de l'aplicació i així, en les següents entregues, poder-se limitar a afegir funcionalitats i mètodes sense haver de reestructurar res.

b) Primer prototip

En aquesta implementació es van crear les funcionalitats principals del moviment en grup. Es va implementar el sistema de locomoció i es van poder començar a fer les primeres proves amb les forces de cohesió, separació i cap a l'objectiu. En aquesta versió encara no hi apareixien enemics ni parets.

c) Afegint enemics i fugides

Per tal de poder provar els diferents estats possibles dels grups, calia afegir elements que provoquessin un augment de la dispersió en els grups i per tant s'activessin tant l'estat de fugir com el de reagrupar. En aquesta iteració es van afegir, doncs, els enemics.

d) L'escenari i els fitxers de configuració

Un cop teníem els grups, capaços de fugir dels enemics, el següent pas era començar a experimentar amb escenaris amb obstacles. Però va ser també en aquest punt quan es va decidir implementar la lectura de fitxers d'escenaris, per tal de poder treballar amb diferents tipus d'escenaris sense haver de modificar el codi i recompilar l'aplicació.

Aquesta fase va ser una de les més costoses (sobretot en temps) ja que introduir l'escenari requeria també començar a treballar amb les forces de repulsió de les parets i dotar als individus d'algun mecanisme capaç d'evitar obstacles. Inicialment es va provar d'utilitzar sistemes alternatius als algoritmes de cerca de camins basats en nodes, però vistos els resultats es va prendre la decisió d'integrar un algoritme com l'A*, de manera que els individus i els grups també es van veure reformats per tal de poder treballar amb els punts intermedis (*checkpoints*).

e) Afegint 3D

Les versions anteriors es representaven de forma bidimensional, de manera que els individus eren triangles sobre una superfície i les parets es pintaven en forma de polígons plans sobre el terra de l'escenari. En aquesta iteració es va afegir la tercera dimensió en la representació i un conjunt de llums i detalls gràfics per tal de fer més agradable la visualització.

f) Entrega final

En la fase final del projecte es van acabar de definir les fórmules de les forces i es van retocar els menús. Aquesta entrega es caracteritza per corregir els defectes de les versions anteriors gràcies a un extens joc de proves per tal d'acabar de perfilar el resultat final.

6.3 - Estructura i funcionament de l'aplicació

L'aplicació resultant d'aquest projecte, ha estat un extens conjunt de classes integrades en un marc d'aplicació MFC (*Microsoft Foundation Classes*). Entre aquestes classes hi trobem un primer grup on s'ha implementat el comportament dels grups i individus, juntament amb l'escenari. Mentre que un segon grup format per les classes *Paint* i *Camera* estan destinades a fer el renderitzat.

Evidentment, també ha fet falta implementar varies classes i estructures menors com la classe *3DVector* (que representa un vector tridimensional) o l'estructura *TCell* (que agrupa els atributs d'una cel·la) que ofereixen funcionalitats genèriques que fan servir les principals classes. Però degut a la seva obvietat no es comenten en aquesta memòria ja que no aporten ni reflecteixen cap concepte teòric rellevant.

Així doncs, les principals classes es relacionen de la següent manera, tal i com s'aprecia en el diagrama de classes de la figura 33.

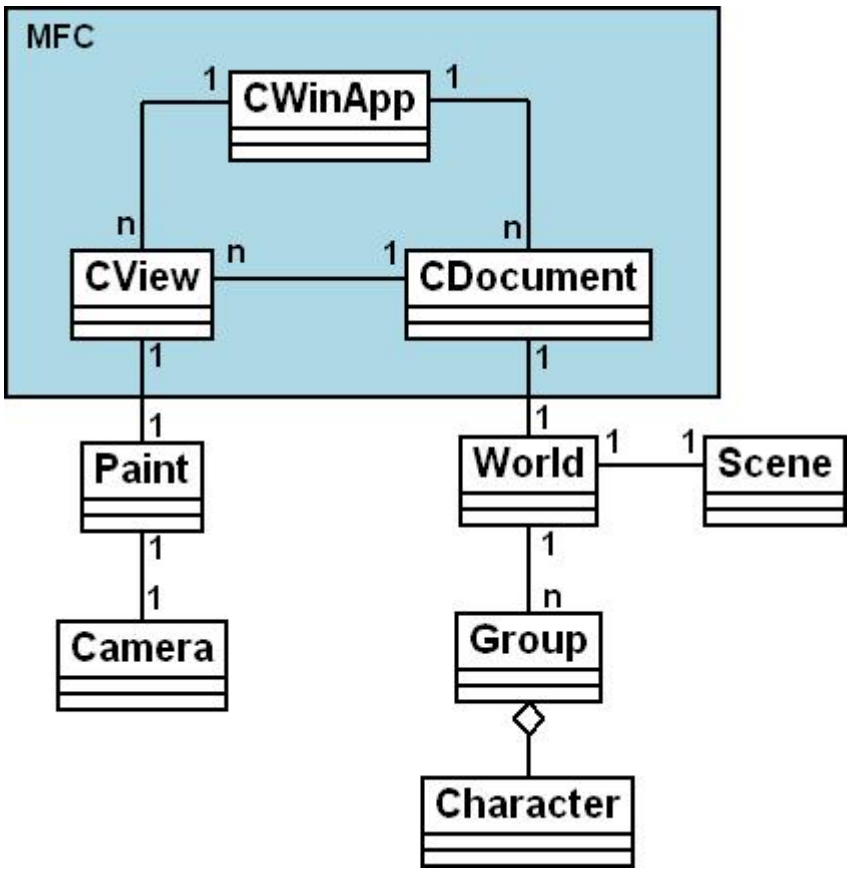


Figura 33. Diagrama de classes de l'aplicació.

En aquest diagrama es pot veure com l'aplicació implementada parteix de l'estructura de múltiples vistes proporcionada per la llibreria MFC (sobre fons blau) i es divideix en dos blocs clarament diferenciats.

Per una banda tenim que una instància de la classe *Paint* és un atribut en la classe *CView*, de manera que la part de visualització queda encapsulada dins de *CView*.

A la part dreta del diagrama es pot apreciar que la classe *CDocument* conté una instància de *World*, la qual representa (juntament amb les classes que en depenen) les estructures que treballen amb les dades.

Si ens fixem en la classe *World*, veiem que conté un escenari i varis grups, alguns d'individus controlats per l'usuari i d'altres d'enemic. Aquests grups, a al vegada, estan composats per un numero indeterminat d'individus.

En el capítol de d'*Anàlisis i Disseny* s'han descrit les funcionalitats i els atributs dels individus (*Character*), els grups (*Groups*) i l'escenari (*Scene*). Aquestes funcionalitats s'han respectat rigorosament en la implementació, de manera que el resultat ha estat l'obtenció d'un conjunt de mètodes en cada classe que encapsulen les diferents funcionalitats exigides a cada un d'aquests elements.

Els mètodes més importants de cada classe són els següents:

World

*void **Update**(unsigned long elapsedTime)*

Actualitza la situació. Rep per paràmetre el període de temps transcorregut des de l'última actualització. A efectes pràctics, aquest mètode l'únic que fa és una crida al metode *Update* dels diferents grups per a que s'actualitzin.

*void **ReadScene**(CString fName)*

Llegeix un fitxer d'escenari. El paràmetre d'entrada és el nom del fitxer (amb el *path*).

Scene

*bool **DetectCollision**(Vector* start, Vector* end)*

Detecta si hi ha col·lisions entre dos punts. Retorna *true* en cas afirmatiu i *false* si no detecta col·lisió.

*bool **CalculatePath**(CList<Vector, Vector&> *cpList, Vector *start, Vector *target)*

Calcula el camí mitjançant l'A* entre dos punts i retorna els *checkpoints* ordenats en l'atribut *cpList*. Retorna *true* si el camí retornat arriba fins l'objectiu i *false* en cas contrari.

Group

*void **Update**(unsigned long elapsedTime, World* world)*

Actualitza la situació del grup i fa que els individus que el formen s'actulitzin també. Rep per paràmetre el període de temps transcorregut des de l'última actualització i un punter al món per poder accedir a elements de l'escenari i d'altres grups.

*void **UpdateStatus**()*

Actualitza els atributs interns del grup com per exemple la posició mitjana, la proporció d'individus en estat de pànic, etc.

void ***RecalcNewGroupState****(Scene* scene)*

Comprova les condicions de sortida de l'estat actual del grup i canvia l'estat si es compleix alguna d'aquests condicions.

Character

void ***Update****(unsigned long elapsedTime, World* world, Group* pGroup)*

Actualitza la situació de l'individu. Rep per paràmetre el període de temps transcorregut des de l'última actualització, un punter al món per poder accedir a elements de l'escenari i un punter al seu propi grup.

void ***RecalcNewCharacterState****(Group *gr, Scene* scene)*

Comprova les condicions de sortida de l'estat actual i canvia l'estat si es compleix alguna d'aquests condicions.

void ***DoMove****(float turn, float acc, const unsigned long elapsedTime, Vector* targetPoint, Scene* scene, Group* pGroup)*

A partir d'una quantitat d'acceleració (*acc*), de la força perpendicular al moviment (*turn*) i del període de temps des de la última actualització calcula la nova posició de l'individu. El paràmetre *targetPoint* s'utilitza per saber la distància fins l'objectiu i així calcular si l'individu ha de començar a frenar per tal d'aturar-se progressivament i no passar-se de llarg. També rep un punter a l'escenari i un altre al grup al qual pertany per poder accedir a alguns dels atributs que contenen.

Paint

void ***Draw****(World* world, bool sceneLoaded)*

Dibuixa tot l'escenari amb els diferents individus i grups continguts en el punter *world*. El booleà *sceneLoaded* serveix per saber si hi ha quelcom a pintar o no.

bool ***SelectGroup****(CPoint, World* world)*

A patir d'un punt en coordenades de la pantalla *CPoint* (posició del ratolí) selecciona el grup que es troba en aquella posició.

void ***SetNewTarget****(CPoint, World* world)*

Assigna l'objectiu al grup seleccionat a patir d'un punt en coordenades de la pantalla *CPoint* (posició del ratolí).

void ***SetNewFormDirection****(CPoint, World* world)*

Assigna l'orientació desitjada al grup seleccionat a patir d'un punt en coordenades de la pantalla *CPoint* (posició del ratolí).

6.3.1 - La visualització

Com ja hem vist, tot el procés de visualització que fa ús de la llibreria *OpenGL* ha estat encapsulat en la classe *Paint* (juntament amb la implementació de la càmera) per tal de construir un codi amb coherència. D'aquesta manera es garanteix una certa independència entre la implementació del model de moviment i les funcionalitats de visualització.

Una instància d'aquesta classe és un atribut de la classe *CView* (de les MFC), el qual garanteix que pugui haver-hi diferents renderitzacions en diferents finestres, tal i com es pot veure a la figura 34, on s'aprecia un escenari amb múltiples vistes.

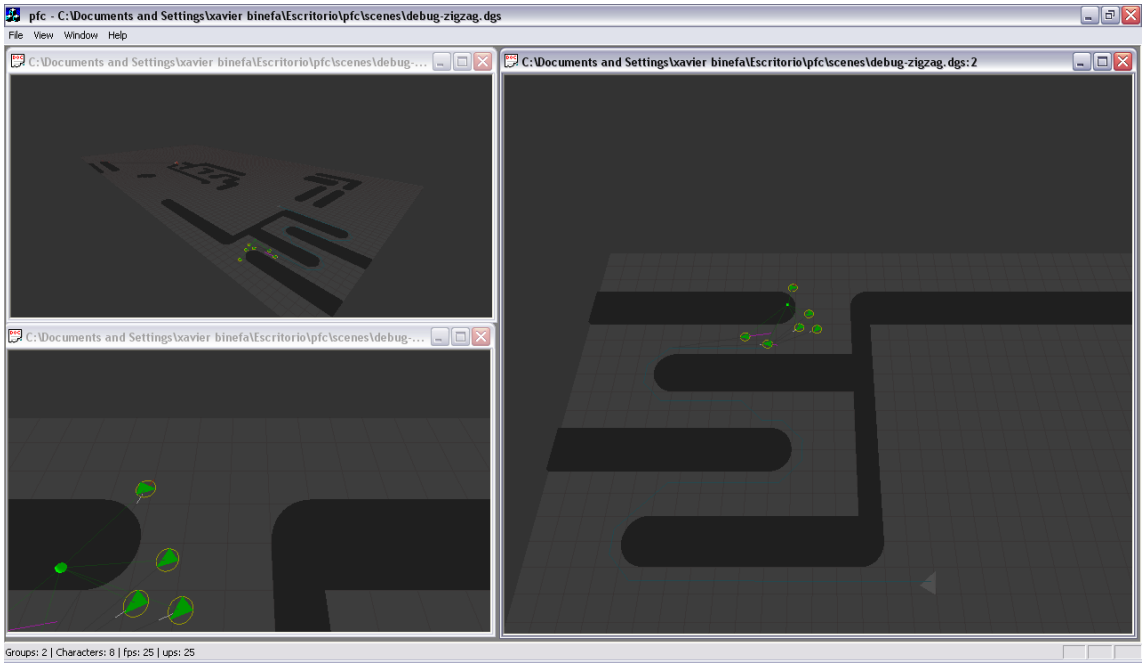


Figura 34. Múltiples vistes d'un sol escenari

Aquesta funcionalitat de múltiples vistes és fruit d'usar el model *Document-Vista* de les *MFC*. Aquest model independitza la informació de visualització de la que conté les dades de l'escenari, de manera que la classe anomenada *CDocumnet* conté una instància de la classe *World* (on es troben totes les dades dels individus, dels grups i de l'escenari) Mentre que la classe *CView* conté una instancia de la classe *Paint* (encarregada de gestionar la càmera i renderitzar l'escenari i els individus). Com que aquest model permet que un 'document' (classe

CDocument) contingui una llista de 'vistes' (classe *CView*), és possible tenir diferents renderitzats per a un sol escenari.

A part de tenir diferents vistes d'un sol escenari funcionant simultàniament, l'aplicació permet carregar múltiples escenaris, cadascun d'ells amb les seves pertinents vistes. Cal destacar que mentre que les diferents vistes d'un sol escenari s'actualitzen totes simultàniament, quan treballem amb diferents escenaris oberts, només s'actualitza un escenari i les seves vistes, de manera que els altres escenaris es queden aturats i ni recalculen posicions ni repinten la seva finestra.

Aquesta funcionalitat és conseqüència de que el tipus d'aplicació creada amb les *MFC* és MDI (*Multi Document Interface*), la qual té una llista amb varies instancies de la classe *CDocumet*, la qual, a la vegada, pot tenir varies instancies independents de *CView*.

Ja hem vist que totes les funcionalitats de pintar per pantalla les té l'anomenada classe *Paint*, la qual dibuixa tota l'escena. Aquesta impressió per pantalla es fa utilitzant les funcionalitats bàsiques de la llibreria de renderitzat *OpenGL*. Entre aquestes funcionalitats destaca l'ús de transparències per tal de pintar les línies de força sobre el terra o la utilització del *Z-Buffer* per tal de donar prioritat a algunes coses a l'hora de ser pintades tridimensionalment. Aquest ús és útil per exemple quan es pinten les línies de força sobre el mateix pla del terra, com aquestes es pinten desactivant el *buffer* de profunditat, l'escenari sempre queda tapat per les petites rectes.

- La il·luminació

En l'aplicació implementada s'han utilitzat dos tipus de llums diferents per tal de renderitzar d'una manera mínimament estètica l'escenari i els individus que s'hi desplacen per sobre.

La llibreria *OpenGL* permet utilitzar 3 tipus de llums diferents per tal de simular els resultats d'un algoritme traçador de raigs (*ray-tracing*). Aquestes tres llums possibles són:

- Llum ambiental

- Llum difusa

- Llum especular

Evidentment aquestes tres possibles propietats de reflexió de la llum per part dels materials són perfectament combinables entre elles. I és aquesta combinació la que genera unes imatges força realistes, al aportar sensació de volum.

Però com ja hem dit, l'aplicació implementada només utilitza dues d'aquestes propietats, la llum ambiental i la llum difusa.

La llum ambiental és aquella que no prové de cap punt en particular, sinó que són fruit de múltiples reflexos en les parets i objectes de l'escena a renderitzar. Així doncs, els objectes il·luminats amb aquest tipus de llum són igualment lluminosos en totes les seves superfícies i direccions. Aquesta llum té la funcionalitat d'aportar luminescència a totes les superfícies, de manera que la llum difusa afegeixi llum sobre un model ja il·luminat, per tal d'evitar zones massa fosques.

La llum difusa, a diferència de la llum ambiental, sí que prové d'una direcció determinada. Aquesta llum incideix sobre les diferents superfícies dels objectes i es reflexa en major o menor mesura depenent de l'angle entre el vector d'incidència de la llum i la normal de

la superfície. Quan més paral·lels siguin aquests dos vectors més brillant apareixerà la superfície. D'aquesta manera es pot crear una major sensació de profunditat i tridimensionalitat.

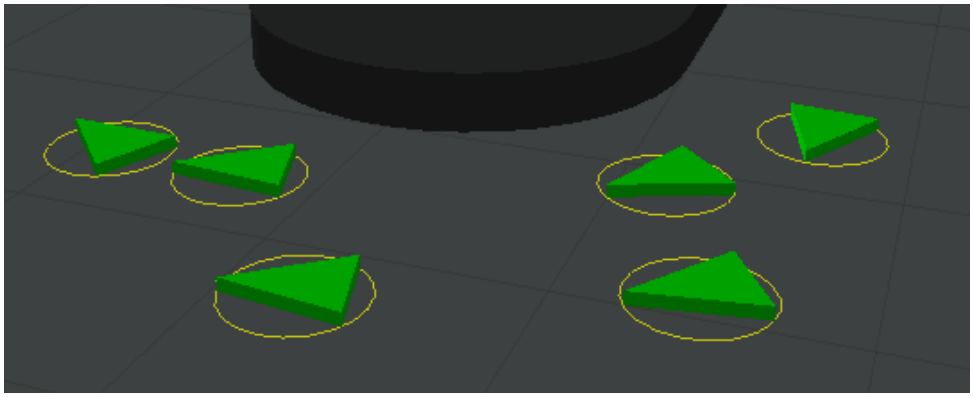


Figura 35. Escena renderitzada amb llum difusa i ambiental

A la figura 35, es pot veure una escena renderitzada amb llum ambiental i llum difusa, de manera que les diferents cares dels individus i de la paret que es veu al fons, reben diferent quantitat de llum, depenent de la seva orientació. En aquest cas en concret, es pot deduir que la llum ve del fons cap endavant (ja que les cares que miren cap a la càmera estan més enfosquides) i de dalt cap a baix, ja que les superfícies superiors reflecteixen més quantitat de llum.

- L'elecció dels colors i la simbologia utilitzada

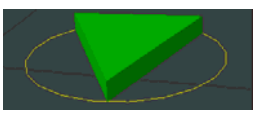
Per tal de facilitar la comprensió de la situació en un moment determinat de la simulació s'han utilitzat diferents colors per tal de representar una sèrie d'atributs dels individus i els grups.

Mentre que el color del terra o de les parets sí que és absolutament arbitrari i no guarda cap relació amb atributs canviables de l'escenari, el color amb el que es pinta un individu, les línies que l'uneixen fins l'esfera que representa el centre del grup o les forces que s'apliquen a cada individu segueixen un codi cromàtic predefinit.

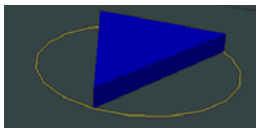
Aquest codi es basa en els estat d'un individu, de forma que cada un dels cinc estats possibles té assignat un color que el representa. Aquest color s'aplica tant als individus (per representar l'estat de cada individu) com als grups (representant l'estat del grup).

Aquests colors són els següents:

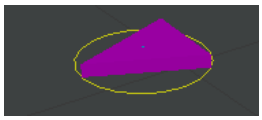
- Verd (estat Avançar)



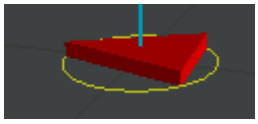
- Blau (estat Formar)



- Lila (estat Reagrupar)



- Vermell (estat Fugir)



- Taronja (estat Intimidar dels enemics)

Però no només els estats es representen mitjançant l'ús de colors. Les forces que s'apliquen sobre un individu també tenen una coloració específica que les fa fàcilment identificables. Aquestes forces són:

- Blanc (*Força cap a l'objectiu*)
- Lila (*Força de cohesió*)
- Vermell (*Força de repulsió*)

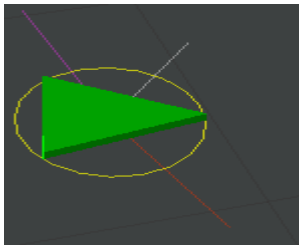


Figura 36. Cromàtica de les forces.

A la figura 36 es poden veure aquestes 3 forces aplicades sobre un individu en estat d'avançar. Només donant un cop d'ull a la imatge es pot veure que rep una força de repulsió cap

a la dreta, una força de cohesió cap a l'esquerra i la força cap a l'objectiu en la direcció anterior esquerra respecte l'individu.

Per altra banda, hi ha diferents atributs que es representen mitjançant una altra simbologia que no sigui la cromàtica.

El primer d'aquests elements és la por de cada individu. A cada individu se li pinta una línia vertical a sobre que representa el nivell de por. Aquesta línea serà més llarga quanta més por tingui l'individu, a la figura 37 es pot veure aquesta línia blava sobre els diferents individus. Els individus més propers als enemics (situats a l'esquerra de la imatge) tenen major nivell de por, el qual es reflexa en una barra vertical més llarga.

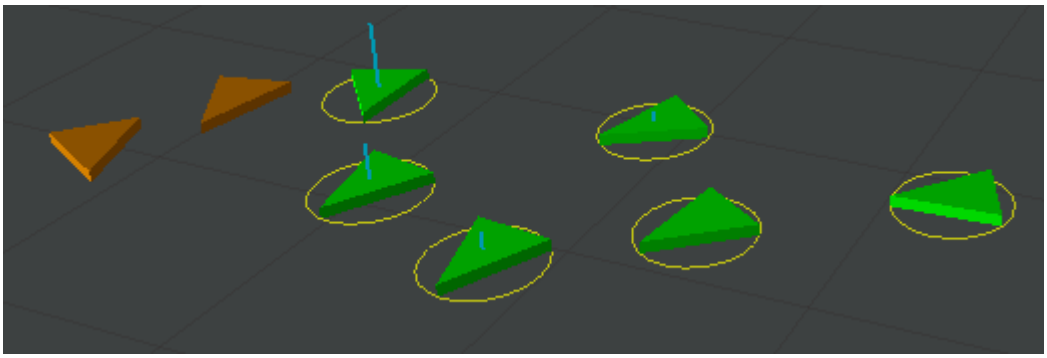


Figura 37. Representació del nivell de por mitjançant barres verticals.

Hi ha dos atributs d'un grup que també es representen mitjançant una simbologia pròpia. Aquests dos atributs es representen de forma combinada en una esfera que representa la posició ponderada del centre del grup i la dispersió dels seus individus. Aquesta representació es fa dibuixant una esfera (del color corresponent a l'estat del grup) sobre la vertical del centre de masses ponderat del grup (situat sobre la superfície del pla). L'alçada a la que es trobi aquesta esfera representa la dispersió mitjana dels seus individus. Així doncs, una esfera molt alta indicarà que el grup té molta dispersió, mentre que una esfera molt baixa indicaria que el grup està molt unit. En l'exemple de la figura 38 es pot veure un grup força unit (amb baixa dispersió) i per tant l'esfera es pinta relativament propera al pla.

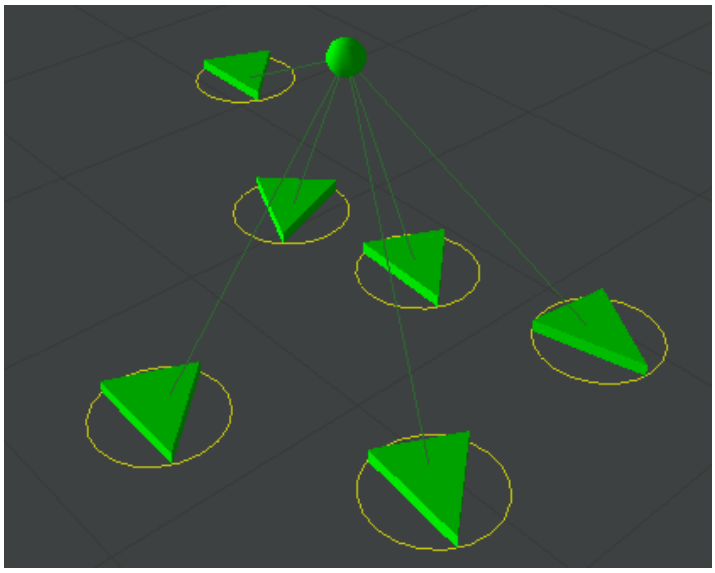


Figura 38, representació del grup.

Existeix un quart atribut que es representa utilitzant primitives de dibuix. En aquest cas estem parlant d'un cercle groc que envolta els individus (es pot veure a la figura 39), el qual indica que el grup que composen està actualment seleccionat i per tant es pot modificar l'objectiu i la direcció d'alineament desitjada per a la formació.

Finalment en l'escenari podem trobar un altre element que simbolitza informació d'un grup. Es tracta d'un triangle gris que es dibuixa sobre la superfície del terra, tal i com es pot veure a la figura 39. Aquest triangle està situat sobre l'objectiu del grup i orientat segons la direcció d'alineament del grup.

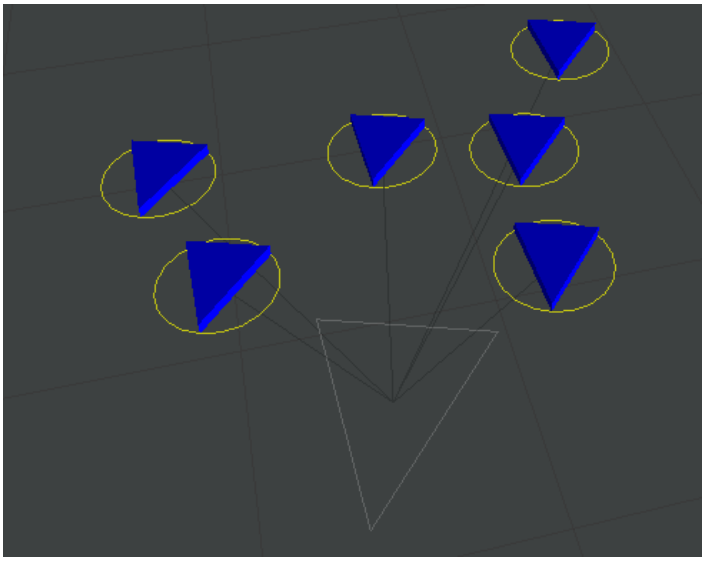


Figura 39. Grup en formació proper a l'objectiu i alineat.

- La càmera

Per tal d'aconseguir una bona visualització que permeti a l'usuari observar amb detall el que succeeix durant la simulació cal que la càmera que s'utilitza per al renderitzat sigui perfectament manipulable en totes les direccions, mentre que a la vegada sigui senzilla d'utilitzar.

Per tal d'obtenir aquesta senzillesa s'ha optat per oferir dues possibilitats de càmera, una primera configuració de la càmera permet moure-la lliurement per l'escenari, mentre que l'altra possibilitat es caracteritza per seguir el centre del grup que estigui seleccionat, de manera que la persona que estigui visualitzant la simulació pugui veure perfectament les reaccions d'un sol grup.

Però totes dues possibilitat es caracteritzen per utilitzar la mateixa càmera, amb la única diferencia que quan es segueix un grup, el punt focal (la posició on mira la càmera) està restringit i sempre serà el centre ponderat del grup, i per tant no es podrà desplaçar segons la voluntat de l'usuari.

Per al que fa a la resta de les propietats de la càmera implementada, es pot dir que és una càmera tridimensional caracteritzada per 5 elements que la defineixen. Aquests 5 elements són els següents:

- Punt focal (coordenada tridimensional cap on mira la càmera)

- Vector de direcció (vector tridimensional que diu l'orientació de la càmera)

- Vector que indica la normal de la càmera (per saber quina es la part de dalt)

- Vector cap a la dreta (per a fer les rotacions, a l'igual que la normal)

- Distància entre l'ull i el punt focal.

A part d'aquests 5 elements, la càmera conte varis mètodes que permeten desplaçar el centre focal i fer les rotacions segons els 3 eixos de la pròpia càmera (davant, dreta, amunt).

Però una càmera tant configurable té la contrapartida que és complicada d'utilitzar, per aquest motiu s'han afegit dues restriccions a la càmera que la fan més senzilla d'utilitzar.

La primera restricció és que el punt focal sempre estarà sobre la superfície del pla, de manera que no es podrà desplaçar en l'eix vertical, el qual no ens interessa degut a que treballem sobre una superfície plana.

La segona restricció, ha estat bloquejar l'eix vertical de la càmera, de manera que independentment de les rotacions que pateixi la càmera, aquesta sempre tindrà la seva normal alineada amb l'eix vertical. Això es pot traduir com que la càmera no podrà balancejar-se lateralment, i per tant mai estarà inclinada cap a un costat.

Per altra banda, per definir una càmera en *OpenGL*, cal especificar tant el punt focal com la posició de la càmera. I tot i que aquesta posició no s'emmagatzema és molt fàcil obtenir-la a partir del punt focal al qual se li resta el vector de direcció multiplicat per l'escalar que representa la distància entre l'ull i el punt focal.

- La interacció de l'usuari

Una altra de les funcionalitats derivades d'utilitzar la llibreria *OpenGL*, és la de poder interactuar amb el ratolí directament sobre la geometria de l'escenari, de forma que seleccionar un grup es tan senzill com fer 'clic' amb el ratolí sobre el punt on esta situat qualsevol dels seus membres.

Aquesta facilitat, fruit d'unir la interactivitat amb la renderització, fa que els mètodes implementats per a la interacció de l'usuari també hagin anat a parar a la classe *Paint*, ja que depenen de la llibreria utilitzada, i en el cas de que es vulgues canviar aquesta llibreria, el codi s'hauria de reescriure.

Les possibles accions d'un usuari amb el ratolí son 6, que detallem a continuació:

- Rotar la càmera

Mitjançant la combinació de prémer la tecla '*control*', el botó esquerre del ratolí i el desplaçament del ratolí, s'obté la rotació de la càmera (el que rota són els 3 vectors que defineixen l'orientació tridimensional de la càmera: *forward*, *up* i *right*). Si el desplaçament és vertical la càmera mirarà més amunt o més avall. Si el desplaçament és horitzontal, la càmera gira al voltant de l'eix vertical.

Aquesta rotació no altera mai el punt focal, ja que el que canvia és l'orientació, i per tant la posició on es trobaria l'ull de l'observador. En altres paraules, es segueix mirant al mateix punt, però des d'un altre angle.

- Desplaçar la càmera sobre el pla

Mitjançant la combinació de prémer la tecla '*majúscules*', el botó esquerre del ratolí i el desplaçament del ratolí, el punt focal de la càmera es modifica, desplaçant-se sempre per la superfície de l'escenari.

- Allunyar i apropar la càmera

Mitjançant la combinació de prémer la tecla '*control*', el botó dret del ratolí i el desplaçament vertical del ratolí s'obté l'augment de la distància entre l'ull i el punt focal. Com que el punt focal es manté constant, el que s'allunya és l'ull de la càmera, creant un efecte de reducció de zoom (o d'augment de zoom si és que s'hi apropa).

- Seleccionar un grup

Si es *clica* sobre un individu que es mostri per pantalla, sense prémer cap altra tecla, el grup al qual pertany aquest individu passarà a estar seleccionat.

- Assignar un punt de destí al grup seleccionat

Si hi ha algun grup seleccionat i es prem el botó dret del ratolí sobre qualssevol punt de l'escenari que estigui dins d'una cel·la buida, aquest punt passarà a ser el nou objectiu del grup seleccionat.

- Modificar la direcció d'alineament del grup seleccionat.

Quan es defineix un nou objectiu per a un grup, es pot mantenir apretat el botó dret del ratolí i desplaçar el punter de manera que la direcció d'alineament es correspongui al vector entre aquet nou punt i l'objectiu.

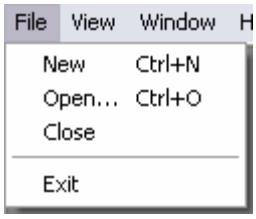
- Els menús

Una altra de les funcionalitats que proporciona l'estructura de les *MFC* són els menús que apareixen en l'aplicació. Aquests menús permeten utilitzar l'aplicació de forma fàcil i intuïtiva per a qualsevol persona, ja que es tracta d'una presentació àmpliament estandarditzada per a milers d'aplicacions quotidianes implementades amb *MFC's*.

A través dels menús es poden realitzar gran varietat d'accions, des d'obrir un arxiu d'escenari fins a escollir quines forces s'ha de mostrar en la simulació.

Tot seguim detallem quines són aquestes funcionalitats, agrupades en els diferents menús de la part superior de l'aplicació.

Menú 'File'



Aquest menú conté les funcionalitats de crear una nova finestra on treballar, obrir un escenari (sobre una finestra prèviament creada), tancar una simulació o sortir del programa.

Menú 'View'

Aquest altre menú permet seleccionar quins elements s'han de veure durant la simulació.

La primera opció (***Status Bar***) fa referència a la barra inferior de la finestra (figura 40) on s'hi poden veure dades de gran utilitat. Aquestes dades són: El número de grups i d'individus que conté la simulació, la mida de l'escenari, els fotogrames (*frames*) per segon i les actualitzacions (*updates*) per segon.

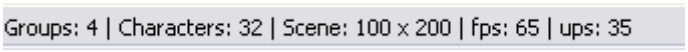


Figura 40. Status Bar (Barra de estat) amb informació sobre la simulació.

Després tenim un conjunt d'opcions seleccionables (totes seleccionades per defecte) sobre quina informació mostrar per pantalla.

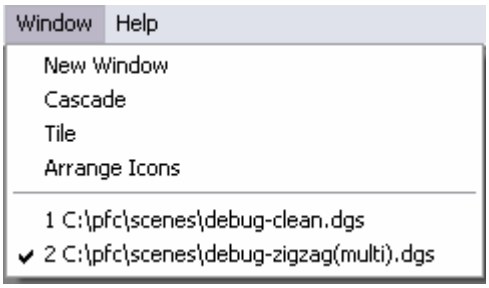
'***Cohesion Force***' fa referència a la força de cohesió (línea lila). '***Collision Force***' fa referència a la força amb repulsió de parets i individus (línea vermella). I '***Target Force***' és la força cap a l'objectiu (línea blanca).

Finalment tenim dues opcions més. '***Path***' fa referència a totes les línies del terra que simbolitzen el camí a seguir fins l'objectiu. Aquest camí pot ser el normal d'un grup fins l'objectiu (en blau cel), el camí de cada individu fins al centre del grup quan es reagrupen (en lila) o el camí dels enemics (en vermell).

Cal recordar que aquests camins no existeixen sempre, sinó que només es pinten quan el camí no és directe i ha requerit de l'ús de l'algoritme de cerca de camins.

Finalment tenim la possibilitat d'activar o desactivar la informació del grup, mitjançant l'opció '***Group***', la qual mostrarà una esfera que representa el centre de masses ponderat del grup i la dispersió, juntament amb unes línies que uneixen els individus amb la citada esfera.

Menú ‘Window’



En aquest menú desplegable hi trobem les funcionalitats referents a la gestió de múltiples vistes d’un sol escenari.

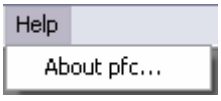
L’opció ‘***New Window***’ permet crear una nova vista (classe *CView* de l’estructura document-vista de les *MFC*), amb una visualització independent e les mateixes dades.

‘***Cascade***’ és l’opció per a endreçar les finestres en cascada.

‘***Tile***’ redimensiona i ordena les finestres, mentre que ‘***Arrange Icons***’ endreça les finestres en el cas de que estiguin minimitzades, sinó no fa res.

Finalment hi trobem un llistat amb les diferents vistes obertes per tal de poder-ne seleccionar l’activa.

Menú ‘Help’



Aquest últim menú només permet una opció: ‘***About pfc***’, la qual mostra per pantalla una petita finestra de diàleg sobre els crèdits de l’aplicació.

6.3.2 - El control del temps. Divisió dels blocs de pintat i de càlcul

Quan es treballa amb entorns complexos que requereixen actualitzacions costoses i constants, l’ús del processador s’ha de racionalitzar el màxim possible per evitar colls d’ampolla en l’aplicació, el qual provocaria un mal funcionament de la simulació, produint-se possibles errors de càlcul o de visualització.

Un dels requeriment no funcionals de l’aplicació és mostrar com a mínim 25 *frames* per segon per tal de crear una sensació de moviment fluida. Aquests 25 *fps* (*frames per second*) signifiquen que tots el mètodes que dibuixen l’escenari s’han d’executar, com a mínim, 25 vegades en un segon. Per altra banda, l’aplicació també ha d’estar calculant constantment les noves posicions i els diferents canvis en l’escenari. Seguint aquest punt de vista, l’aplicació està dividida en dos blocs separats, per una banda tenim la part de recaluclar les dades (*Recalc*), i per l’altra la de pintar l’escenari (*Redraw*). Aquests dos mètodes es podrien cridar de forma consecutiva i constant, però això provocaria que per tal de pintar l’escena 25 vegades per segon, l’escena en sí hauria de recalcular-se 25 vegades en un segon, el qual pot ser força crític en alguns moments puntuals on es requereixen càlculs molt costosos per tal d’actualitzar les dades de l’escenari (com per exemple, quan es fa servir l’algoritme de cerca de camins A*).

Per tal d’evitar que els *fps* baixin, l’escena es repinta de forma constant, mentre que en el moment de calcular les dades de l’escenari, es mira quan de temps fa que s’ha actualitzat i si no es supera un cert període no es calcula. Aquest procediment queda resumit en el diagrama d’activitats de la figura 41, on es pot veure una simplificació del bucle principal de l’aplicació.

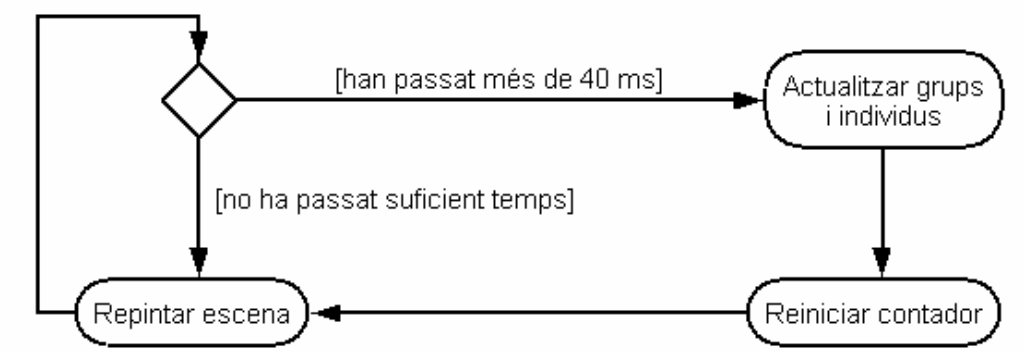


Figura 41. Diagrama d'activitat del Recalc() i el Redraw().

6.3.3 - La integració de l'algoritme de cerca de camins

Tota aplicació tendeix a tenir uns punts on és possible que es requereixi tanta capacitat de càlcul que el fluid funcionament de la simulació es vegi interromput de forma constant o intermitent. Aquests punts són els anomenats colls d'ampolla i evidentment, s'han d'evitar a qualsevol preu.

Ja hem vist, que per a evitar aquesta saturació del processador l'escena es recalcula de forma intermitent, donant prioritat a la renderització. Això és una estratagema per tal d'evitar colls d'ampolla, però no és la única.

L'algoritme de cerca de camins emprat és l'A*, el qual consisteix en recorrer les diferents cel·les possibles per tal de buscar el camí més curt. En el cas de que el punt de destí i el d'origen estiguin molt separats i amb molts obstacles pel mig, la cerca combinada d'amplada i profunditat que realitza l'algoritme d'A*, pot requerir una gran quantitat de memòria i de temps de càlcul.

Per tal d'evitar que es produeixin pics de retard deguts a l'ús intensiu d'aquest algoritme en alguns moments determinats de la simulació, s'ha limitat la capacitat de cerca de manera que hi ha una possibles sortida prematura que farà aturar la cerca i es retorni un camí parcial, el qual no arriba fins l'objectiu sinó que només s'hi aproximarà. Aquesta condició és que el número de nodes visitats no superi els 1500. Aquest límit és absolutament arbitrari i podria ser substituït per un límit variable depenent de la capacitat de càlcul de l'ordinador utilitzat. Però per a fer això s'haurien d'utilitzar costos mètodes de gestió del temps, el qual encara accentuaria més el coll d'ampolla. Per altra banda, una limitació que retornes camins sempre incomplerts alterarien excessivament la simulació, de manera que a vegades és millor assegurar-se tenir un camí mínimament llarg a canvi de tenir algun salt en la visualització, que no pas arriscar-se a que el maquinari alteri el resultat de la simulació.

De totes maneres, s'ha testejant l'aplicació amb diferents combinacions de maquinari per tal de comprovar que aquest límit, parcialment arbitrari, sigui òptim per a un ordinador no gaire desfasat a data d'avui.

Evidentment, en el cas de que s'explorin massa nombre de nodes i l'algoritme retorni un camí incomplert, el grup (o l'individu) activa una *flag* per tal de saber que quan arribi al aparent objectiu, en comptes d'aturar-se, haurà de recalcular el camí fins l'objectiu real, emmagatzemat en una variable especial per a aquest cas.

6.3.4 - La selecció interactiva dels grups i l'assignació d'objectius

Una de les funcionalitats que ofereix la llibreria *OpenGL* és la de canviar el mode de renderitzat de manera que es pugui saber quin objecte hi ha sota el ratolí. Aquest procediment és molt simple i consisteix en dibuixar certs objectes en un *buffer* particular anomenat *buffer* de selecció (que no pinta per pantalla) i etiquetar cada primitiva (o grup de primitives). D'aquesta manera es poden pintar només els objectes seleccionables per tal d'evitar obstruccions amb parets o altres objectes.

De fet, en aquesta aplicació l'únic element seleccionable son els individus. Així doncs, l'únic que es pinta son aquest individus, però ni tan sols en la seva forma triangular, sinó que es pinten esferes del mateix radi que els individus per tal de facilitar la selecció en casos que l'usuari no *cliqui* exactament sobre un individu. Evidentment tots els individus d'un mateix grup es pinten amb la mateixa etiqueta, ja que el que volem realment es seleccionar un grup mitjançant un individu, independentment del individu en concret que es seleccioni.

En cas de que hi hagi dos individus, un davant del altre, es seleccionarà el més proper a la càmera.

Per altra banda també tenim la selecció dels objectius mitjançant el ratolí. Aquesta selecció no es fa utilitzant el *buffer* de selecció, sinó que es calcula una recta entre la posició de la càmera i el punt bidimensional mitjançant el mètode *gluUnProject()* de la llibreria *OpenGL* (utilitzant la posició del ratolí en *pixels*) i el qual retorna un punt tridimensional de l'escena.

Un cop tenim la posició de la càmera i el punt en coordenades absolutes de l'escenari només resta definir la recta que passa per aquests dos punts i buscar la intersecció d'aquesta recta amb el pla que defineix el terra de l'escenari.

6.4 - Els arxius d'escenari

Els escenaris, juntament amb els individus i els enemics que prenen part en una simulació, es carreguen des d'un arxiu de text acabat amb l'extensió '*dgs*'. Aquests fitxers contenen tota la informació per tal de crear un entorn de simulació complert.

L'estructura de l'arxiu està dividida en 3 blocs. Primerament tenim la informació sobre els diferents grups controlats per l'usuari, després un bloc per als enemics i finalment l'escenari.

Per tal de deixar més clara l'estructura d'un escenari s'ha adjuntat en l'annex un escenari d'exemple de 40 cel·les d'alt i 40 d'ample, amb varis grups i enemics.

- El primer bloc: *Els grups*

La primera línia d'aquest bloc indica el número de grups que podrà controlar l'usuari. Després hi ha una línia en blanc i tot seguit la informació dels diferents grups.

La informació de cada grup consisteix en una línia on s'indica quants individus té el grup i seguidament les 7 propietats i atributs de cada individu. Aquests 7 atributs són:

- *maxSpeed* (velocitat màxima)
- *maxAccForce* (màxima força aplicable al centre de masses)
- *maxBrakeForce* (màxima força de frenada)
- *maxTurnForce* (màxima força de gir)
- *maxFear* (límit de por)
- *position* (posició inicial)
- *forward* (vector de direcció inicial)

Un exemple d'un grup amb dos individus seria com el següent:

```
n_groups: 1

group_nCharacters: 2

character_maxSpeed: 5.000
character_maxAccForce: 1.000
character_maxBrakeForce: 3.000
character_maxTurnForce: 0.1000
character_maxFear: 15.000
character_position: 20.500 -10.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 10.000
character_maxAccForce: 4.000
character_maxBrakeForce: 3.000
character_maxTurnForce: 0.1000
character_maxFear: 15.000
character_position: 23.500 -10.500 0.000
character_forward: 0.000 1.000 0.000
```

Figura 42. Exemple d'un grup format per 2 individus.

- El segon bloc: *Els enemic*

Aquest bloc és idèntic a l'anterior, simplement es canvien els noms dels atributs (ja que són d'enemic i no d'individu). La primera línia indica el número de grups d'enemic que hi haurà en l'escenari i seguidament la informació d'aquests grups amb les 7 propietats de cada individu que el forma.

La figura 43 pot servir d'exemple per tal d'il·lustrar el concepte.

```
n_enemy_groups: 1

group_nEnemies: 1

enemy_maxSpeed: 4.500
enemy_maxAccForce: 1.800
enemy_maxBrakeForce: 3.000
enemy_maxTurnForce: 0.1000
enemy_position: 305.500 -10.500 0.000
enemy_forward: 0.000 1.000 0.000
```

Figura 43. Exemple d'un grup d'enemics amb un sol individu.

- El tercer bloc: *L'escenari*

Aquest bloc comença amb dues línies on s'especifica l'amplada i l'alçada de l'escenari en cel·les. Seguidament hi trobem tantes línies com files tingui l'escenari (cada línia representa una fila de cel·les). A la vegada, aquestes línies tenen tants caràcters com cel·les hi hagi horitzontalment en l'escenari.

Un petit exemple seria el següent:

```
scene_width: 10
scene_height: 4
----------
---1002---
---3004---
----------
```

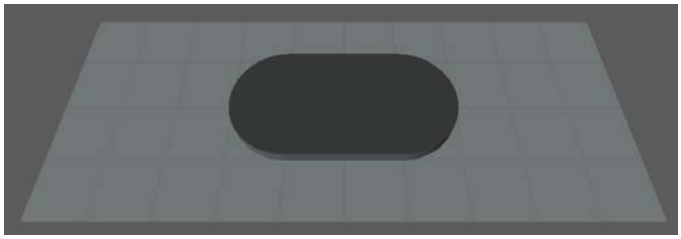


Figura 44. Exemple d'un escenari senzill de 40 cel·les.

Com es pot veure en l'exemple de la figura 44, hi ha 6 tipus de caràcters diferents per a construir l'escenari, els quals es corresponen als 6 possibles tipus de cel·la que pot tenir l'escenari.

- '1' – Cel·la rodona amb el centre en el vèrtex inferior – dret.
- '2' – Cel·la rodona amb el centre en el vèrtex inferior – esquerre.
- '3' – Cel·la rodona amb el centre en el vèrtex superior – dret.
- '4' – Cel·la rodona amb el centre en el vèrtex superior – esquerre.
- '0' – Cel·la plena quadrada.
- '-' – Cel·la buida

D'aquesta manera, es fàcil i intuïtiu generar un escenari complex a partir d'un fitxer de text, de manera que donant una simple ullada al text de l'escenari, una persona mínimament familiaritzada amb l'aplicació, enseguida hi descobrirà la geometria de l'escenari.

7. Conclusions i treball futur

Un cop implementada l'aplicació capaç de simular el moviment dinàmic de grups, cal aturar-se per mirar tot el procés des d'una certa distància i valorar la feina realitzada.

Aquesta valoració ha de ser primerament objectiva, mirant si es compleixen tots els requeriments especificats en les primeres fases i observant del projecte i analitzant els resultats obtinguts en la fase de proves i testeig final.

7.1 - Validació dels requeriments funcionals

El més important a l'hora d'avaluar l'aplicació resultant és comprovar que les especificacions requerides inicialment s'hagin complert.

Els requeriments funcionals que ara es llisten per validar si han estat implementats estan degudament detallats i comentats en el capítol 2, titulat *Anàlisi de requeriments*.

- Grups formats per un mínim d'entre 2 i 10 individus

A l'aplicació resultant, el nombre d'individus d'un grup afecta al tipus de desplaçament que té, ja que quan major és el nombre d'individus que formen el grup, els individus a els extrems, en estar més allunyats del nucli que en un grup normal, tenen una major força de cohesió, fent que premin cap a l'interior i el grup tingui un comportament més rígid que si fos menys nombrós. De totes maneres tot i que aquest comportament és detectable, grups de 20 o fins i tot 30 individus poden desplaçar-se sense gaires dificultats per l'escenari. A major nombre d'individus, fer passar el grup per un passadís estret es fa gairebé impossible degut a l'estirament que es produeix al grup, mentre els individus cada cop tendeixen més a desplaçar-se cap al centre en comptes de cap a l'objectiu.

Però en grups d'entre 2 i 10 individus, aquesta variació de comportament és mínima i els grups tenen un moviment fluid i coherent.

- Interacció de l'usuari: Modificació d'objectius en temps real

L'usuari, efectivament, pot interactuar en temps real, modificant els objectius dels diferents grups.

- Varietat en les propietats dels individus

Els individus són fàcilment configurables en la totalitat dels seus atributs mitjançant els fitxers de configuració descrits a l'apartat "6.4 - Arxius d'escenari".

- Escenaris complexos amb parets, passadissos, habitacions, etc.

Finalment, els escenaris poden ser realment grans. Concretament s'han provat amb èxit escenaris de fins a 20.000 cel·les. Si tenim en consideració que hi ha 6 tipus de cel·les es pot calcular que, com a mínim, hi ha $10^{15.536}$ escenaris diferents possibles d'aquesta mida. Tot i que només una petita porció d'aquesta xifra siguin escenaris viables (amb una organització coherent de les parets) la quantitat és immensa.

Per altra banda, la implementació de l'algoritme de cerca de camins fa que (si existeix un camí possible) els grups vagin fins a l'objectiu sense quedar-se encallats.

En contrapartida s'ha de dir que si es treballa amb grups grans, és aconsellable no fer passadissos massa estrets (mínim dues cel·les d'amplada) i preferiblement evitar els angles rectes mitjançant la utilització de parets rodones que fan que el moviment sigui molt més suau i realista.

- Escenaris definits en arxius per a carregar directament

Aquesta funcionalitat està implementada i funciona correctament. De fet, la seva implementació ha estat molt útil per poder fer proves amb l'aplicació durant tota la fase d'implementació del projecte, ja que ha permès veure les mancances i les virtuts de les funcionalitats implementades en cada moment.

- Distribució i alineament dels individus en assolir l'objectiu

La utilització de l'estat *Formar* permet aquest alineament quan un objectiu arriba al seu destí.

- Detecció de col·lisions i evitar obstacles

La combinació de les forces de repulsió de les parets amb la utilització d'un algoritme de cerca de camins permet que grups d'individus es desplacin per escenaris complexos, evitant i rodejant els obstacles.

- Interacció amb enemics

La por generada per la proximitat dels enemics provoca el canvi de comportament desitjat en els individus.

- Possibilitat de fugides massives

La propagació del pànic entre els individus quan aquests comencen a fugir, provoca un augment de pànic dels individus més propers que en alguns casos , efectivament, provoca fugides massives de tot un grup.

7.2 - Validació dels requeriments no funcionals

Un cop comprovat que tots els requeriments funcionals s'han complert, per tal de poder assegurar que l'aplicació implementada compleix les especificacions per a les quals va ésser pensada i dissenyada, només ens falta mirar si també es compleixen els requeriments no funcionals.

En aquest apartat només hi havia dos requeriments a complir, que són:

- Eficiència en temps i 'frames' per segon:

Tal i com s'argumenta en el capítol de requeriments, l'aplicació ha de ser capaç de simular un escenari de 10.000 cel·les amb 100 individus i 5 enemics a 25 *fps* (*frames* per segon).

Feta la prova en un ordinador amb la configuració de la taula de la figura 2, s'han obtingut els següents resultats, depenent del número de vistes:

Número de vistes	frames per segon	updates per segon
1 vista	120 fps	40 ups
2 vistes	65 fps	40 ups
3 vistes	40 fps	40 ups

Figura 45. Taula de resultats.

Així doncs, és pot afirmar que l'aplicació compleix perfectament aquest requeriment no funcional, fins i tot triplicant la quantitat de vistes estipulades inicialment.

Per altra banda, en proves realitzades en altres configuracions de hardware més potents, una escena de 20.000 cel·les i 100 individus pot arribar a ser renderitzada a més de 300 *frames* per segon. Això demostra clarament l'eficiència del renderitzat, mentre que les actualitzacions lògiques (*updates*) estan restringides al voltant dels 40 *ups* per tal de no consumir recursos de forma innecessària.

- Consum de Recursos (Memòria):

En les especificacions dels requeriments, s'exigeix que en cap cas l'aplicació hauria d'ocupar més de 512 MB (*MegaBytes*) de memòria RAM.

Segons les proves realitzades, la simulació en simultani de tres escenaris de 20.000 cel·les, amb més d'un centenar d'individus cada un, no supera els 20 MB de memòria RAM i aquesta quantitat no augmenta amb el temps, el qual ens indica que l'aplicació mai arribarà a

requerir tanta memòria com per a considerar-se ineficient en termes de consum d'espai, en comparació amb els 512 MB de limitació inicial.

7.3 - Limitacions i propietats del model aplicat

Tot i que l'aplicació compleix tots els requeriments funcionals i no funcionals estipulats a la primera fase de l'elaboració d'aquest projecte, la utilització de certes premisses i metodologies impossibilita que el model escollit sigui capaç de treballar de forma perfecta en totes les situacions possibles.

Així doncs, l'aplicació d'un model de forces determinat, la utilització d'un sistema de locomoció específic o la possible sortida prematura de l'algoritme de cerca de camins, fan que la solució proposada i implementada tingui certes limitacions.

7.3.1 - Limitacions de l'escenari

La utilització d'un escenari dividit en cel·les homogènies està perfectament justificada pels avantatges que aporta comentades al capítol d'*Anàlisis i Disseny*. Però aquests avantatges tenen la contrapartida que les possibles configuracions de l'escenari estiguin limitades a una combinació finita de tipus de cel·les.

Una altra limitació del escenari és que els passadissos han de permetre el pas de dos individus en sentits oposat, ja que els individus i els grups no tenen cap mecanisme capaç de diagnosticar un embossament en una cavitat estreta i actuar en conseqüència buscant un camí alternatiu. Així doncs, per a que es compleixi aquesta condició d'amplada mínima, qualsevol passadís haurà de tenir dues cel·les d'amplada, ja que d'aquesta manera hi caben fins a 3 individus en paral·lel.

Per altra banda, tot i no ser estrictament necessari, és molt aconsellable fer ús de les cel·les de paret rodones per tal de facilitar el moviment als individus. Aquest comportament es deu a que les parets rodones apliquen progressivament la força de repulsió en els girs, mentre que si un individu voreja una cantonada en angle recte pot xocar-hi i alterar notablement la seva trajectòria. Aquest xoc, en algunes situacions, pot provocar una separació de l'individu del grup i el posterior reagrupament de tot el conjunt, i tot i que es reprengui la marxa i el grup acabi per arribar a l'objectiu, si s'eviten aquests angles rectes la simulació guanya més realisme.

De totes maneres, concebre l'escenari com una graella regular es pot considerar encertat, ja que compleix eficientment els requeriments de navegació que se li exigien. Concretament, permet trobar camins entre dos punts qualssevol i evitar col·lisions de forma precisa i ràpida.

7.3.2 - Limitacions de l'algoritme de cerca de camins

Per tal d'evitar que l'algoritme de cerca de camins emprat generi salts en el rendiment de l'aplicació, s'ha limitat en quant a número de nodes a explorar. D'aquesta manera si porta

més de 1500 cel·les visitades sense trobar cap camí fins l'objectiu, retorna un camí parcial i indica a l'individu que quan arribi a aquest objectiu temporal recalculi el camí.

Aquesta solució funciona molt bé en la immensa majoria dels casos, mentre que a la vegada assegura l'eficiència de l'aplicació.

Però es pot produir que un escenari tingui zones inaccessibles, aïllades de la part de l'escenari on es troba el grup amb el que estem treballant. Si es fixa l'objectiu del grup en una d'aquestes cel·les, l'algoritme de cerca de camins s'expandirà fins a les 1500 cel·les visitades i retornarà un camí incomplert, ja que resulta impossible saber si aquella zona està aïllada sense mirar totes les cel·les visitables. Un cop l'individu arribi al final del camí tornarà a cercar el camí i es trobarà en la mateixa situació, i així successivament, de manera que es dedicarà a desplaçar-se per la zona més propera de forma indefinida fins que se li assigni un nou objectiu.

Una situació semblant es pot produir si un individu ha de recórrer 1500 cel·les per tal de rodejar un obstacle abans d'arribar a l'objectiu, però aquest cas és francament difícil si no es treballa amb escenaris molt grans i amb obstacles també molt grans.

7.3.3 - Limitacions de la composició dels grups

Aquesta aplicació ha estat pensada per tal de simular grups relativament reduïts d'individus (no més de 10 individus). Però tot i això, no hi ha cap limitació tècnica que fixi la màxima capacitat d'un grup i per tant es poden crear grups de la mida que es vulgui.

Però el comportament d'aquests macro-grups amb desenes d'individus no és el mateix que el que podria tenir un grup de fins a 10 o 12 individus. Per sobre d'aquesta xifra, els grups tendeixen a reagrupar-se amb excessiva facilitat degut a que els individus situats en la perifèria del grup fan pujar el nivell de dispersió (en estar relativament lluny del centre del grup). A part d'aquest efecte, els grups densament massificats també pateixen d'un cert comportament excessivament rígid. Aquesta rigidesa és deguda a que la força de cohesió és lineal i independent de la mida del grup, de manera que els individus situats a la perifèria reben una força de cohesió relativament gran, el que provoca que oprimeixin als individus del centre, i el grup, conseqüentment, es compacta. Aquesta compactació es nota en grups de 8 o 10 individus, però no és fins a arribar a grups de més d'una vintena d'individus quan es converteix en un problema i genera un comportament irregular dels individus.

Així doncs, es pot dir que aquest model funciona de forma òptima amb grups de fins a 12 individus. De forma irregular amb grups de fins a 20 individus, i de forma incorrecta per a grups de més de 20 individus.

D'altra banda, aquesta limitació sembla lògica si ens fixem en la vida real. En la vida real, tal i com en aquest projecte, quan més gran és un grup, més costós i complex resulta moure tots els seus individus de forma coordinada.

7.3.4 - Limitacions de les propietats dels individus

Tot i que els individus són enterament configurables a través del fitxer d'escenari, hi ha algunes restriccions i limitacions que cal seguir per tal d'assegurar que el grup es desplaci segons el model elaborat en aquest projecte.

Primerament, tres de les propietats d'un individu (màxima força d'acceleració, de frenada i de gir) pateixen una normalització i una multiplicació per un escalar abans de ser

truncades. Aquesta normalització comporta que les 3 propietats tinguin un rang màxim, per sobre del qual perden validesa i es considera que no s'apliquen.

Aquests rangs són els descrits a la figura 46:

Força	Màxim valor
maxAccForce (acceleració)	2
maxBrakeForce (frenada)	2
maxTurnForce (gir)	0.2

Figura 46. Rang de les 3 limitacions de força d'un individu.

A part d'aquests rangs a tenir en compte, l'individu ha de complir certes restriccions en terme de velocitat màxima si se'l pretén fer moure per un escenari complex amb multitud de passadissos i portes estretes. Aquesta limitació és deguda a que el sistema de locomoció gira de la mateixa manera i magnitud independentment de la velocitat de l'individu, per tant un individu més ràpid requerirà major espai per girar que un individu més lent. Aquesta variació en l'espai requerit per girar fa que si un individu es desplaça massa ràpid, pugui tenir una trajectòria massa recta en passar pel costat d'un obstacle i hi xoqui. Per tal d'evitar aquesta alteració, els individus haurien de tenir velocitats màximes no superiors a 8-10. A partir d'aquests valors els individus comencen a tenir problemes per desplaçar-se en fer trajectòries corbes.

7.3.5 - L'estabilitat dels grup

L'ús de diferents forces sobre un individu té com a objectiu fer que aquest individu es desplaci seguint una trajectòria on aquestes forces s'anul·len per complet, a excepció de la component paral·lela al vector de direcció de l'individu. Quan succeeix això, l'individu es desplaça en línia recta i no pateix variacions en la trajectòria, fins que algun factor altera aquest equilibri i l'individu es veu obligat a girar.

Així doncs, per a que un grup es desplaci en línia recta, els seus individus han d'estar en una situació d'estabilitat on les forces que actuen sobre ells estiguin en equilibri i només afectin a la tracció i no al gir. D'aquesta manera, els individus no faran moviments estranys i per tant la trajectòria del grup serà suau.

Per tal d'assegurar aquesta estabilitat, cal ponderar degudament totes les forces possibles i fer que la seva influència creixi i decreixi progressivament, de manera que no s'apliquin canvis bruscos en la suma de forces que rep un individu.

Per altra banda, cal ser molt cautelosos amb les funcions que determinen les magnituds de les forces depenent de les distàncies (força de repulsió amb parets, separació entre individus i cohesió) ja que la combinació d'aquestes 3 forces determina el comportament dels individus. Així doncs, una força de cohesió massa forta faria que els individus obstaculitzessin els seus companys de grups, en tractar tots d'apropar-se al mateix punt. Una força de cohesió massa dèbil permetria que la força de separació entre individus descohesionés el grup. Una força de repulsió amb les parets massa gran faria que els individus no es poguessin apropar als obstacles i podrien arribar a no ser capaços de passar per una porta, mentre que una massa dèbil provocaria constants xocs amb els obstacles.

Per tots aquests exemples exposats i per moltes altres situacions, l'elecció de les fórmules que permeten calcular la magnitud de les forces ha de ser meticulosament estudiada, i de fet és un dels punts més delicats de la implementació. D'aquestes magnituds depèn que els

individus puguin viatjar de forma estable, i conseqüentment que el grup viatgi seguint una trajectòria suau i realista.

De fet, la limitació que els grups no tinguin més d'una dotzena d'individus es pot interpretar de manera que un grup tant nombrós provoca una alteració a la magnitud de la força de cohesió que fa perdre l'estabilitat del moviment als individus i, per tant, el desplaçament del grup se'n resenteix.

7.3.6 - L'homogeneïtat en els grups

A diferència del que podria semblar, la no homogeneïtat dels individus que formen un grup no resulta un aspecte que alteri en excés el comportament d'un grup.

Evidentment, si un individu està limitat a una velocitat, el grup no podrà desplaçar-se a major velocitat, de manera que els individus més ràpids l'avancen i es desplacen a la seva mateixa velocitat, una mica per davant del centre del grup. Aquest comportament fa que un grup es desplaci a la velocitat de l'individu més lent, de manera que un grup amb 19 individus ràpids i 1 de lent, tindrà un comportament similar al d'un grup on tots siguin lents. Evidentment que quan un individu s'escapa o s'està reagrupant no té en consideració les propietats dels seus companys, així que sí que hi haurà variacions substancials en alguns moments. De la mateixa manera que la distribució dels individus serà molt més uniforme si les propietats dels individus són semblants.

En les proves efectuades s'han configurat grups on alguns individus quintuplicaven la velocitat màxima i l'acceleració dels individus més lents. Tot i això, el model aplicat ha estat capaç de simular un comportament cohesionat d'aquests grups no homogenis, sense que patissin un exagerat estirament que obligués al grup a reagrupar-se

7.4 – Ampliacions possibles

Un cop elaborat el projecte, una de les primeres ampliacions possibles, per tal de poder simular situacions més reals seria permetre que els escenaris abandonessin la seva forma plana i permetessin ser generats a partir de mapes d'alçada, el qual ampliaria notablement les situacions simulables per l'aplicació. Aquesta capacitat a la vegada comportaria la capacitat de l'aplicació de convertir l'energia cinètica en potencial i a l'inrevés. De manera que quan un individu estigués pujant una rampa, perdés progressivament velocitat i quan la baixés, augmentés la velocitat.

També s'hauria de recalcular constantment la normal (vector perpendicular al pla) de l'objecte que realitza el moviment per tal de poder usar els sistemes actuals de rotació dels individus.

També es podria considerar enriquir el comportament dels enemics, de manera que fossin capaços de perseguir els individus quan estiguessin a prop, simulant un depredador que intenta caçar una víctima potencial. Aquesta ampliació requeriria una modificació en els estats dels individus, afegint varis estats possibles als individus i als grups, de manera que els enemics en comptes de tenir un únic estat possible (*intimidar*), tinguessin com a mínim un altre estat encarregat d'implementar el comportament d'un depredador perseguint a un individu.

Gràcies a la utilització d'un escenari dividit en cel·les, una possible ampliació podria ser la de simular diferents tipus de terrenys, alguns més fàcils de travessar que d'altres. Emulant així situacions on un individu requereix més esforç per a travessar cert tipus de terrenys (per exemple, amb vegetació alta). Per a realitzar aquesta ampliació només caldria afegir diferents costos a les cel·les, de manera que a l'executar-se l'algoritme de cerca de camins ho tingués en consideració per tal de calcular el cost total del camí. També s'hauria de modificar el sistema de locomoció per a que ho tingues en compte a l'hora d'efectuar el moviment, alentint la marxa en cas d'estar travessant una cel·la amb un elevat cost.

Per altra banda, un dels punts més delicats i complexos de l'aplicació és el carregat d'escenaris en format de text '*.dgs*'. Una possible millora d'aquesta part seria convertir els arxius de text en arxius en format *XML*, un estàndard àmpliament estès. La utilització d'aquest format comportaria integrar un *parser* lector de XML a l'aplicació, que seria molt més fiable ja que detectaria possibles males configuracions dels arxius d'escenaris.

Una altra ampliació molt interessant seria facilitar una aplicació auxiliar que oferís una interfície agradable per generar escenaris sense haver de tenir coneixements sobre l'aplicació i les possibles configuracions dels arxius d'escenari. Aquesta petita aplicació permetria primerament escollir la mida de l'escenari i després anar canviant els tipus de cel·les per tal de generar l'escenari desitjat. A més, també hauria de permetre afegir i eliminar individus i grups en qualsevol part de l'escenari, d'igual manera que seria possible afegir enemics.

Finalment una ampliació força ambiciosa seria utilitzar totes les funcionalitats implementades per generar un videojoc real. Això pot sonar gairebé impossible si es té en ment un videojoc actual, elaborat per desenes de persones i amb pressupostos astronòmics. Però un videojoc senzill on l'objectiu final fos simplement dominar un escenari a base d'intimidar els individus enemics a partir de la presència dels nostres propis individus no resulta impensable. Aquest videojoc requeriria una ampliació dels estats possibles dels individus i la generació i eliminació d'individus en temps d'execució, de manera que l'escenari tendís a tenir un equilibri entre els 2 (o més) bàndols que els diferents jugadors intentarien trencar per tal de guanyar la partida.

8. Annexos

8.1 - Pseudocodi de l'A*

L'algoritme de cerca de camins de l'A* es basa en el següent pseudocodi:

```
// Initialization
OPENED := [Initial Node]
CLOSED := []                    // Empty list

f'(INICIAL) := h'(INICIAL)
do
   if  OPENED.IsEmpty() then FAILED

   else
      get BESTNODE from OPENED (with less f')
      move BESTNODE from OPENED to CLOSED
      if BESTNODE is TARGETNODE
         SOLUTION_FOUND := TRUE
      else
         get NEIGHBOURS of BESTNODE
         for each  NEIGHBOUR do NEXT_NODE()
while !SOLUTION_FOUND or FAILED
```

El qual fa ús del següent mètode per tractar els nodes successors del node avaluat:

```
NEXT_NODE()

NODE.PARENT := BEST_NODE
g(NODE):= g(BEST_NODE) + cost(BEST_NODE -> NODE) // Path cost

case NODE = OLDNODE in CLOSED
      if g(NODE) < g(OLDNODE)
            OLDNODE = NODE;

case NODE = OLDNODE in OPENED
      if g(NODE) < g(OLDNODE)
            OLDNODE = NODE;

case NODE is not in OPENED or CLOSED
      put NODE in OPENED
      f(NODE) := g(NODE) + h'(NODE)
```

8.2 - Pseudocodi del Diagonal Shortcut

Per calcular l'heurística mentre es propaga l'algoritme de l'A* s'utilitza el mètode de la drecera diagonal (*Diagonal Shortcut*), el pseudocodi del qual és el següent:

```
xDistance = abs(currentX-targetX)
yDistance = abs(currentY-targetY)
if xDistance > yDistance
     H=DIAGONAL_COST*yDistance+PERPENDICULAR_COST*(xDistance-yDistance)
else
     H=DIAGONAL_COST*xDistance+PERPENDICULAR_COST *(yDistance-xDistance)
end if
```

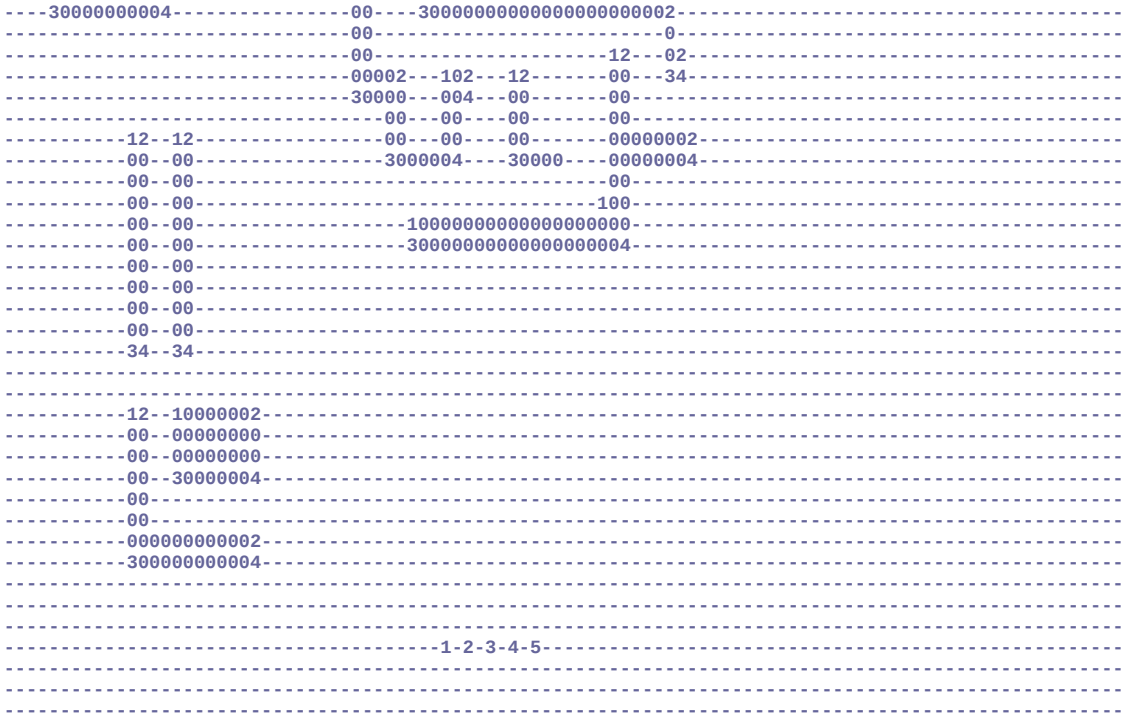
8.3 - Exemple d'un fitxer d'escenari

Seguidament s'adjunta un exemple d'escenari contingut en un arxiu '*.dgs*'. Aquest escenari en concret conté 2 grups de 6 individus cadascun, 2 grups de 2 i 3 enemics respectivament i un escenari de 100 cel·les d'amplada per 50 d'alçada.

```
n_groups: 2

group_nCharacters: 6

character_maxSpeed: 5.000
character_maxAccForce: 1.000
character_maxBrakeForce: 3.000
character_maxTurnForce: 0.1000
character_maxFear: 15.000
character_position: 20.500 -10.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 10.000
character_maxAccForce: 2.000
character_maxBrakeForce: 4.000
character_maxTurnForce: 0.1000
character_maxFear: 15.000
character_position: 23.500 -10.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 8.000
character_maxAccForce: 3.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.2000
character_maxFear: 15.000
character_position: 26.500 -10.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 6.000
character_maxAccForce: 1.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.2000
character_maxFear: 15.000
```

```
character_position: 29.500 -10.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 10.000
character_maxAccForce: 2.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.4000
character_maxFear: 15.000
character_position: 32.500 -10.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 6.000
character_maxAccForce: 3.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.1000
character_maxFear: 15.000
character_position: 35.500 -10.500 0.000
character_forward: 0.000 1.000 0.000

group_nCharacters: 6

character_maxSpeed: 5.000
character_maxAccForce: 1.000
character_maxBrakeForce: 3.000
character_maxTurnForce: 0.1000
character_maxFear: 15.000
character_position: 20.500 -100.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 10.000
character_maxAccForce: 2.000
character_maxBrakeForce: 4.000
character_maxTurnForce: 0.1000
character_maxFear: 18.000
character_position: 23.500 -100.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 8.000
character_maxAccForce: 2.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.2000
character_maxFear: 15.000
character_position: 26.500 -100.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 6.000
character_maxAccForce: 1.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.2000
character_maxFear: 15.000
character_position: 29.500 -100.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 15.000
character_maxAccForce: 2.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.4000
character_maxFear: 15.000
character_position: 32.500 -100.500 0.000
character_forward: 0.000 1.000 0.000

character_maxSpeed: 8.000
```

```
character_maxAccForce: 5.000
character_maxBrakeForce: 5.000
character_maxTurnForce: 0.1000
character_maxFear: 15.000
character_position: 35.500 -100.500 0.000
character_forward: 0.000 1.000 0.000

n_enemy_groups: 2

group_nEnemies: 2

enemy_maxSpeed: 4.500
enemy_maxAccForce: 1.800
enemy_maxBrakeForce: 3.000
enemy_maxTurnForce: 0.1000
enemy_position: 5.500 -10.500 0.000
enemy_forward: 0.000 1.000 0.000

enemy_maxSpeed: 5.000
enemy_maxAccForce: 2.000
enemy_maxBrakeForce: 3.000
enemy_maxTurnForce: 0.1000
enemy_position: 2.500 -10.500 0.000
enemy_forward: 0.000 1.000 0.000

group_nEnemies: 3

enemy_maxSpeed: 4.500
enemy_maxAccForce: 1.800
enemy_maxBrakeForce: 3.000
enemy_maxTurnForce: 0.1000
enemy_position: 5.500 -80.500 0.000
enemy_forward: 0.000 1.000 0.000

enemy_maxSpeed: 5.000
enemy_maxAccForce: 2.000
enemy_maxBrakeForce: 3.000
enemy_maxTurnForce: 0.1000
enemy_position: 2.500 -80.500 0.000
enemy_forward: 0.000 1.000 0.000

enemy_maxSpeed: 5.000
enemy_maxAccForce: 2.000
enemy_maxBrakeForce: 3.000
enemy_maxTurnForce: 0.1000
enemy_position: 2.500 -84.500 0.000
enemy_forward: 0.000 1.000 0.000

scene_width: 100
scene_height: 50
----------------------------------------------------------------3000000000000004----------------
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
00000000002---10000000000000000002----------------------------10000000000000002----------------
00000000004---00000000000000000004----------------------------3000000000000004----------------
-------------0------------------------------------------------------------------------------------
-------------0-------------------------------------12-----------------------------------------
----10000000000-----------------------------------100-----------------------------------------
----30000000000-----------------------------------1004----------------------------------------
-------------0-----------------------------------304-----------------------------------------
-------------0------------------------------------------------------------------------------------
00000000002---0------------------------------------------------------------------------------------
00000000004---0------------------------------------------------------------------------------------
-------------0------------------------------------------------------------------------------------
-------------0------------------------------------------------------------------------------------
----10000000000---------------12----12-------------------------------------------------------------
```


1-2-3-4-5

9. Bibliografia

[1] Craig W. Reynolds. *Steering Behaviors For Autonomous Characters*. Game Developers Conference, San Jose, CA, March 1999.

[2] Craig W. Reynolds. *Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics*. 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.

[3] Bruce Blumberg & Tinsley A. Galyean. *Multi-level direction of autonomous creatures for real-time virtual environments.* Proceedings of SIGGraph, 1995.

[4] Richard S. Wright Jr. & Michael Sweetl. *OpenGL superbible*. Waite Group Press, 1996.

[5] Youssef Saab & Michael VanPutte. *Shortest Path Planning on Topographical Maps*. IEEE Transactions on systems, man, and cybernetics – PartA: System and humans, Vol. 29, No. 1, January 1999.

[6] G. F. Luger. *Artificial Intelligence, Strategies and Structured for Complex Problem Solving*. New York: Benjamin Cummings, 1993.

[7] J. S. B. Mitchell. *An algorithm approach to some problems in terrain Navigation.* Artif. Intell., vol. 37, pp. 171–201, December 1988.

[8] K. Fujimura. *Motion Planning in Dynamic Environments.* New York, Springer-Verlag, 1991.

[9] K. Fan & L. Po-Chang. *Solving find-path problem in mapped environment using modified A_ algorithm*. IEEE Trans. Syst., Man, Cybern., vol. 24, pp. 1390–1397, Sept. 1994.

[10] J. A. Storer. *Shortest paths in the plane with polygonal obstacles*. J. ACM, vol. 41, pp. 982–1012, 1994.

[11] R. Kimmel & N. Kiryati. *Finding shortest paths on surfaces by fast global approximation and precise local refinement*. SPIE Vision Geometry III, vol. 2356, pp. 198–209, 1994.

[12] Roger S. Pressman. *Ingeniería del Software. Un enfoque práctico*. Ed. MacGraw Hill/Interamericana de España, 5º edición. Madrid 2002

[13] Armin Bruderlin & Tom Calvert. *Goal-Directed, Dynamic Animation of Human Walking*. ACM SIGGRAPH'89, Proceedings, vol. 23, pp 233-242. 1989

[14] Dave Pottinger. *"Coordinated Unit Movement" and "Implementing Coordinated Movement"*, Game Developer Magazine, January, 1999.

[15] David J. Kruglinski. *Programación Avanzada con Microsoft Visual C++*. Mc Graw Hill, 1998.

[16] Tom Archer & Andrew Whitechapel. *La biblia de Microsoft Visual C++ .Net.* Anaya Multimedia, 2003.

[17] Alan Watt & Fabio Policarpo. *3D games, animation and advanced real-time rendering (volume 2)*. Pearson Education, 2003.

[18] Donald Hearn & M. Pauline Baker. *Gràficos por computadora con OpenGL*. Pearson Educación, 2006.