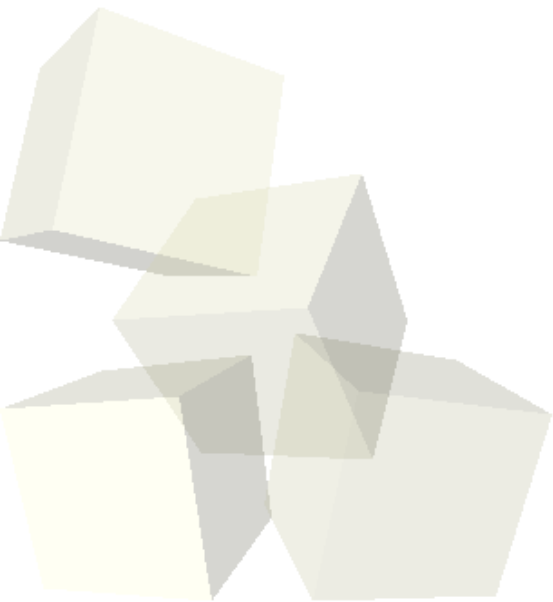




Multi-Agent Systems e Videogiochi

Fabio Marchesi
Sistemi multi-agente
a.a. 2007/2008





Motivi dell' indagine

I videogame sono un campo di applicazione molto interessante per i lavori di ricerca

- Sono simulazioni piuttosto fedeli dei vari problemi
- Hanno vincoli che richiedono molta efficienza (stress test della tecnica)
- Sono immediatamente comprensibili

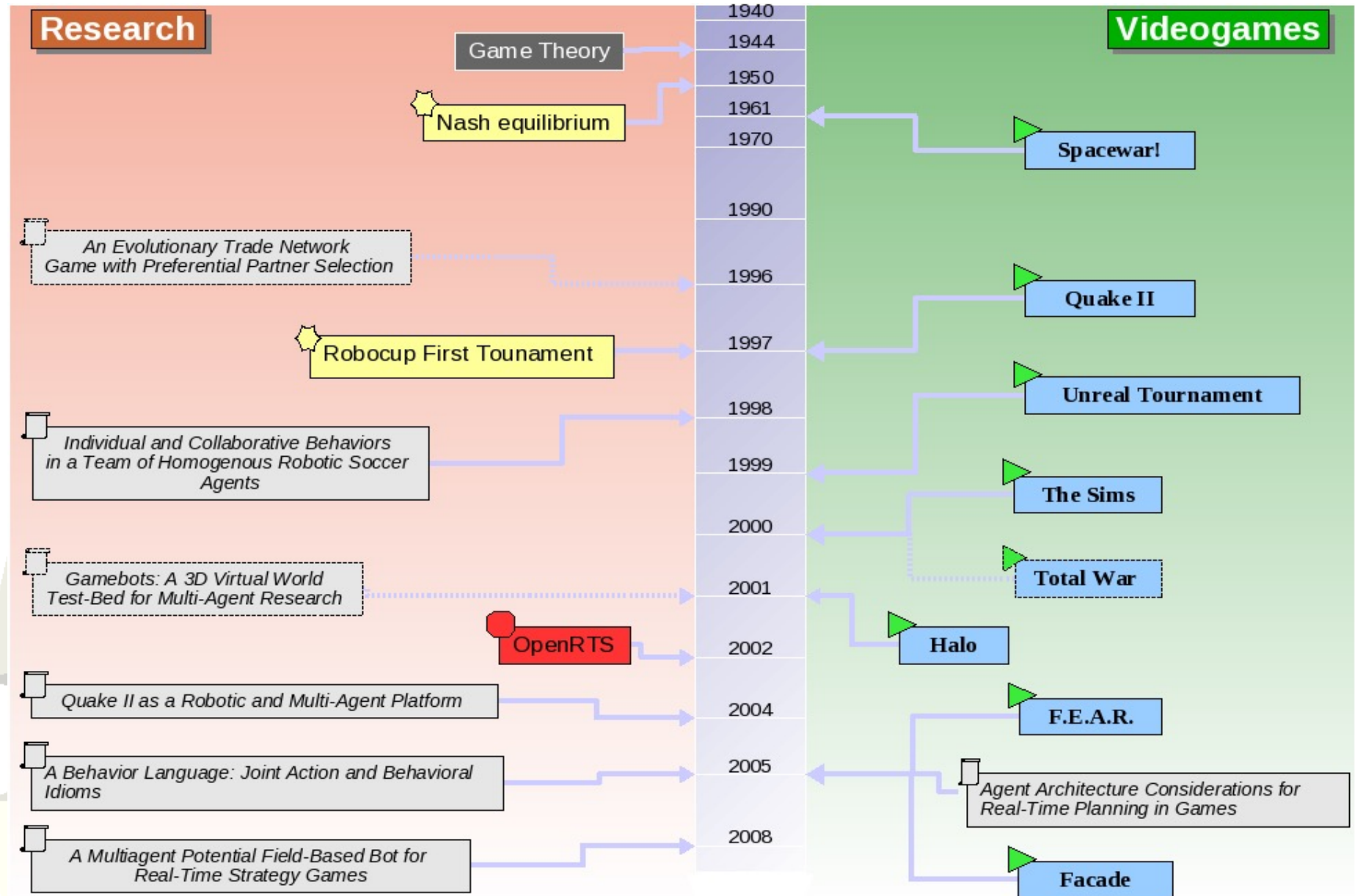
...

- *Sono un mercato formidabile*





Timeline





I MAS nella ricerca

Nella ricerca i Multi Agent Systems hanno una storia piuttosto lunga.

- La teorizzazione risale agli anni 40

Vengono sfruttati per modellare sistemi complessi difficilmente descrivibili nell'insieme ma suddivisibili in sub-unità operative più o meno indipendenti/omogenee.

Dato uno scopo generale del MAS (risolvere un problema, eseguire una funzione), gli agenti perseguono degli scopi più specifici, solitamente senza la conoscenza diretta della visione complessiva, attuando una serie di comportamenti individuali.

Come dare una base teorica a questi tipi di sistemi?

- Un approccio diffuso è la Game Theory.





Game theory

La **Game Theory**, nascendo nella matematica applicata, è stata adottata in varie branche della scienza, per modellare sistemi in cui vari giocatori sono coinvolti in situazioni strategiche che tendono ad un determinato punto di equilibrio.

Applicazione multidisciplinare: economia, psicologia, sociologia, politica, biologia, informatica (MAS).

I giochi possono avere queste caratteristiche:

- **Cooperatività**
- **Zero-sum**
ad una vincita corrisponde una perdita
- **Simmetricità**
una mossa ha un esito indipendente da chi la fa
- **Sequenzialità**
- **Perfect Information**
tutte le mosse precedenti di tutti sono conosciute





Trade Network Game

Citazione

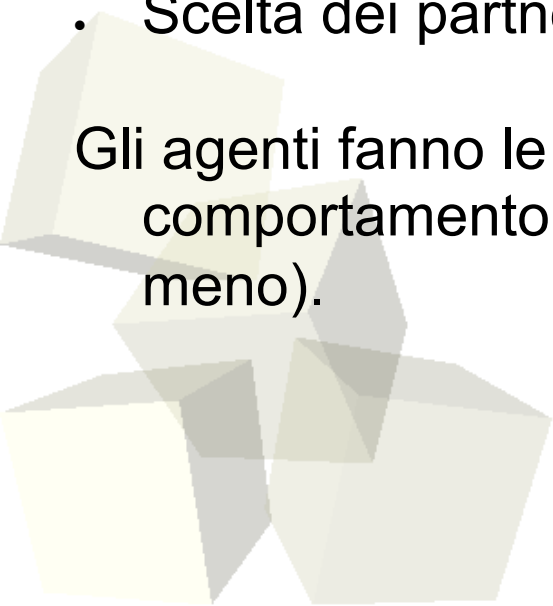
“An Evolutionary Trade Network Game With Preferential Partner Selection”, Leigh Tesfatsion, 1996

TNG è un game in cui i partecipanti sono venditori e compratori che cercano di massimizzare i propri profitti.

Il paper descrive un TNG che prevede i caratteri di:

- Evoluzione delle dinamiche
- Scelta dei partner in base alle preferenze

Gli agenti fanno le proprie scelte sugli accordi da fare in base al comportamento passato degli altri agenti (se hanno cooperato o meno).





Soccer Agents

I giochi complessi, come il calcio, sono terreno di sfida per sviluppo in robotica e per i MAS.

Robotica:

- controllare i movimenti -> equilibrio
- percepire ambiente (e attraversarlo senza problemi) -> visione, collision avoidance
- eseguire azioni complesse (calciare, parare)

MAS:

- modellare gli altri (avversari, compagni)
- usare behaviour adattivi all'ambiente e alle azioni degli altri
- teamwork con comunicazione diretta o indiretta (es. blackboard)

Robocup

- Creata nel '93 come showcase della ricerca con un valore aggiunto mediatico (il calcio è molto diffuso).
- Nel 97 il primo torneo tra team di diversi paesi.





Robocup: CMUnited-97

Riferimento

"Individual and Collaborative Behaviors in a Team of Homogeneous Robotic Soccer Agents", M. Veloso e P. Stone, 1998

Idea:

Presentare la struttura del team CMUnited-97 che ha vinto nell'edizione del 1997 del Robocup.

Il sistema è un MAS che controlla un team di giocatori-robot.

Il team è composto da 5 robot ognuno governato da un agente in grado di percepire il mondo e attuare azioni in base a

- i propri goal
- la strategia di squadra.

Gli agenti sono omogenei e non comunicano direttamente tra loro, ma condividono la conoscenza degli obiettivi di squadra.



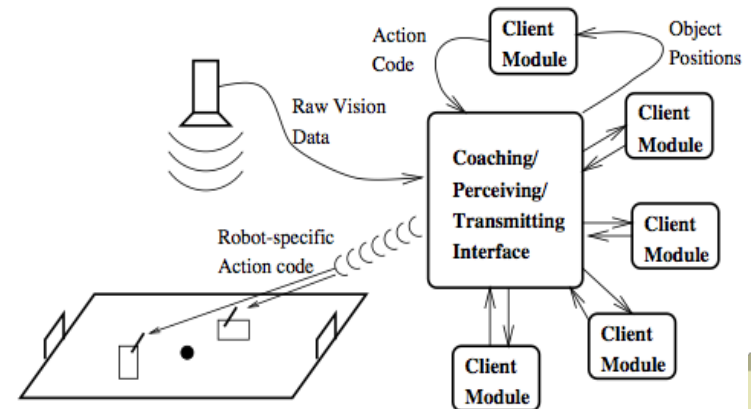


CMUnited-97: sistema

Il sistema comprende gli agenti e un'unità centralizzata (coach) che colleziona percezioni sul mondo (campo, team e avversari) con tecniche di computer vision e le ridistribuisce.

Inoltre permette di ricalcolare i **locker-room agreements** (strategie di squadra) quando avvengono certe condizioni (trigger).

Queste vengono poi, propagate agli agenti durante periodiche fasi di **team-synchronization**.





CMUnited-97: l'Agente

L'agente è composto dalle seguenti entità:

- **Locker-room Agreement**

Aggiornato durante le re-sincronizzazioni periodiche di squadra. Indicano i compiti dell'agente nella squadra.

- **World State**

Percezioni ed effetti previsti degli External Behaviour.

- **Internal State**

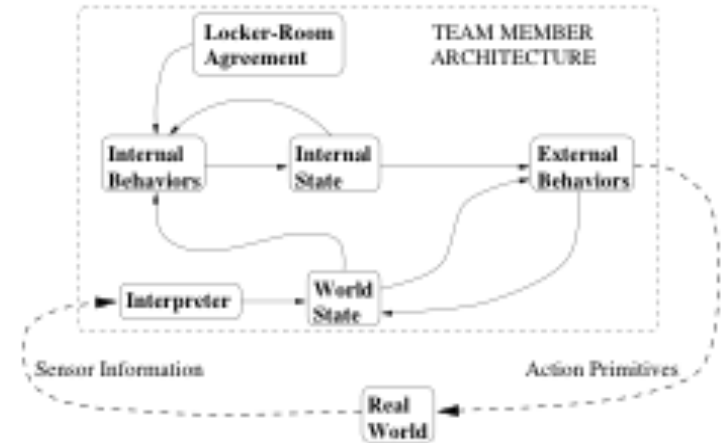
Variabili interne: es. il ruolo corrente, i record dei World State passati, etc.

- **Internal Behaviour**

La politica di aggiornamento dell' Internal State.

- **External Behaviour**

La strategia usata per agire nell'ambiente.





CMUnited-97: Teamwork

Il gioco di squadra è ottenuto definendo i concetti di:

■ Ruolo

- ◆ definisce la zona di competenza dell'agente
- ◆ definisce i behaviour che deve attuare

■ Formazione

- ◆ comprende un insieme (cardinalità uguale al numero di agenti) di ruoli
- ◆ Varia dinamicamente in base ai locker-room agreement. Es. si è in svantaggio quindi si passa a due attaccanti.

■ Active Modes

- ◆ *Attivo*
L'agente ha la palla
- ◆ *Ausiliario*
L'agente deve collaborare all'azione (es. ricevere il passaggio)
- ◆ *Inattivo*

Un agente sa ricoprire ogni ruolo (a parte il portiere che ha HW specializzato) e può cambiarlo dinamicamente per adattarsi alla situazione.

Il sistema è in grado di fare giocate collaborative come il passaggio.

Step 1 dovuto al *Locker-room agreement*:

- Il robot in possesso della palla diventa *active*.
- Il compagno più vicino diventa *auxiliary*.

Step 2 dovuto all' Internal Behaviour del robot *active*:

- Il robot con la palla decide se tirare o effettuare un passaggio (reasoning con funzione di valutazione della riuscita).

Step 3 dovuto all'Internal Behaviour del robot *auxiliary*:

- Il robot *auxiliary* sceglie come Internal Behaviour l'intercettazione della palla (per ricevere il passaggio).

Step 4 dovuto all'External Behaviour del robot *active*:

- Il robot esegue le azioni per passare/tirare la palla.



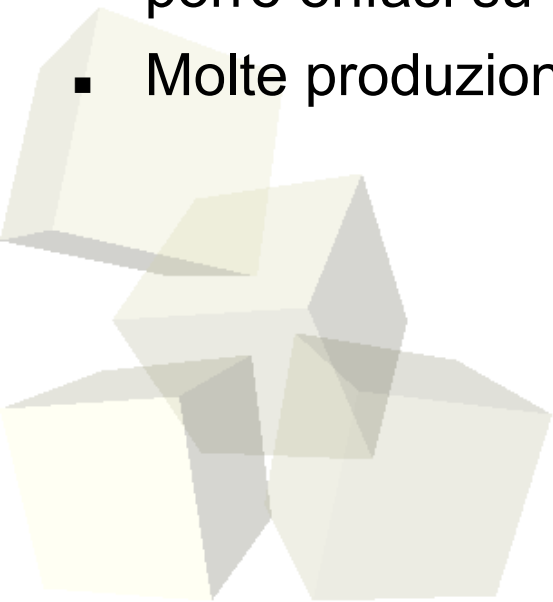
*“Spacewar!”, del 1961, è considerato da molti il primo videogioco.
Fu programmato su un DEC PDP-1 nei laboratori del MIT.*

Che scopo ha il videogioco?

→ Divertire.

Come si ottiene un gioco divertente?

- Non c'è una formula magica: diversi tipi di giochi richiedono di porre enfasi su di un aspetto piuttosto che su di un altro.
- Molte produzioni (milionarie) hanno fallito nel divertire...





Rapporto con la ricerca

Gli sviluppatori guardano alla ricerca per trovare soluzioni innovative ai problemi o per introdurre caratteristiche avanzate

- Di **gameplay**

es. Motion detection (Wii, Playstation Eye, Eye of Judgement,...)

- Di **presentazione**

es. Grafica 3D (Lightening HDR, Shader,...), Animazione (IK, Morphing,...)

Ma i vincoli sono più stretti rispetto a quelli della ricerca:

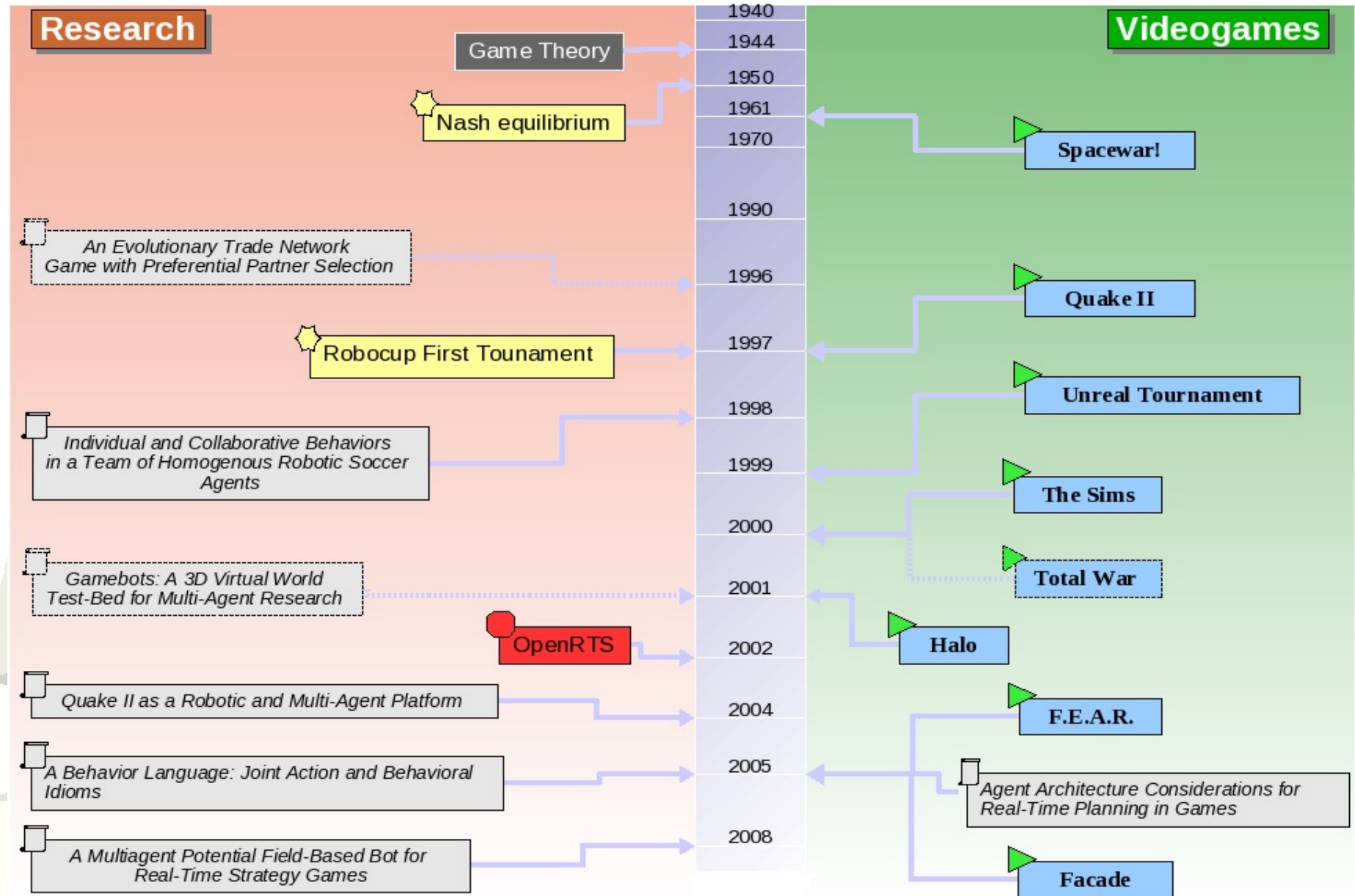
- Capacità computazionali limitate
- Dinamica Real-Time (molto spesso)

Devono quindi usare l'idea contestualizzandola e ottimizzando al massimo l'implementazione.

- La soluzione della ricerca tende ad essere **general purpose**.
- La soluzione videoludica è **context-oriented**.



Timeline



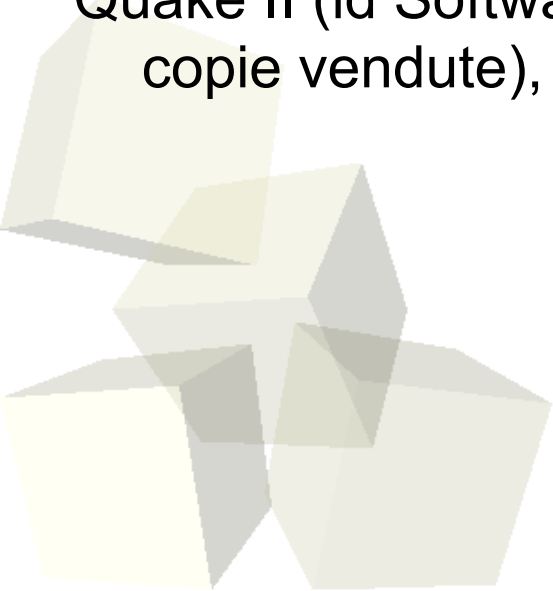


Videogames: MAS

Appena fu disponibile più potenza computazionale, la tendenza fu di rendere l'esperienza del giocatore più immersiva dando profondità alle capacità dei NPC (o avversari).

Dopo anni di AI centralizzate (realizzate con script e FSA), dalla metà degli anni 90 si cominciò a pensare di avere tante semplici AI indipendenti in competizione con il giocatore.

Il concetto che naque, fu quello del BOT e fu reso famoso da Quake II (id Software, 1997) che, con un successo enorme (milioni di copie vendute), aprì la strada allo sviluppo in questo senso.



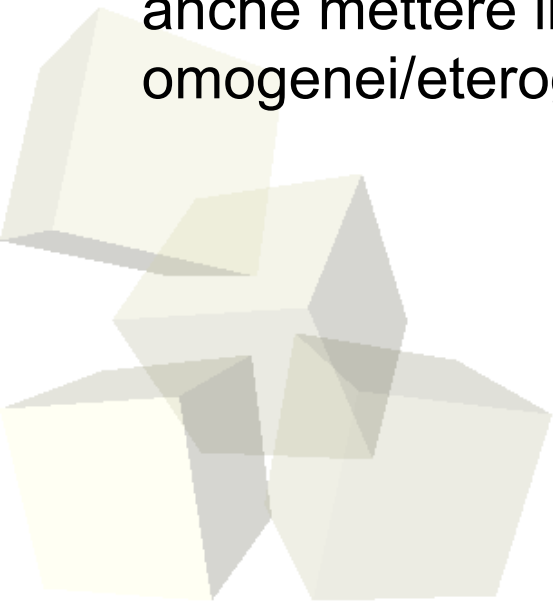


Videogames: BOT

Il concetto dei BOT presenta diverse caratteristiche che lo accomunano agli agenti:

- Capacità percettive
- Uno o più behaviour a seconda della situazione
- Uno scopo proprio (sconfiggere/uccidere il giocatore)

Il videogioco comprendente BOT, quindi, diventava più o meno dichiaratamente un MAS con la funzione primaria di divertire (ma anche mettere in difficoltà) e composto da agenti (i BOT) omogenei/eterogenei senza capacità di comunicazione.





Videogames: Tool di ricerca

Grazie al successo e al conseguente lavoro di raffinazione del concetto, i giochi a BOT divennero un perfetto ambiente di sperimentazione anche per la ricerca avendo:

- Un ambiente virtuale pronto all'uso (per la presentazione)
- Il supporto per diversi tipi di percezioni (suono, vicinanza, vista)
- Il supporto per la distribuzione in rete
- Spesso tool di sviluppo specifici per i BOT

Esempio:

- **Quake II 1997**

“Quake II as a Robotic and Multi-Agent Platform”, Chris Brown, Peter Barnum, Dave Costello, George Ferguson, Bo Hu, Mike Van Wie, 2004

Uso del gioco per didattica/ricerca su tecniche di computer vision, coordinazione, MAS, etc.

- **Unreal Tournament 1999**

“*Gamebots: A 3D Virtual World Test-Bed For Multi-Agent*”, Research Rogelio Adobbati, Andrew N. Marshall, Gal Kaminka, Steven Schaffer, Chris Sollitto Andrew Scholer, Sheila Tejada, 2001

Ambiente specializzato di sperimentazione di tecniche MAS basato su UT.





Real Time Strategies: MAS

Nel campo degli RTS, i MAS sono stati considerati ma per molto tempo il loro uso è stato proibito a causa dei limiti computazionali:

- In questi giochi il numero di agenti sarebbe troppo alto

D'altronde, alcune tecniche dei MAS sono state comunque applicate in determinati giochi

Es. la serie Total War prevede un simil-agente per ogni unità militare in grado di fare limitate percezioni ed adottare alcuni behaviour diversi.

La ricerca, d'altronde, ha considerato lo scenario RTS molto promettente per la ricerca sui MAS per via della sua simulazione di molti aspetti dei casi reali (correlazione anche con la robotica).



Ambiente di sviluppo per RTS.

<http://www.cs.ualberta.ca/~mburo/orts/>

- Iniziato nel 2002 da Michael Buro (University of Alberta)
- Pensato per lo sviluppo di giochi e per la ricerca su AI.
- Architettura a client-server
 - ◆ i client possono essere totalmente eterogenei
 - controllati da umani
 - controllati da AI (es. tornei di bot)
 - ◆ l'AI non può “barare”
- GPL, sia il source code che l'artwork
- Platform independant
- Supporto per OpenGL e scripting (con linguaggio custom)





Potential fields e ORTS

Riferimento

“A Multiagent Potential Field-Based Bot for Real-Time Strategy Games”, **Johan Hagelback** and Stefan J. Johansson, 2008

Idea:

Sperimentare tecniche di potential fields nel campo degli strategici in tempo reale.
Uso di Open RTS come ambiente di sviluppo del MAS.
Partecipazione ai tornei (2007-2008) per verificare l'efficacia della soluzione contro altri bot.

Il bot è composto da un numero variabile di agenti con compiti specifici:

- **Unità**
agente semplice che agisce nel mondo virtuale
- **Agente coordinatore** (opzionale, legato ad uno scenario di test)
un agente che permette di implementare strategie di team
- **Agente di interfaccia**
Data la natura client-server di ORTS, questo agente comunica con il server per dare al bot (e ai suoi agenti interni) gli aggiornamenti sullo stato dell'ambiente.





I **potential fields** permettono di assegnare un valore di attrazione/repulsione alle varie locazioni dell'ambiente.

Per aggiungere significato, si usano diversi potential fields per i task dell'agente:

- **fields of navigation**

Permettono all'agente di evitare collisioni con i punti fissi del terreno.

- **strategic fields**

Definisce la posizione degli obiettivi correnti dell'agente.

- **tactical fields**

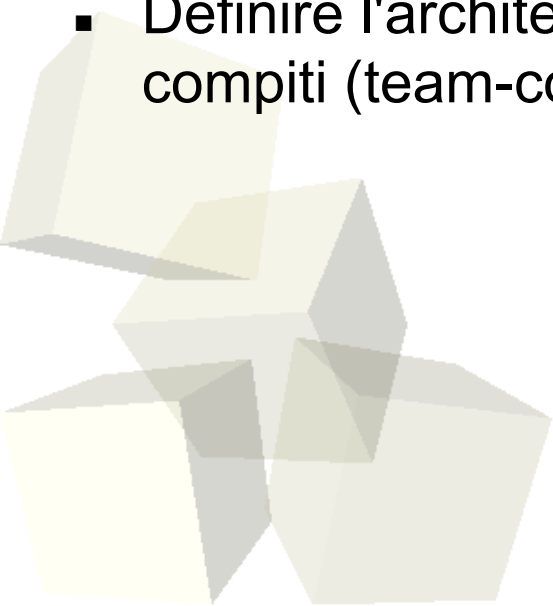
Rileva la posizione degli altri agenti per permettere cooperazione/coordinazione.





Per sviluppare il bot con potential fields si segue un metodo in 6 passi:

- Identificare gli oggetti (statici-dinamici, attrattivi-repulsivi, etc.)
- Identificare le forze, i potential fields
Un PF per ogni bisogno/task/compito.
- Assegnare le cariche agli oggetti nei PF.
- Decidere la risoluzione di tempo/spazio.
- Decidere quali oggetti diventano agenti e come sono strutturati.
- Definire l'architettura del MAS, aggiungendo altri agenti per altri compiti (team-coordination, etc.)





PF: Raccolta risorse

Scenario “Collaborative Pathfinding”.

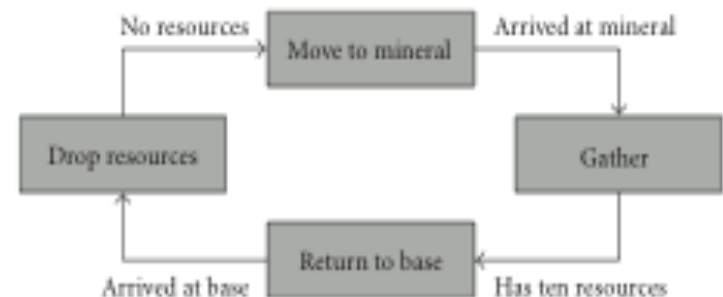
- 20 lavoratori
- n risorse in posizioni random
- k sheep (oggetti in movimento, per disturbo)
- 1 base di raccolta

Il bot prevede tre Potential Fields:

- field of navigation
Usato dai lavoratori per evitare punti della mappa non attraversabili (cliff).
Repulsivo.
- strategic field
Usato per trovare le risorse e la base. Attrattivo/repulsivo in base allo stato dell'agente.
- tactical field
Indica le cariche di altri lavoratori, sheeps e base.
Repulsivo per evitare collisioni.

L'agente è modellato come una semplice FSA.

I risultati ottenuti sono simili a quelli di un tradizionale approccio con A*.





PF: TankBattle

Scenario “TankBattle”

- battaglia di carri armati
(1 colpo poi cooldown per tempo t)
- n basi per fazione
- Sheep (oggetti-disturbo in movimento, indistruttibili)
- scopo distruggere tutte le basi



Il bot è un MAS fatto da 4 Potential Fields:

- field of navigation
- tactical field
- strategic field



Attrattivo. Evidenzia le unità nemiche e il range migliore da cui attaccare.

- defensive field

Repulsivo. Evidenzia le unità nemiche permettendo al tank manovre evasive durante il tempo di cooldown dell'arma.

Inoltre ha un agente coordinatore che permette di implementare strategie di team (es. coordinare una squadra di tank per cercare di abbattere più bersagli possibili invece di disperdere i colpi).

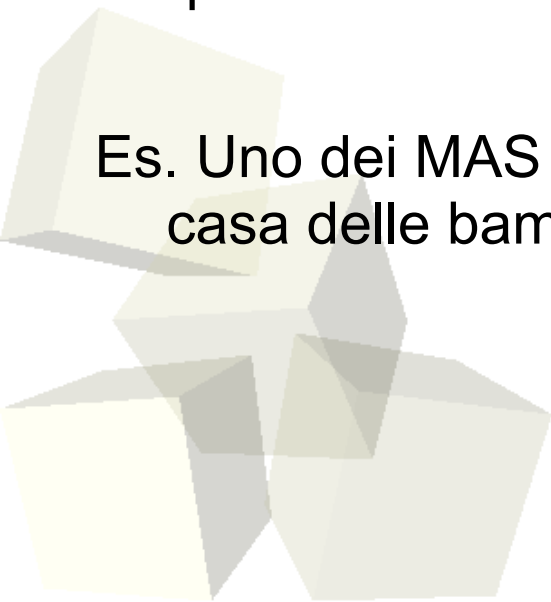


SIMULATION

Data la loro capacità di rendere sistemi complessi, i MAS sono stati considerati nei giochi di simulazione per rendere le situazioni di vita sociale.

Avendo bisogno di singoli agenti più complessi per aumentare la credibilità dei personaggi virtuali, però, in questi giochi si tende ad usare MAS con un numero basso di agenti anche se, grazie alla capacità computazionale in continua crescita, questo trend sta cambiando.

Es. Uno dei MAS più di successo nei videogiochi è stato una casa delle bambole virtuale: THE SIMS.





The Sims: People & Object

The Sims, 2000, prodotto da Maxis e creato da Will Wright.

- 100 milioni di copie vendute
- 1.6 miliardi di dollari
- 7 espansioni pubblicate

Idea:

I Sim sono agenti semplici capaci di provare pulsioni fisiche e sociali.
Il mondo è popolato di Smart Object che permettono

- Interazione
- Collaborazione con altri
- Soddisfazione dei desideri del Sim.

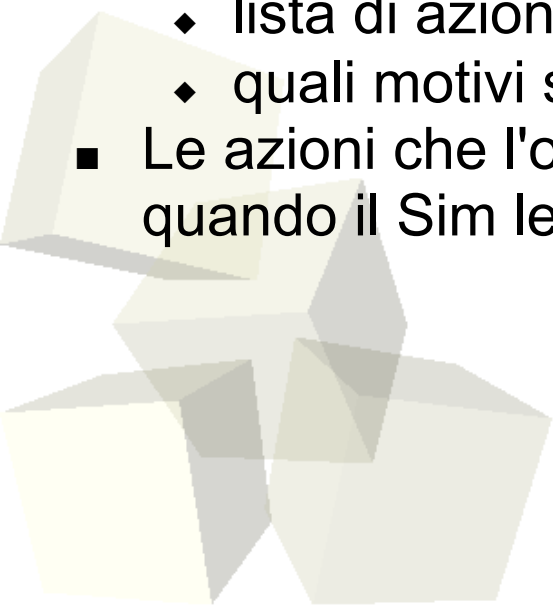
In effetti questo approccio ricorda molto la teoria del A&A.



The Sims: Smart Object

Gli Smart Object incorporano sia i dati che il controllo.

- Hanno un proprio micro-thread
- Hanno un interfaccia standard verso l'engine
 - ◆ funzione di *startup*
 - ◆ funzione di *update*
- Hanno un interfaccia standard che permette ai Sims di interagire con essi.
- Pubblicano automaticamente il loro “modo d'uso”.
 - ◆ lista di azioni disponibili
 - ◆ quali motivi soddisfano
- Le azioni che l'oggetto pubblica, sono script che vengono eseguiti quando il Sim le invoca.





The Sims: il Sim

E' un agente semplice che prova pulsioni di base:

- fisiche (fame, comfort, igiene, bisogni)
- sociali (energia, divertimento, socializzazione, atmosfera)

Ogni pulsione è caratterizzata da una funzione che lega il suo valore all'umore generico del Sim. Queste funzioni non sono lineari.

Planning

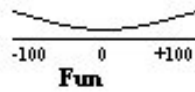
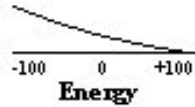
Con l'abbassarsi dell' umore, il Sim decide di eseguire un azione. In base alla pulsione predominante, il Sim cerca (attraverso una tecnica di Potential Fields) di interagire con l'oggetto più vicino che può soddisfarlo migliorando l'umore.



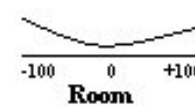
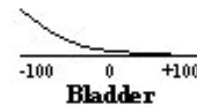
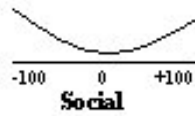
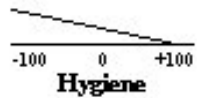


The Sims: pulsioni

Happiness Weights



= Mood (-100..+100)



Le pulsioni del sim generano un potential field attorno a lui in base al potere che hanno gli altri oggetti di soddisfarlo.

Le pulsioni hanno valori che variano in $[-100, 100]$.

Influiscono sull'umore del Sim in maniera non lineare.





Dynamic Narrative

Questa tecnica sta prendendo piede nei videogiochi (ma anche in altri media) per offrire al giocatore esperienze

- Coinvolgenti
- Sempre diverse

La narrazione viene influenzata dalla interazione dei player con le altre entità coinvolte.

In particolare questo richiede di programmare dei MAS con agenti molto complessi in grado di “recitare” la propria parte

Interazioni Complesse -> Agenti Complessi

Un risultato molto interessante lo ha ottenuto Facade nel 2005 nella costruzione di un dramma interattivo.

Si può anche ricordare il super agente The Director di Left For Dead (2008) ovvero un AI che manovra i mostri (agenti semplici) e l'ambiente come se fosse un regista con lo scopo dichiarato di mettere in difficoltà e spaventare il giocatore.



Publicato nel 2005 come open-source è frutto della ricerca di Micheal Mateas e di Andrew Stern nel campo della narrativa procedurale.

Il gioco consiste in un dramma ad atto singolo interattivo.

L'ambientazione è la casa di Trip e Grace, una coppia sposata di trentenni che rivela nel corso della storia seri problemi di coppia.

Il giocatore, attraverso una visuale in prima persona, può spostarsi nell'ambiente e manipolare gli oggetti (ma anche fare azione su Trip e Grace come abbracciare, confortare o baciare), inoltre può colloquiare con gli altri personaggi usando il linguaggio naturale (Inglese).

Lo svolgimento non è predeterminato, ma varia a seconda delle interazioni e delle scelte che il giocatore compie e delle conseguenti reazioni dei personaggi.



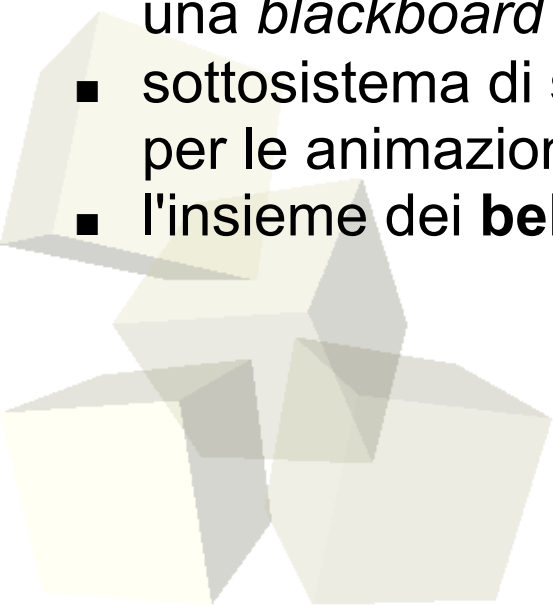


Gli agenti in Facade sono quattro:

- **Trip**
- **Grace**
- **Drama Manager**
- **L'avatar del Giocatore**
fa solo da tramite agli input (motorii e testuali) del giocatore

Trip e Grace sono modellati secondo la struttura:

- **working memory**
una *blackboard* dinamica e con fatti tipizzati
- sottosistema di **sensori** e **attuatori** fisici
per le animazioni, ricevere stimoli dall'ambiente, etc.
- l'insieme dei **behaviour**





Riferimento

“*A Behavior Language for Story-based Believable Agents*”, Michael Mateas
Andrew Stern, 2002

Idea:

ABL è il linguaggio utilizzato per programmare gli agenti in Facade.
E' un linguaggio compilato basato su Hap, ed è in grado di generare codice Java eseguibile.

Permette di descrivere il comportamento dell'agente specificando una serie di behaviour in grado di ottenere il soddisfacimento di un goal prefissato. I behaviour sono composti da step che vengono eseguiti sequenzialmente e che possono risultare in un successo o un fallimento: il fallimento di uno step porta al fallimento del relativo behaviour.



Facade: ABL Steps

I tipi di step base sono:

- **wait**

Permette di aspettare il verificarsi di una condizione (controllando la working memory dell'agente)

- **act**

Permette di comandare al sottosistema motorio dell'agente di eseguire un azione

- **subgoal**

Permette di specificare un sottogoal da soddisfare durante questo behaviour.

- **mental_act**

Esegue un azione puramente computazionale o astratta (coinvolgendo di solito solo la working memory)



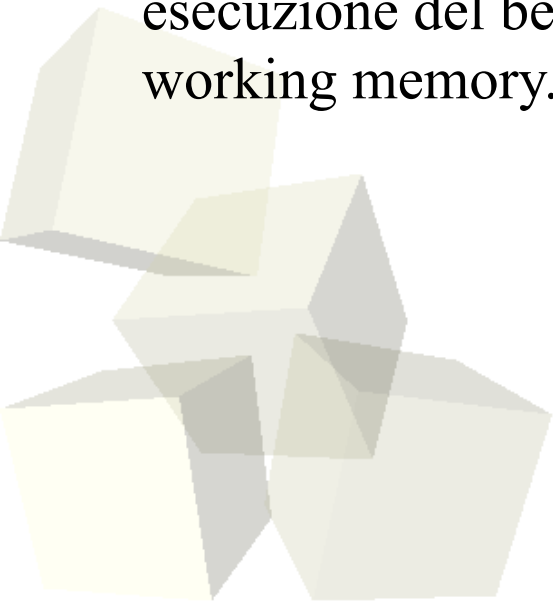


Facade: ABL Behaviours

I behaviour specificano delle precondizioni che devono essere verificate nella working memory per poterli selezionare.

Inoltre a parità di precondizioni, i behaviour vengono selezionati in base al loro valore di specificity (meccanismo di fall-back nel caso un behaviour fallisca).

I behaviour possono eseguire i propri steps in maniera sequenziale o parallela: per mantenere il controllo sull'esecuzione comunque l'albero di esecuzione del behaviour (ABT) viene mantenuto come fatto nella working memory.





Per permettere uno svolgimento drammatico e credibile della storia, gli agenti devono coordinarsi tra loro.

Gli agenti sono per natura fortemente autonomi, ma la narrazione ha bisogno di comportamenti coerenti alla storia.

Joint Behaviour

- Dichiarare un team di agenti che lo condivide.
- Esplicita la sincronizzazione dell'esecuzione dell'agente
 - ◆ nell'entrata/uscita dal behaviour
 - ◆ in uno o più step del behaviour
- E' un rilassamento del vincolo dell'autonomia dell'agente per ottenere un comportamento funzionale alla storia.



Facade: Drama Manager

Il **Drama Manager** si occupa di mantenere il filo narrativo.
Usa i **beat** ovvero unità narrative (simili a scene di un film).

Un **beat** è composto da goal e behaviour sia individuali che joint ed ha varie proprietà:

- precondizioni
- priorità
- livello di tensione
- effetti

Il **Drama Manager** sceglie (da un set prestabilito) il beat da attivare come se si muovesse su un grafo cercando il beat più prioritario, con le precondizioni verificate che abbia il livello di tensione giusto.

I behaviour e i goal associati al beat vengono assegnati agli agenti che ne iniziano “l'interpretazione”.



MAS usati per avere dei compagni artificiali credibili (agenti che collaborano in maniera efficace) ma anche avversari con tattiche complesse e behaviour coordinati.

Uno dei primi a mostrare un livello di tattica superiore fu **No One Lives Forever** (2000), ma il successo commerciale nel campo degli FPS tattici fu totalmente di **Halo** (2001).

Halo

- tattiche di gruppo basate su concetti di aree e team
- gli agenti (nemici) hanno behaviour tree
- reagiscono a situazioni esterne in maniera credibile
 - ◆ es. Cercano di evitare le bombe e se è il caso scappano.

Un passo avanti nella tecnica di realizzazione di agenti veramente competitivi è dovuto a F.E.A.R.



F.E.A.R. : agenti e planning

Riferimento

“Agent Architecture Considerations for Real-Time Planning in Games”, Jeff Orkin, 2005

Idea:

Usare un Planner tipo STRIP in real time per implementare l'AI degli NPC.

- ◆ Comportamenti più realistici
- ◆ Adattabilità alla situazione
- ◆ Tattiche di gruppo

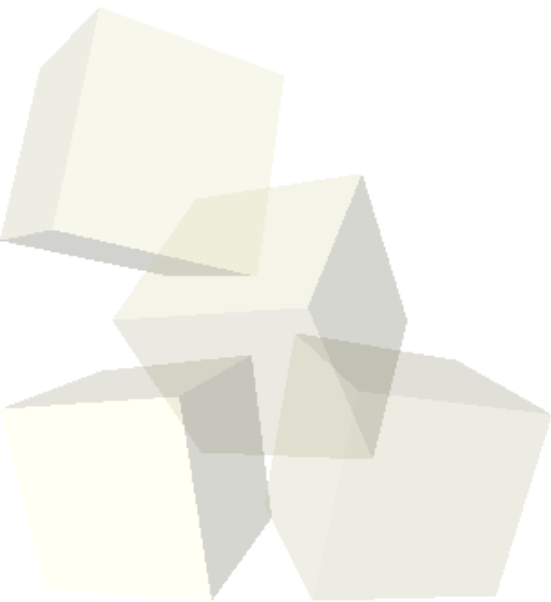




F.E.A.R.: Restrizioni

Il planning è un metodo dispendioso e, nel caso di un videogame, deve dividere le risorse computazionali con la grafica, l'animazione, la fisica etc. nel tempo di un frame.

Diverse tecniche devono essere usate per ottimizzare l'elaborazione dei piani.





L'agente ha una struttura simile a quella del MIT Media Lab C4.

- sensori
- sottosistemi
 - ◆ targetting system
 - ◆ navigation system
 - ◆ animation system
 - ◆ weapon system
- working memory
- blackboard
 - contiene le azioni che i sottosistemi devono svolgere





Natura:

- esterna (vista, suono..)
- interna (dolore, desideri)

Tipi:

- polling
per “catturare” proprietà dell'ambiente (potenziali “coperture”)
- event-driven
per eventi istantanei (danno, suoni)

Le conoscenze sono pubblicate come fatti nella *Working Memory*

Distribute Processing

I sensori sono sempre attivi, ma hanno tempi di refresh diversi.

Sensori a bassa complessità fanno update per ogni frame, altri (pathfinding, ray intersection) sono più lenti.



F.E.A.R.: Working Memory

Contiene la conoscenza. Realizza una sorta di “caching” dei fatti.

- 10 tipi di fatti diversi (*Task*, *Disturbance*, etc.)
- ogni fatto ha 16 campi diversi ad ognuno dei quali è affidato un Confidence Value tra 0.0 e 1.0.
Il Confidence Value ha un significato legato a quello dell'attributo (es. urgenza dell'attributo Desire)

Il planner usa questi fatti per verificare le precondizioni delle azioni.





Goal Oriented Action Planning (GOAP) basato su STRIP.

Fissato un goal, elaborare un “piano” di “azioni” per arrivare a soddisfarlo.

Differenze

- Non ci sono Add/Delete list. Il mondo ha un set di proprietà prefissate (1 set = uno stato del mondo).
- Le azioni hanno costi in maniera da “guidare” la scelta.
Il piano è un prodotto di un A* ottimizzato su un grafo dove le azioni sono archi e gli stati del mondo i nodi.
- Procedural Precondition
- Procedural Effects

Re-planning

se un piano diventa invalido nelle sue precondizioni il planner viene eseguito di nuovo.



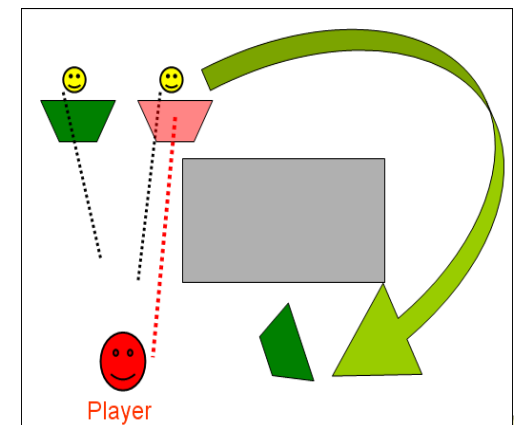
Esiste un AI a livello più alto che:

- organizza gli agenti in team per vicinanza
- assegna goal di team
 - ◆ **Get-to-Cover** Tutti i membri trovano un riparo e fanno fuoco di copertura.
 - ◆ **Advance-Cover** — Mentre fanno fuoco di copertura devono avanzare in ripari più vicini al giocatore.
 - ◆ **Orderly-Advance** — Movimento in fila ordinata verso una posizione, ogni membro copre un lato.
 - ◆ **Search** — Divisione in gruppi di due e ricerca sistematica in un'area.

Il goal di team è visto da un agente come goal ad alta priorità, ma può essere ignorato a favore di un goal interno (es. auto-conservazione).

Collaborazione/coordinazione: non è un meccanismo esplicito, ma emerge a causa dell'esecuzione non deterministica di goal propri e di team da parte dei vari agenti.

Esempio: Il giocatore invalida il riparo dell'agente (distrugge, etc.). L'agente esegue un re-plan del goal Get-to-Cover e calcola un path attorno al muro per arrivare ad un riparo valido. Il giocatore lo percepisce come una tattica di aggiramento.





Role Playing Games

I MAS in questo genere hanno avuto poco utilizzo: i nemici e i NPC sono molto basati su script tradizionali.

Cause (cosiderazioni personali)

- Causa contingente: forse sarebbero necessari troppi agenti
- Causa dinamica: combattimento generalmente non è molto importante
 - ◆ tattica di team non necessaria
 - ◆ semplicità dei nemici accettabile
- Causa gameplay: rischio di scardinare alcune meccaniche
 - ◆ es. quest devono essere prefissate

Sport

Generalmente c'è un IA centrale che dirige i personaggi: nel caso qualche autonomia sia data al singolo personaggio, questo è talmente semplice da essere difficilmente considerabile *agente*.



MAS sono usati anche per gli effetti speciali specialmente nel caso di scene di massa.

Es.

Il software *Massive* è stato usato per generare le scene con grandi folle in diverse produzioni per

■ Cinema

- ◆ Pirates of The Caribbean: At World's End.
- ◆ 300
- ◆ Die Hard 4
- ◆ The Chronicles of Narnia: Prince Caspian
- ◆ King Kong (Jackson 2005)
- ◆ Flags of our Fathers (per battaglie, folle, ma anche le flotte nel Pacifico)
- ◆ I, Robot
- ◆ Harry Potter and the Order of the Phoenix
- ◆ The Mummy: The Tomb Of The Dragon Emperor
- ◆ Etc.

■ Pubblicità