

Intelixencia Artificial aplicada a Videoxogos  
Top-Down en tempo real  
Grao en Enxeñaría Informática  
Universidade de Santiago de Compostela

Intelixencia Artificial aplicada a Videoxogos  
Top-Down en tempo real  
Grao en Enxeñaría Informática  
Universidade de Santiago de Compostela

Autor: Rubén Osorio López

Titor: Manuel Mucientes Molina  
Cotitor: Pablo Rodríguez Mier

21 de xullo de 2017

2017-07-20

# Intelixencia Artificial aplicada a Videoxogos Top-Down

Esto é a defensa da memoria do traballo de fin de grado nombrado **Intelixencia Artificial aplicada a Videoxogos Top-Down en tempo real**, eu son o autor, Rubén Osorio López, e os tutores son Manuel Mucientes Molina e Pablo Rodríguez Mier

Intelixencia Artificial aplicada a Videoxogos  
 Top-Down en tempo real  
 Grao en Enxeñaría Informática  
 Universidade de Santiago de Compostela

Autor: Rubén Osorio López

Titor: Manuel Mucientes Molina  
Coritor: Pablo Rodríguez May

21 de xullo de 2017

- 1 Introducción
- 2 Videoxogo baseado en axentes
- 3 Análise de requisitos
- 4 Xestión do proxecto
- 5 Arquitectura
- 6 Deseño
- 7 Validación e probas
- 8 Conclusións

### └ Táboa de contidos

Durante esta presentación seguiremos unha estrutura similar á memoria, centrándonos máis en algúnhos aspectos concretos do proxecto que expliquen en que consistiu o traballo realizado.

- 1 Introducción
- 2 Videoxogo baseado en axentes
- 3 Análise de requisitos
- 4 Xestión do proxecto
- 5 Arquitectura
- 6 Deseño
- 7 Validación e probas
- 8 Conclusións

- Proxecto que aborda a creación dun **videoxogo** con necesidades de **comportamento complexo** por parte do inimigo.

Videoxogo

## Loita 1 contra 1, Top-Down en dúas dimensións

Axente

Capaz de percibir e actuar sobre o **entorno competitivo** do videoxogo mediante **sensores** e **actuadores**

De forma general, búscase a creación dun videoxogo que requira un enemigo con comportamento complexo. O axente que representará o enemigo necesita ser un competidor capaz, para o que se realizou unha etapa de entrenamiento na que obtivo a información que necesitaba.

Para definir o videoxogo, **Loita 1 contra 1** significa que solamente dous personaxes competirán entre eles contando ambos as mesmas capacidades, accións posibles e atributos. **Top-Down** refírese ó plano picado utilizado para visualizar o combate. Por outra parte que sea en **dúas dimensións** implica que todo o contido do videoxogo son imaxes planas desenhadas unha a unha, sen que existan modelos en tres dimensións.

**Un axente** é aquilo capaz de percibir o entorno mediante **sensores** e actuar sobre o mesmo en consecuencia mediante **actuadores**, ambos son proporcionados pola súa interface co videoxogo. Ademáis atoparase nun entorno competitivo o que implica que buscará maximizar o seu **rendimiento** mentres se minimiza o do contrincante.

- ❖ Proyecto que aborda a creación dun **videoxogo** con necesidades de **comportamento complexo** por parte do inimigo.

Videocage

Loita 1 contra 1, Top-Down en dúas dimensións

**Appendix**

Capaz de percibir e actuar sobre o entorno competitivo do videoxogo mediante sensores e actuadores

2017-07-20

## Intelixencia Artificial aplicada a Videoxogos Top-Down

- Introdución

- Obxectivos

- Obxectivos

Os obxetivos to proxecto son os seguintes:

- Implementación do videoxogo: Xa que necesitaremos unha plataforma que nos permita que o axente e o xogador interactúen seguindo unha serie de regras comúns para competir entre eles.
- Necesitarase ademais implementar o axente capaz de desenvolverse correctamente durante a competición.
- O axente necesitará obter a información necesaria para logo comportarse adecuadamente grazas ó aprendido durante a etapa de entrenamento.
- Obteráanse datos sobre o rendemento do axente contra outras implementacións máis sencillas.
- Ca información obtida durante todo o proxecto, e especialmente na etapa anterior, realizarase un análise que describa o comportamento do axente e o que se conseguiu.

- Implementación do videoxogo

Os obxetivos to proyecto son os seguintes:

- Implementación do videoxogo: Xa que necesitaremos unha plataforma que nos permita que o axente e o xogador interactúen seguindo unha serie de regras comúns para competir entre eles.
- Necesitarase ademais implementar o axente capaz de desenvolverse correctamente durante a competición.
- O axente necesitará obter a información necesaria para logo comportarse adecuadamente grazas ó aprendido durante a etapa de entrenamento.
- Obteráanse datos sobre o rendemento do axente contra outras implementacións máis sencillas.
- Ca información obtenida durante todo o proxecto, e especialmente na etapa anterior, realizarase un análise que describa o comportamento do axente e o que se conseguiu.

- Implementación do videoxogo
- Implementación do axente

Os obxectivos to proxecto son os seguintes:

- Implementación do videoxogo: Xa que necesitaremos unha plataforma que nos permita que o axente e o xogador interactúen seguindo unha serie de regras comúns para competir entre eles.
- Necesitarase ademais implementar o axente capaz de desenvolverse correctamente durante a competición.
- O axente necesitará obter a información necesaria para logo comportarse adecuadamente grazas ó aprendido durante a etapa de entrenamento.
- Obteráanse datos sobre o rendemento do axente contra outras implementacións máis sencillas.
- Ca información obtenida durante todo o proxecto, e especialmente na etapa anterior, realizarase un análise que describa o comportamento do axente e o que se conseguiu.

- Implementación do videoxogo
- Implementación do axente
- Realizar o adestramento do axente

Os objetivos to proyecto son os seguintes:

- Implementación do videoxogo: Xa que necesitaremos unha plataforma que nos permita que o axente e o xogador interactúen seguindo unha serie de regras comunes para competir entre eles.
- Necesitarase ademáis implementar o axente capaz de desenvolverse correctamente durante a competición.
- O axente necesitará obter a información necesaria para logo comportarse adecuadamente gracias ó aprendido durante a etapa de entrenamiento.
- Obteránse datos sobre o rendemento do axente contra outras implementacións máis sencillas.
- Ca información obtenida durante todo o proyecto, e especialmente na etapa anterior, realizarase un análisis que describa o comportamento do axente e o que se conseguiu.

- Implementación do videoxogo
- Implementación do axente
- Realizar o adestramento do axente
- Obter datos sobre as capacidades do axente

Os objetivos to proyecto son os seguintes:

- Implementación do videoxogo: Xa que necesitaremos unha plataforma que nos permita que o axente e o xogador interactúen seguindo unha serie de regras comunes para competir entre eles.
- Necesitarase ademáis implementar o axente capaz de desenvolverse correctamente durante a competición.
- O axente necesitará obter a información necesaria para logo comportarse adecuadamente gracias ó aprendido durante a etapa de entrenamiento.
- Obteránse datos sobre o rendemento do axente contra outras implementacións máis sencillas.
- Ca información obtenida durante todo o proyecto, e especialmente na etapa anterior, realizarase un análisis que describa o comportamento do axente e o que se conseguiu.



- Implementación do videoxogo
- Implementación do axente
- Realizar o adestramento do axente
- Obter datos sobre as capacidades do axente
- Analizar os resultados obtidos

Os objetivos to proyecto son os seguintes:

- Implementación do videoxogo: Xa que necesitaremos unha plataforma que nos permita que o axente e o xogador interactúen seguindo unha serie de regras comúns para competir entre eles.
- Necesitarase ademais implementar o axente capaz de desenvolverse correctamente durante a competición.
- O axente necesitará obter a información necesaria para logo comportarse adecuadamente grazas ó aprendido durante a etapa de entrenamento.
- Obteráanse datos sobre o rendemento do axente contra outras implementacións máis sencillas.
- Ca información obtenida durante todo o proxecto, e especialmente na etapa anterior, realizarase un análise que describa o comportamento do axente e o que se conseguiu.

## Movimento

## Movimiento libre nunha habitación rectangular.

## Ataque

Permítese atacar a zona que se atopa cada onde o personaxe está mirando.

## Defensa

Posibilidade de defenderse dun ataque permitindo atacar se a defensa ten éxito

Para explicar agora o funcionamento do videoxogo, empezárase polas mecánicas principais que o compoñen: **O Movimiento** é libre nunha habitación rectangular o que suma a complexidade de evitar situacións nas que non se poida escapar do contrincante por estar ó lado dunha parede ou unha esquina. Ademáis a única maneira de mirar cara unha dirección é moverse cara ela.

**Como** solo se permite atacar a zona directamente enfrente do personaxe é importante ter en cuenta cada donde se está mirando. Isto favorece unha actitude agresiva pois hai que moverse na dirección do enemigo antes de atacalo.

**Pódese** realizar unha manioobra defensiva de alto riesgo e alta recompensa que permite evitar un ataque. Se se evita con éxito poderase realizar un ataque propio pero se non serase vulnerable durante uns instantes.

**Os tres estilos principais** que permiten estas mecánicas serán agresivo, defensivo ou contraataque. Nunha situación similar a un *pedra, papel, tixeiras* cada un deles gaña e perde contra ún dos outros dous eliminando así a posibilidade de adoptar un comportamento único que sexa óptimo en todos os casos.

Primeira implementación realizada con **Unity3D**, estándar de facto para videoxogos de este tamaño.

## Problemas de simulación

Impossibilidade de escalar o tempo sem romper o funcionamento do videoxogo.

2017-07-20

# Intelixencia Artificial aplicada a Videoxogos Top-Down

- └ Videoxogo baseado en axentes

## — Prototipo de Unity

## Prototipo de Unity

### Prototipo de Unity

Primeira implementação realizada com **Unity3D**, estándar de facto para videoxogos de este tamaño.

Problemas de simulación

Impossibilidade de escalar o tempo sem romper o funcionamento do videogame.

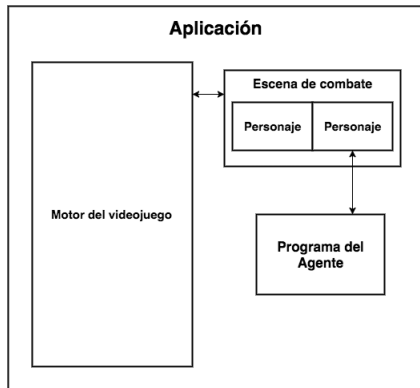
Para a primeira implementación destas mecánicas realizouse un prototipo inicial o antes posible no coñecido motor **Unity3D**, que é considerado estándar de facto para proxectos fora de grandes estudos.

Gracias a estratexia realizada para mitigar o **riesgo 9** fíxose que este se materializara nunha etapa inicial do proxecto evitando unha situación imposible de gestionar posteriormente.

**Ensinar vídeo:** Veremos agora um vídeo que mostra alguns destes problemas.

**Descubriuse** que facer pasar o tempo máis rápido dentro do videoxogo, pese a estar soportado polo motor, rompe o funcionamento de moitos elementos necesarios como as caixas de colisións que impiden que un personaxe se saia do mapa ou a funcionalidade que xestiona os ataques do personaxe. Nestas circunstancias é imposible realizar un entrenamiento acelerado válido para o axente polo que a aplicación non sería útil.

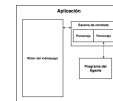
# Implementación de un motor desde cero en C++



# Intelixencia Artificial aplicada a Videoxogos Top-Down

- └ Videoxogo baseado en axentes
  - └ Segunda aplicación
    - └ Segunda aplicación

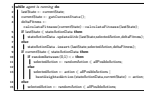
## Implementación de un motor desde cero en C++



A situación anterior fixo que surxira a necesidade de implementar a aplicación dende cero, para o cal se escolleu a lingua C++ cunha serie de librerías a baixo nivel.

**Na figura mostrada** podese ver como a aplicación completa estará composta por un motor que se encargará de todas as funcionalidades genéricas.

Ademáis contarase cunha escena de combate con dous personaxes, un dos  
cuaes está controlado polo programa do axente, aínda que poderían ser  
ámbos.



```

1 while agent is running do
2   lastState ← currentState;
3   currentState ← getCurrentState();
4   deltaFitness ←
     calculateFitness(currentState) - calculateFitness(lastState);
5   if lastState ∈ stateActionData then
6     stateActionData.updateWith(lastState,selectedAction,deltaFitness);
7   else
8     stateActionData.insert(lastState,selectedAction,deltaFitness);
9   if currentState ∈ stateActionData then
10    if randomBetween(0,1) < ε then
11      selectedAction ← randomAction ∈ allPossibleActions;
12    else
13      selectedAction ← action ∈ allPossibleActions |
        bestWeightedAction(stateActionData,currentState) = action;
14   else
15     selectedAction ← randomAction ∈ allPossibleActions;

```

Pasarase agora a explicar a grandes rasgos o funcionamento do algoritmo do axente. O mesmo é similar a unha implementación moi sencilla de *Q-Learning*.

Hasta a línea 8 da figura o que se fai é obter o estado actual do combate e calcular a diferenza de utilidade ou *fitness* entre este e o anterior. Logo actualízase a base de coñecemento do axente introducindo esta diferenza de fitness asociada a acción tomada para o último estado.

**A partir da liña 9** e hasta o final escollerase que acción se vai a levar a cabo cunha probabilidade proporcional ó fitness que se espera de cada unha delas. Se o estado non se coñece escollerase unha acción aleatoria.

Ademáis, aínda que o estado se visitara, se un valor aleatorio é menor que certo **epsilon** escollerase unha acción aleatoria de todas formas, facendo así que aínda que se coñeza ben o estado actual se permita explorar algunha das accións consideradas non optimas con máis frecuencia.

Parámetro	Valor
INITIAL_FITNESS_VALUE	1000
MY_HEALTH_MULTIPLIER	100
ENEMY_HEALTH_MULTIPLIER	100
DISTANCE_MULTIPLIER	3
LOOKING_BONUS	200
WALL_BONUS	50

```

input : playerHealth, enemyHealth, distance,
       lookingAtEnemy, noWallsNear
output: fitness
1 fitness ← INITIAL_FITNESS_VALUE;
2 fitness ← fitness +(playerHealth *
  MY_HEALTH_MULTIPLIER);
3 fitness ← fitness -(enemyHealth *
  ENEMY_HEALTH_MULTIPLIER);
4 fitness ← fitness -(distance *
  DISTANCE_MULTIPLIER);
5 if lookingAtEnemy then
6   fitness ← fitness + LOOKING_BONUS;
7 if noWallsNear then
8   fitness ← fitness + WALL_BONUS;

```

Parámetro	Valor
INITIAL_FITNESS_VALUE	1000
MY_HEALTH_MULTIPLIER	100
ENEMY_HEALTH_MULTIPLIER	100
DISTANCE_MULTIPLIER	3
LOOKING_BONUS	200
WALL_BONUS	50

Para obter o **fitness** simplemente se considerarán os valores de certas variables que representan o estado do combate. A saude do personaxe aumenta de forma **directamente proporcional** ó fitness mentres que a saude do inimigo e a distancia o fan de forma **inversamente proporcional**.

Ademáis proporcionaránse dous bonus por estar mirando cara o enemigo e por evitar posicionarse ó lado dos muros da escena.

Os valores usados para modificar o peso de cada unha das variables que se ven no cadro foron obtidos grazas a coñecemento experto sobre o combate. Céntranse en buscar un comportamento agresivo e eficaz por parte do personaxe que dea lugar a pelexas dinámicas e entreteidas para un xogador real.

## Contra axente baseado en regras

Pretende simular un aprendizaxe contra xogadores reais.

## Contra él mesmo

Buscando unha exploración mais extensa de estados que o axente baseado en regras non pode aportar.

2017-07-20

## Intelixencia Artificial aplicada a Videoxogos Top-Down

- └ Videoxogo baseado en axentes

- └ Tipos de adestramento

- └ Tipos de adestramento

O algoritmo usarase de dúas formas para entrenar ó axente, ben facendoo pelear contra outro axente baseado en regras simulando a un xogador real de forma moi sinxela. Ou ben facendoo pelear contra él mesmo buscando a exploración estados mui inusuales cuando se combate contra o axente baseado en regras.

Contra axente baseado en regras

Pretende simular un aprendizaxe contra xogadores reais.

Contra él mesmo

Buscando unha exploración mais extensa de estados que o axente baseado en regras non pode aportar.

- └ Análise de requisitos
  - └ Casos de uso
    - └ Casos de uso



Entrando ahora dentro do bloque de analisis de requisitos, empezouse por identificar os diferentes casos de uso existentes no proxecto.

Empezando pola parte esquerda vemos en azul e rojo os casos de uso de abrir e cerrar a aplicación. En tonos **verdes** temos os casos de uso relacionados con seleccionar unha opción do menú e entrar na escena de combate apropiada, esta pode involucrar unha pelexa entre **dous axentes**, un **xogador real e un axente** ou **dous xogadores reales**.

Finalmente vemos en amarillo o caso de uso de visualizar os resultados acumulados dos combates o cual será necesario para ver o rendimiento do axente de forma sencilla especialmente cuando realiza combates sen que estes se mostren por pantalla.



- RF-1/2/3: Funcionalidades do menú
- RF-4/15: Consola con comandos/resultados
- RF-5: Saír da aplicación
- RF-6: Entrar na escena de combate
- RF-7/8/9: Moverse/Atacar/Defender
- RF-10: Gañar/Perder partida
- RF-11: Esgotar o tempo de combate
- RF-12: Volver ó menú
- RF-13: Visualizar combate entre axentes
- RF-14: Simular múltiples combates
- RNF-1: Rendemento da aplicación
- RNF-2: **Velocidade das simulacións**
- RNF-3: Extensibilidade do motor
- RNF-4: Facilitade para depurar
- RNF-5: Aplicación autocontida
- RNF-6: Extensibilidade de escenas
- RNF-7: Documentación
- RNF-8: Usabilidade da interface

Dos anteriores casos de uso surxen unha serie de requisitos:

O primeiro grupo de requisitos funcionales relacionarase con moverse e seleccionar opcións do menú e o seguinte cas funcionalidades da consola. Desde o requisito 5 hasta o 12 defínense funcionalidades relacionadas co combate e finalmente os dous últimos céntranse especificamente nos combates entre axentes.

Dentro do conxunto de requisitos non funcionales vense algúnhos relacionados co rendimientto da aplicación, a documentación ou usabilidade. Destácase o **requisito non funcional 2** xa que é o que define formalmente a necesidade de poder simular peleas entre axentes de forma acelerada, o cual, como se viu anteriormente, tuvo un peso importante no proxecto.

## Identificación de riesgos

Identificáronse e especificáronse un total de **10** riscos, destacándose o **RSK-1** (retraso na planificación), **RSK-2** (cambio de requisitos) e **RSK-9** (dificultade para realizar simulacións).

## Acción correctiva AC-1

Xestionou a materialización do **RSK-9** e implicou implementar dende cero a aplicación do videoxogo sen facer uso dos motores dispoñibles.

2017-07-20

# Intelixencia Artificial aplicada a Videoxogos Top-Down

- └ Xestión do proxecto
  - └ Xestión de riscos
    - └ Xestión de riscos

Chegando á gestión de riscos, identificáronse un total de 10, dos cuales destácanse pola súa probabilidade é impacto o **1** relacionado con retrasos na planificación, o **2** relacionado con cambios nos requisitos e o **9** relacionado con non poder realizar simulacións de combates.

A única acción correctiva que e levou a cabo é a asociada ó prototipo de Unity e á implementación da aplicación desde cero como xa se mencionou anteriormente.

## Contexto do proxecto

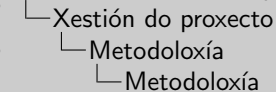
- Traballador único
- Duración relativamente corta
- Necesidade de avanzar rapidamente nas etapas iniciais

## Programación Extrema

- Flexibilidade ante cambios
- Evitase utilizar demasiado tempo en tarefas de xestión
- Rápida iteración
- Reunións entre *sprints*

2017-07-20

## Intelixencia Artificial aplicada a Videoxogos Top-Down



Metodoloxía
Contexto do proxecto
• Traballador único
• Duración relativamente corta
• Necesidade de avanzar rapidamente nas etapas iniciais
Programación Extrema
• Flexibilidade ante cambios
• Evitase utilizar demasiado tempo en tarefas de xestión
• Rápida iteración
• Reunións entre sprints

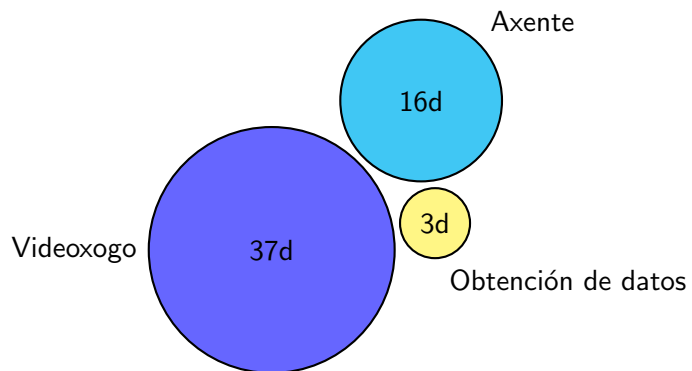
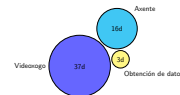
A elección de metodoloxía tuvo moito que ver coas circunstancias do proxecto. Neste caso non era necesario dedicar moitos recursos a coordinar ós traballadores xa que solo había un, a duración do proxecto foi relativamente corta comparada con outros dentro da industria dos videoxogos e necesitábase avanzar rapidamente para rematar a aplicación o antes posible é poder dedicar tempo ó axente.

Dentro das metodoloxías áxiles escolleuse **Programación Extrema** pola súa flexibilidade ante os cambios que probablemente ocorrerían. Ademais non se dedica un tempo excesivo a tarefas de gestión que ralenticen o desenvolvemento do proxecto. E os Sprints realizados entre as distintas reunións permiten que os tutores axuden a detectar problemas en ventás de tempo muy pequenas.

2017-07-20

## Intelixencia Artificial aplicada a Videoxogos Top-Down

- └ Xestión do proxecto
  - └ Planificación temporal
    - └ Planificación temporal



A planificación temporal final divídese tal e como se mostra nesta figura, pero os grupos de tarefas que se levaron a cabo durante todo o proxecto como documentación e reunións non están presentes.

Vemos como o desenvolvemento do prototipo en Unity e da aplicación final teñen un peso muy importante na planificación con 37 días, o que deixou uns 16 días de traballo para a creación e entrenamento do axente. Finalmente dedicáronse 3 días a obter datos sobre o rendemento do mesmo.

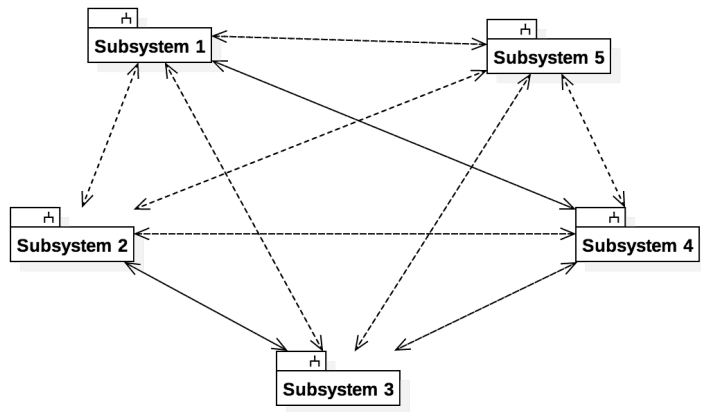
## Custos a incluir

- Soldo do traballador: 3952.50 €
- Amortización do equipo de traballo: 194.90 €
- Impresión da memoria + CD: 100 €
- 10 % de custos indirectos: 424.74 €

O que resulta nun custo final de **4672.14 €**

De forma breve, mostrase aquí que elementos contribuíron a aumentar o coste do proxecto dando lugar a un importe final de **4672.14 €**. Débese destacar que todas as ferramentas usadas eran ou ben libres ou versións de proba polo que non implicaban un coste monetario.

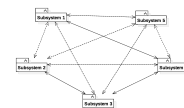
## Subsistemas conectados



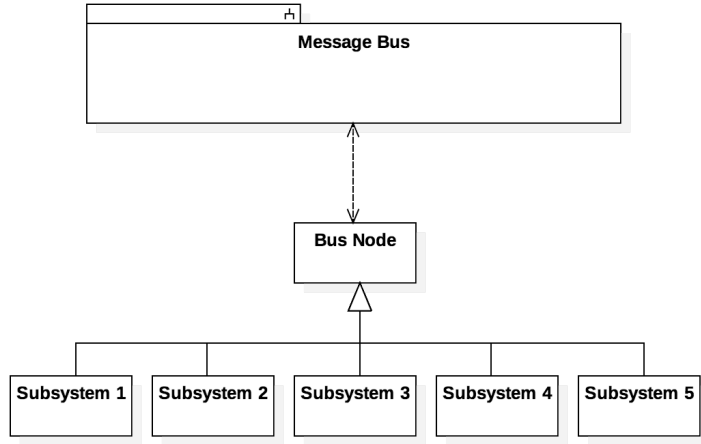
2017-07-20

# Intelixencia Artificial aplicada a Videoxogos Top-Down

- └─Arquitetura
  - └─Arquitetura do sistema
    - └─Subsistemas conectados



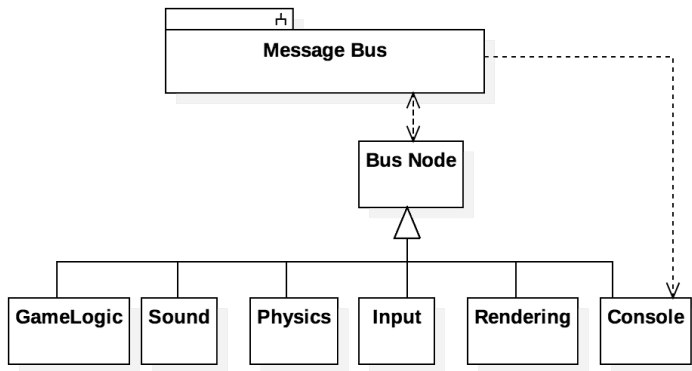
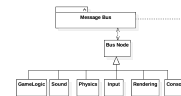
A arquitectura da aplicación componse por unha serie de subsistemas que necesitan conectarse unhas cos outras con moita frecuencia. Na figura vemos como unha aproximación simple na que todos teñen referencias a todos pode levar a moitos problemas orixinados por ter que cambiar calquera cousa en algún deles.



A solución deseñada foi crear un bus de mensaxes. Nesta arquitectura todos os subsistemas son un nodo dese bus, podendo comunicarse con calquera dos outros subsistemas ou con todos á vez mediante o envío de mensaxes, evitando referencias innecesarias. Nesta arquitectura cada subsistema elige como gestionar os mensaxes que recibe.

A arquitectura terá un aspecto similar á seguinte:

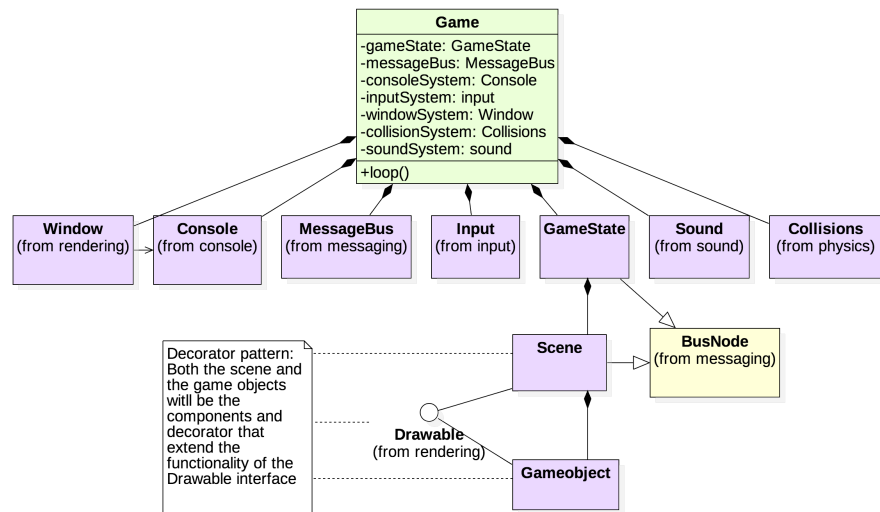
- Arquitectura
  - Arquitectura do sistema
    - Arquitectura final



Aquí vemos os diferentes subsistemas existentes así como unha dependencia do bus de mensajes ca co subsistema da consola. Isto permite que en tempo de execución se poida visualizar en calquera momento calquera mensaje relevante na ventá da propia aplicación ademáis de introducir mensajes co contido que se queira dentro do bus de mensajes.

Esto resultou extremadamente útil xa que facilitou moito o proceso de debug e axudou a unha iteración muy rápida.

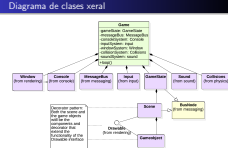




2017-07-20

↳ Diseño

↳ Diagrama de clases xeral



No que respecta a diseño e implementación, para non entrar demasiado en materia, mostrase solamente o diagrama que representa a clase principal de cada subsistema e a clase *xogo* que os contén e itera sobre eles.

Ademáis podese ver como se conta ca clase *scene*, que representa calquera escena na que pode estar o videoxogo como un menú ou o combate. Esta clase contará cunha serie de *GameObjects* que poderán ser calquera entidade dinámica no contexto do videoxogo como un personaxe, unha barra de vida ou unha opción do menú. Isto permite muita flexibilidade á hora de implementar cada unha de estas entidades.

2017-07-20

- Validación e probas

- Aplicación

- Validación e probas da aplicación

#### Probas unitarias

Unha ou mais probas por cada requisito tanto funcional como non funcional superadas na súa totalidade.

#### Probas de integración

Comproban a integración entre subsistemas e do o axente ca aplicación.

### Probas unitarias

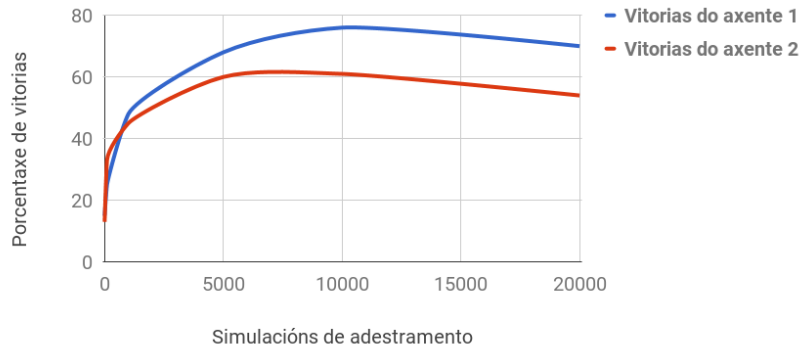
Unha ou mais probas por cada requisito tanto funcional como non funcional superadas na súa totalidade.

### Probas de integración

Comproban a integración entre subsistemas e do o axente ca aplicación.

Chegando xa ó apartado de validación e probas, realizáronse unha serie de probas unitarias para os múltiples requisitos funcionales ou non do proxecto.

Ademáis levaronse a cabo probas de integración para comprobar o funcionamento dos subsistemas conectados e do axente dentro da aplicación.



2017-07-20

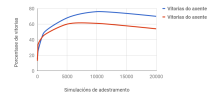
## Intelixencia Artificial aplicada a Videoxogos Top-Down

Validación e probas

Validación do Axente

Comparativa de vitorias

Comparativa de vitorias



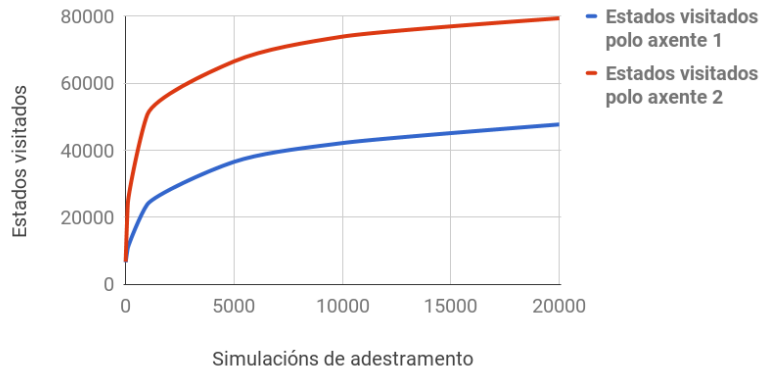
En relación á **validación do axente** realizase nesta figura un análise que compara o rendemento do **axente 1** entrenado únicamente contra o personaxe baseado en regras contra o **axente 2** entrenado únicamente contra él mesmo.

Dado que para calcular o porcentaxe de victorias se realizaron 100 combates contra o personaxe baseado en regras, obsérvase como o axente que entrenou contra él representado pola línea azul é significativamente mellor que o outro.

Tamén é interesante como o axente 2 gaña con moi poucas simulacións debido a que está aprendendo o dobre de rápido xa que os dous personaxes están aprendendo á vez e combinando a información aprendida.

Finalmente hay que destacar como unha vez que se superan uns 10.000 combates se sofre de *overfitting* ou sobreaxuste. Neste caso, pese a que o fitness dos estados siga aumentando co entrenamento, o mesmo non ten por que estar relacionado linealmente coa probabilidade de obter unha victoria.

## Comparativa de estados visitados



2017-07-20

# Intelixencia Artificial aplicada a Videoxogos Top-Down

- └ Validación e probas
  - └ Validación do Axente
    - └ Comparativa de estados visitados



Mui brevemente, nesta gráfica vemos como o axente que entrena contra él mesmo sí realiza unha exploración de estados moito máis extensa e dunha forma moito máis rápida polo que é menos probable que ó combatir contra un xogador real se atope en situacións nas que non sepa como responder.

A partir de unhas 20.000 iteracións o aumento de estados visitados non compensa o tempo necesario para seguir entrenando.

## Combinación de ámbolos dous métodos de adestramento.

## Contra o axente baseado en regras

Favorece un aprendizaxe moi rápido nas primeiras simulacións.

## Contra él mismo

Aporta unha exploración de estados superior.

Finalmente o axente escollido intenta combinar o mellor dos dous métodos de entrenamiento, empezando a aprender contra o axente baseado en regras para aprender rápidamente a gañar partidas e seguindo con combates contra él mismo para ter acceso a unha grán cantidade de estados.

Ahora mostrarase o comportamento do axente en tempo real nun **combate contra un xogador humano**, que seréi eu. E despois describiranse dous comportamentos interesantes que aprendeu na etapa de **entrenamento ca axuda dun vídeo**.

## Logros do proxecto

- O comportamento, aspecto e rendemento da aplicación cumpriu as expectativas.
- O axente é capaz de competir contra outras implementacións e contra xogadores humanos.

## Leccións aprendidas

- Importancia de ter en conta os posibles riscos do proxecto o antes posible.
- Utilidade de un deseño flexible previo á implementación.
- Calidade dos resultados de implementacións sinxelas de Intelixencia Artificial.

2017-07-20

## Intelixencia Artificial aplicada a Videoxogos Top-Down

### Conclusións

### Conclusións e leccións aprendidas

### Conclusións e leccións aprendidas

#### Logros do proxecto

- O comportamento, aspecto e rendemento da aplicación cumpriu as expectativas.
- O axente é capaz de competir contra outras implementacións e contra xogadores humanos.

#### Leccións aprendidas

- Importancia de ter en conta os posibles riscos do proxecto o antes posible.
- Utilidade de un deseño flexible previo á implementación.
- Calidade dos resultados de implementacións sinxelas de Intelixencia Artificial.

A modo de conclusión, considérase que a aplicación cumpriu as expectativas xa que o seu rendemento é mellor do esperado. Ademais o axente resultou ser muy capaz incluso contra xogadores relativamente experimentados neste tipo de videoxogos e este en concreto.

Algunhas das leccións importantes que se aprenderon temos:

**A importancia de dedicar** un momento a considerar que pode saír mal en calquera proxecto para lidiar cos problemas antes de que sean demasiado importantes.

**As facilidades** que proporciona seguir un deseño flexible e definido en vez de empezar a implementar directamente por considerar que non hay o tempo suficiente.

**E que as técnicas intelixencia artificial**, incluso con implementacións tan sencillas como esta, son realmente capaces e os seus límites están máis alá do que nun primeiro momento se podería pensar.

- 2017-07-20

## -Conclusiones

## – Posibles ampliaciones

- Posibles ampliaciones

- ♥ Melloras de compatibilidade.
- ♥ Ampliación do proceso de probas con xogadores humanos
- ♥ Implementación de máis técnicas para o axente.
- ♥ Novas mecánicas para o videoxogo.

**A mais sinxela** e util seguramente sería mellorar a compatibilidade da aplicación portándoa a outros sistemas operativos como Linux ou Windows. Xa que todo o código é C++ e as librerías están dispoñibles para eses sistemas non debería dar moitos problemas. Como muito pode ser que se necesite un instalador en sistemas Windows.

**Poderíase tamén ampliar o proceso de probas** compoñendo grupos de xogadores humanos que compitan contra diferentes versións do axente e cubran un cuestionario sobre o rendemento do mesmo, como simula a un humano ou a dificultade que presenta.

**Finalmente poderíanse implementar técnicas máis complexas de intelixencia artificial, empezando por *Q-Learning* xa que non requiriría demasiados cambios. Sería interesante utilizar *Mapas Autoorganizados*, unhas redes neuronales utilizadas para discretizar os estados de forma automática e non de forma manual.**

**SI as novas técnicas resultan ser demasiado efectivas**, pode ser que se necesiten máis mecánicas que aumenten a complexidade do videoxogo tales como ataques con rango, máis opcións de movemento ou novas accións defensivas.

# Intelixencia Artificial aplicada a Videoxogos Top-Down en tempo real

Grao en Enxeñaría Informática  
Universidade de Santiago de Compostela

Autor: Rubén Osorio López

Titor: Manuel Mucientes Molina  
Cotitor: Pablo Rodríguez Mier

21 de xullo de 2017

2017-07-20

## Intelixencia Artificial aplicada a Videoxogos Top-Down

- └─ Conclusións
- └─ Posibles ampliacións

Intelixencia Artificial aplicada a Videoxogos  
Top-Down en tempo real  
Grao en Enxeñaría Informática  
Universidade de Santiago de Compostela

Autor: Rubén Osorio López

Titor: Manuel Mucientes Molina  
Cotitor: Pablo Rodríguez Mier

21 de xullo de 2017

Esto é todo pola miña parte, pñoome a disposición do tribunal para intentar responder ás preguntas que poidan ter sobre o proxecto