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 Rodrígez Mier

21 de xullo de 2017

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 Rodrígez Mier

21 de xullo de 2017

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 Rodrígez Mier

Táboa de contidos

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

Durante esta presentación seguiremos unha estructura similar á memoria, centrándonos máis en algunhos aspectos concretos do proyecto que expliquen en que consistiu o traballo realizado.

Introdución

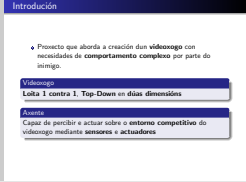
- Proxecto que aborda a creación dun **videoxogo** con necesidades de **comportamento complexo** por parte do inimigo.

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 videojuego, **Loita 1 contra 1** significa que solamente dous perxonajes competirán entre eles contando ambos cas mismas capacidades, acciói̇s posibles e atributos. **Top-Down** refierese ó plano picado utilizado para visualizar o combate. Por outra parte que sea en **dúas dimensiói̇s** implica que todo o contido do videoxogo son imagenes planas dibujadas unha a unha, sin que existan modelos en tres dimensiois.

Un axente é aquilo capaz de percibir o entorno mediante **sensores** e actuar sobre o mismo 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.

Obxectivos

Obxectivos

Os objetivos do proyecto foron os siguientes: (**CLICK**)

- 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 además **implementar o axente capaz** de desenvolverse correctamente durante a competición.
- O axente necesitará **obtener a información necesaria para logo comportarse adecuadamente** gracias ó aprendido durante a etapa de entrenamiento.
- **Obteránse datos sobre o rendemento do axente** contra outras implementacióis máis sencillas.
- **Ca información obtenida** durante todo o proyecto, e especialmente na etapa anterior, **realizarase un análisis que describa o comportamiento do axente** e o que se conseguiu.

Obxectivos

- Implementación do videoxogo

Obxectivos
Implementación do videoxogo

Os objetivos do proyecto foron os siguientes: (**CLICK**)

- 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 además **implementar o axente capaz** de desenvolverse correctamente durante a competición.
- O axente necesitará **obtener a información necesaria para logo comportarse adecuadamente** gracias ó aprendido durante a etapa de entrenamiento.
- **Obteránse datos sobre o rendemento do axente** contra outras implementacióis máis sencillas.
- **Ca información obtenida** durante todo o proyecto, e especialmente na etapa anterior, **realizarase un análisis que describa o comportamiento do axente** e o que se conseguiu.

Obxectivos

- Implementación do videoxogo
- Implementación do axente

Obxectivos
- Implementación do videoxogo
- Implementación do axente

Os objetivos do proyecto foron os siguientes: (**CLICK**)

- 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 además **implementar o axente capaz** de desenvolverse correctamente durante a competición.
- O axente necesitará **obtener a información necesaria para logo comportarse adecuadamente** gracias ó aprendido durante a etapa de entrenamiento.
- **Obteránse datos sobre o rendemento do axente** contra outras implementacióis máis sencillas.
- **Ca información obtenida** durante todo o proyecto, e especialmente na etapa anterior, **realizarase un análisis que describa o comportamiento do axente** e o que se conseguiu.

Obxectivos

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

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

Os objetivos do proyecto foron os siguientes: (**CLICK**)

- 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 además **implementar o axente capaz** de desenvolverse correctamente durante a competición.
- O axente necesitará **obtener a información necesaria para logo comportarse adecuadamente** gracias ó aprendido durante a etapa de entrenamiento.
- **Obteránse datos sobre o rendemento do axente** contra outras implementacióis máis sencillas.
- **Ca información obtenida** durante todo o proyecto, e especialmente na etapa anterior, **realizarase un análisis que describa o comportamiento do axente** e o que se conseguiu.

Obxectivos

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

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

Os objetivos do proyecto foron os siguientes: (**CLICK**)

- 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 además **implementar o axente capaz** de desenvolverse correctamente durante a competición.
- O axente necesitará **obtener a información necesaria para logo comportarse adecuadamente** gracias ó aprendido durante a etapa de entrenamiento.
- **Obteránse datos sobre o rendemento do axente** contra outras implementacióis máis sencillas.
- **Ca información obtenida** durante todo o proyecto, e especialmente na etapa anterior, **realizarase un análisis que describa o comportamiento do axente** e o que se conseguiu.

Obxectivos

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

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

Os objetivos do proyecto foron os siguientes: (**CLICK**)

- 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 además **implementar o axente capaz** de desenvolverse correctamente durante a competición.
- O axente necesitará **obtener a información necesaria para logo comportarse adecuadamente** gracias ó aprendido durante a etapa de entrenamiento.
- **Obteránse datos sobre o rendemento do axente** contra outras implementacióis máis sencillas.
- **Ca información obtenida** durante todo o proyecto, e especialmente na etapa anterior, **realizarase un análisis que describa o comportamiento do axente** e o que se conseguiu.

Mecánicas

Movemento

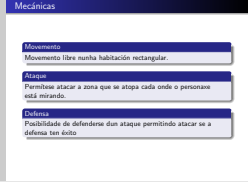
Movemento libre nunha habitación rectangular.

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



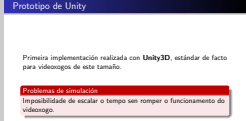
Para explicar ahora o funcionamiento do videoxogo, empezarase polas mecánicas principales que o compoñen: **O Movimiento** é libre nunha habitación rectangular o que suma a complexidade de evitar situacióis nas que non se poida escapar do contrincante por estar ó lado dunha parede ou unha esquina. Ademáis a única maneira de mirar cara unha dirección é moverse cara ela.
Como solo se permite atacar a zona directamente enfrente do personaje **é importante ter en cuenta cada donde se está mirando**. Esto favorece unha **actitude agresiva** pois hai que moverse na dirección do enemigo antes de atacalo.
Pódese realizar unha maniobra 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 principales 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.

Prototipo de Unity

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

Problemas de simulación

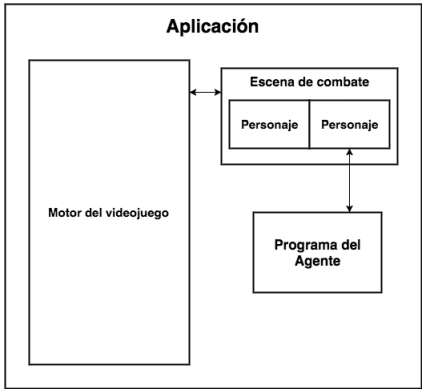
Imposibilidade de escalar o tempo sen romper o funcionamento do videoxogo.



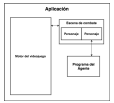
Para a primeira implementación destas mecánicas realizouse un prototipo inicial no coñecido motor **Unity3D**, que é considerado **estándar de facto** para proxectos fora de grandes estudios.
Gracias a estratexia realizada para mitigar un **riesgo que se mostrará en apartados posteriores** fíxose que este se materializara nunha etapa inicial do proyecto evitando unha situación imposible de gestionar posteriormente.
Ensinar vídeo: Veremos ahora un vídeo que mostra algúns destes problemas.
Descubriuse que facer pasar o tempo máis rápido dentro fo videoxogo, pese a estár 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 entrenamento acelerado válido para o axente polo que a aplicación non sería útil.

Segunda aplicación

Implementación de un motor desde cero en C++



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 linguaxe **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**.

Además contarase cunha escena de combate con dous personajes, **un dos cuales está controlado** polo programa do axente, ainda 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 ∈ allPosibleActions;
12         else
13             selectedAction ← action ∈ allPosibleActions |
                 bestWeightedAction(stateActionData,currentState) = action;
14     else
15         selectedAction ← randomAction ∈ allPosibleActions;
```


Algoritmo

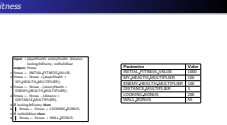
Pasarase ahora a explicar a grandes rasgos o funcionamento do algoritmo. O mesmo é similar a unha implementación simplificada de *Q-Learning*. Hasta a línea 8 da figura o que se fai e **obter o estado actual do combate** e calcular a **diferencia de utilidade** ou *fitness* entre este e o anterior. Logo actualizarase a base de conocimiento do axente introducindo esta diferencia 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 fitnes que se espera de cada unha delas. Se o estado non se coñece escollerase unha acción aleatoria.

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

```
input  : playerHealth, enemyHealth, distance,
         lookingAtEnemy, noWallsNear
output: fitness
fitness ← INITIAL_FITNESS_VALUE;
fitness ← fitness +(playerHealth ∗
  MY_HEALTH_MULTIPLIER);
fitness ← fitness −(enemyHealth ∗
  ENEMY_HEALTH_MULTIPLIER);
fitness ← fitness −(distance ∗
  DISTANCE_MULTIPLIER);
if lookingAtEnemy then
    fitness ← fitness + LOOKING_BONUS;
if noWallsNear then
    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. O fitness aumenta de forma **directamente proporcional** á saude do personaxe e de forma **inversamente proporcional** á saude do inimigo e a distancia ó mismo.

Ademáis proporcionaránse **dous bonus** por estár **mirando cara o contrincante** 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 contexto do combate**. Céntranse en buscar un **comportamento agresivo e eficaz** por parte do personaxe que dea lugar a **pelexas dinámicas e entretenidas** para un xogador real.

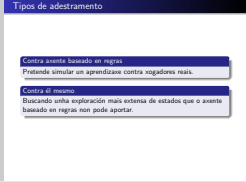
Tipos de adestramento

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.



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

Casos de uso

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

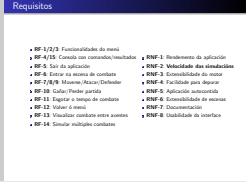
Empezando pola parte izquierda vemos en azul e rojo os casos de uso de **abrir e cerrar a aplicación**. En tonos **verdes** temos os casos de uso relacionados con seleccionar unha opción do menú e entrar na escena de combate apropiada, esta pode involucrar unha pelexa entre **dous axentes**, un **xogador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

Requisitos

- **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**: Facilidade 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 siguiente cas funcionalidades da consola. Desde o requisito 5 hasta o 12 defínense funcionalidades relacionadas co combate e finalmente os dous últimos céntranse específicamente nos combates entre axentes.

Dentro do conxunto de requisitos non funcionales vense algunhos relacionados co **rendimiento 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, tuvo un peso importante no proyecto.

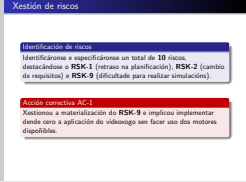
Xestión de riscos

Identificación de riscos

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.



Chegando á gestión de riesgos, 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.

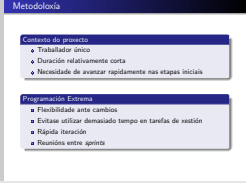
Metodoloxía

Contexto do proxecto

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

Programación Extrema

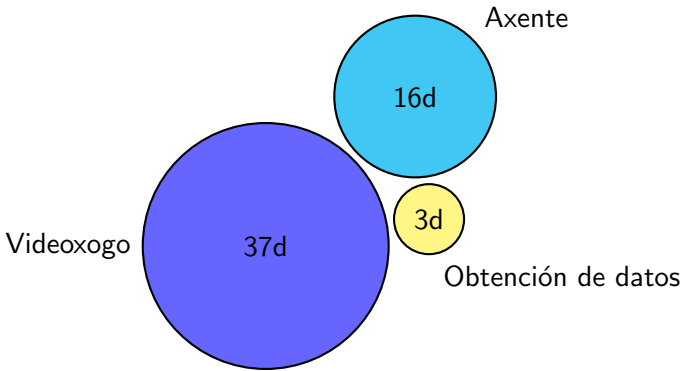
- Flexibilidade ante cambios
- Evitase utilizar demasiado tempo en tarefas de xestión
- Rápida iteración
- Reunións entre *sprints*



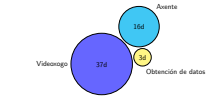
A elección de metodología tuvo muito que ver cas circunstancias do proyecto. Neste caso **non era necesario dedicar muitos recursos a coordinar ós traballadores** xa que solo había un, a **duración** do proyecto foi relativamente corta **comparada con outros dentro da industria dos videoxogos** e necesítábase avanzar rápidamente para **rematar a aplicación o antes posible** é poder dedicar tempo ó axente.

Dentro das metodologías ágiles escolleuse **Programación Extrema** pola súa **flexibilidade ante os cambios que probablemente ocurrirían**. Además non se dedica un **tempo excesivo a tareas de gestión que ralenticen o desenvolvemento do proyecto**. E os Sprints realizados entre as distintas reunións permiten que os **tutores axuden a detectar problemas** en ventás de tempo muy pequenas.

Planificación temporal



Planificación temporal



A planificación temporal final dividese tal e como se mostra nesta figura, pero os grupos de tareas que se levaron a cabo durante todo o proyecto como **documentación e reunións non están presentes** na misma.

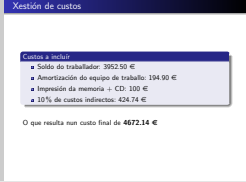
Vemos como o desarrollo do prototipo en Unity e da aplicación final teñen un peso muy importante na planificación con 37 días, o que deixóu unhos 16 días de traballo para a creación e entrenamiento do axente. Finalmente dedicáronse 3 días a obter datos sobre o rendimiento do mismo.

Xestión de custos

Custos a incluír

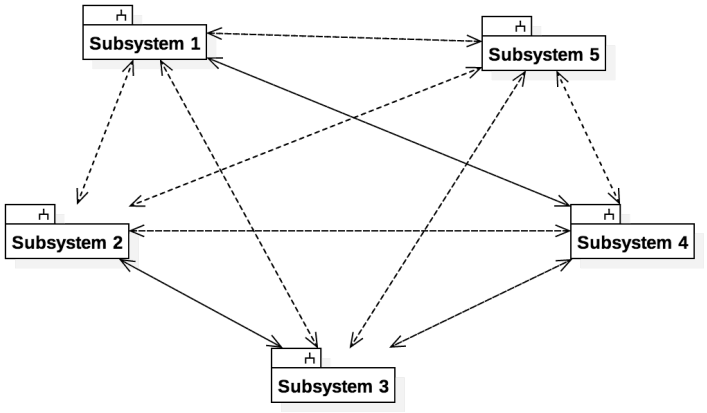
- Soldo do traballador: 3952.50 €
- Amortización do equipo de traballo: 194.90 €
- Impresión da memoria + CD: 100 €
- 10 % de custos indirectos: 424.74 €

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



De forma breve, mostrase aquí **qué elementos contribuiron** a aumentar o coste do proyecto 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.

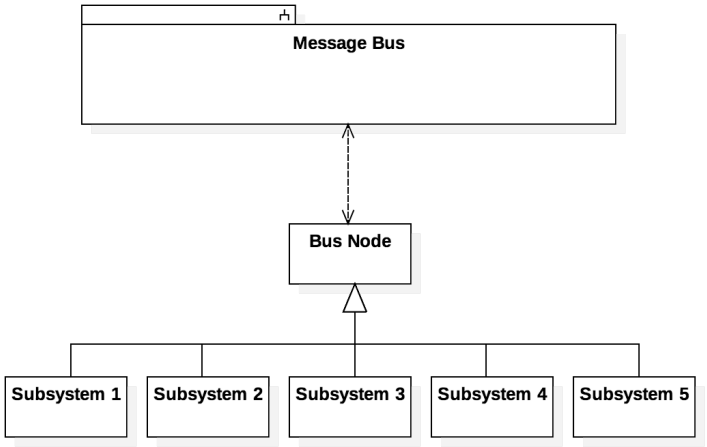
Subsistemas conectados

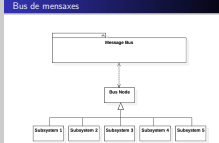


Subsistemas conectados

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

Bus de mensaxes

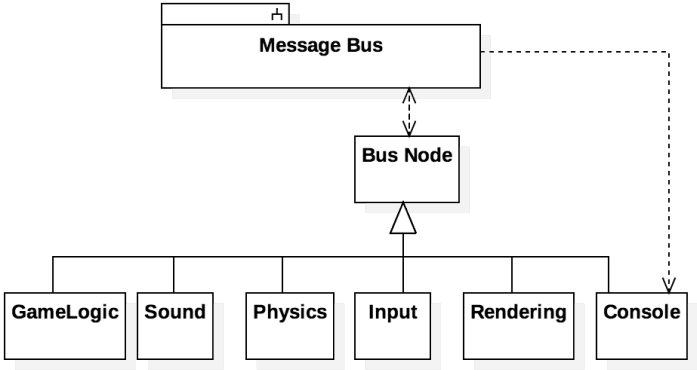


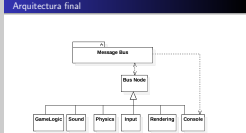


A solución diseñada foi crear un **bus de mensajes**. Nesta arquitectura todos os **subsistemas son un nodo** dese bus, podendo comunicarse con calquera dos outros subsistemas ou con todos á vez mediante o **envío de mensajes**, evitando referencias innecesarias. Nesta arquitectura cada subsistema elige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

A arquitectura terá un aspecto similar á siguiente:

Arquitectura final

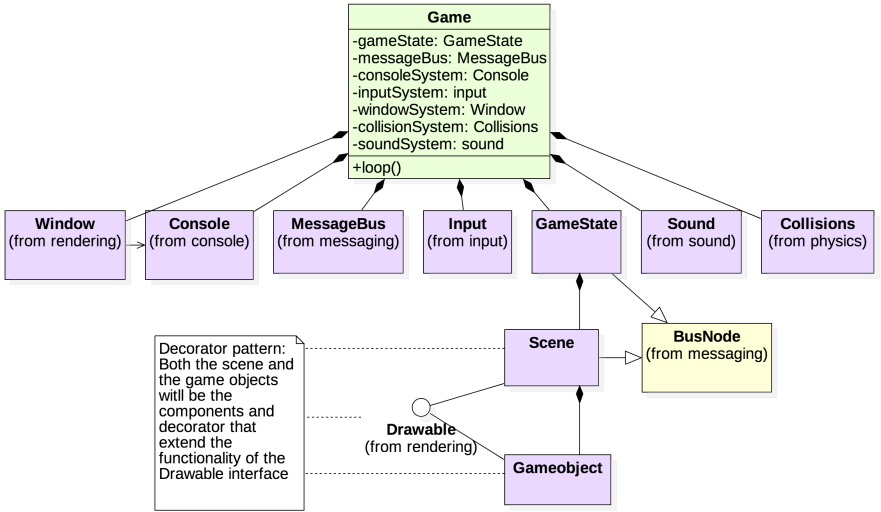


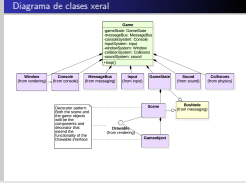


Aquí vemos os diferentes subsistemas existentes así como unha **dependencia** do bus de mensajes ca co subsistema da **consola**. Esto permite que en tempo de ejecución se poida **visualizar** en calquera momento calquera mensaje relevante na ventá da propia aplicación ademáis de **introducir mensajes**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

Esto resultóu extremadamente útil xa que facilitóu moito o proceso de **debug** e axudou a unha **iteración muy rápida**.

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.

Además podese ver como se conta ca clase **scene**, que representa calquera **escena** na que pode estár 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 personaje, unha barra de vída ou unha opción do menú. Esto permite muita **flexibilidade á hora de implementar** cada unha de estas entidades.

Patróns: Decorator, Pub-Sub, Singleton e Strategy.

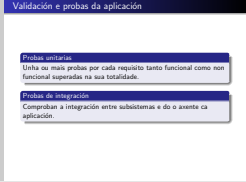
Validación e probas da aplicación

Probas unitarias

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

Probas de integración

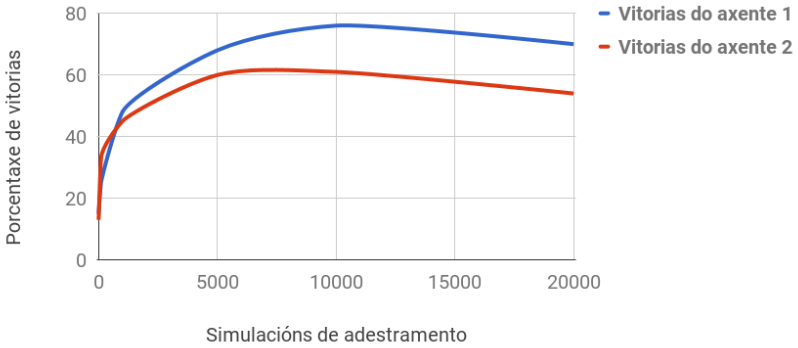
Comproban a integración entre subsistemas e do o axente ca aplicación.



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

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

Comparativa de vitorias



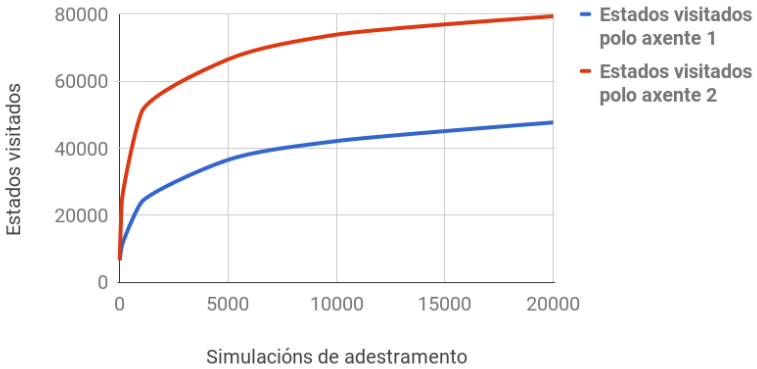
En relación á **validación do axente** realizase nesta figura un análisis que compara o rendemento do **axente 1** entrenado únicamente contra o personaxe basado en reglas contra o **axente 2** entrenado únicamente contra él mesmo.

Dado que para calcular o porcentaxe de victorias se realizaron **100 combates contra o personaxe basado en reglas**, 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 **entrenando o doble de rápido** xa que os dous personajes están aprendendo á vez e combinando a información obtida.

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

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 situaciois nas que non sepa como responder.

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

Axente escollido

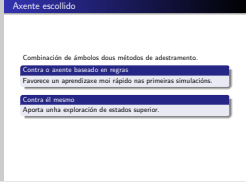
Combinación de ámbolos dous métodos de adestramento.

Contra o axente baseado en regras

Favorece un aprendizaxe moi rápido nas primeiras simulacións.

Contra él mesmo

Aporta unha exploración de estados superior.



Finalmente o axente escollido intenta **combinar o mellor dos dóus métodos de entrenamento**, empezando a pelexar contra o axente basado en reglas 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 comportamiento 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**.

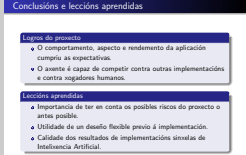
Conclusións e leccións aprendidas

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 cumpliu as espectativas xa que o seu **rendemento é mellor do esperado**. Además o **axente resultou ser muy capaz** incluso contra xogadores relativamente experimentados neste tipo de videoxogos e este en concreto.

Algunhas das **lecciói̇s importantes** que se aprenderon temos:
A importancia de dedicar un momento a considerar que pode **sair mal** en calquera proxecto para lidiar cos problemas antes de que sean demasiado importantes.
As facilidades que proporciona seguir un **diseño 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.

Posibles ampliacións

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

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

Como posibles ampliacións **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 requeriría demasiados cambios. Sería interesante utilizar **Mapas Autoorganizados**, unhas redes neuronales utilizadas para para discretizas 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 complejidad do videoxogo tales como **ataques con rang**o, 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 Rodrígez Mier

21 de xullo de 2017

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 Rodrígez Mier
21 de xullo de 2017

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