Departamento de Informática
Universidad Carlos III de Madrid

Ingeniería en Informática

Aprendizaje Automático

Curso 2017-2018

Tutorial 1

5 de febrero de 2018

- El objetivo de estos ejercicios es familiarizarse con el simulador del juego de Super Mario Bros que se utilizó en la competición Mario AI Championship.
- Es importante seguir el orden establecido para realizar los ejercicios correctamente.

Introducción

Mario AI es un simulador basado en el videojuego de Super Nintendo llamado Super Mario World (1990). Para el desarrollo de estas prácticas se utilizará una versión modificada de Infinite Mario Bros, la cual está programada en Java y sirvió como base para la celebración de varias competiciones de agentes automáticos (bots) para controlar a Mario.

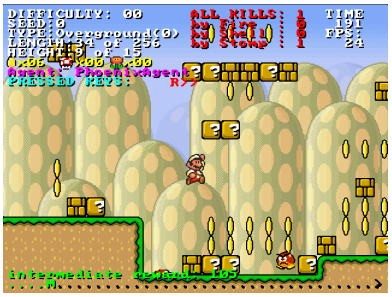

Figura 1: Captura del videojuego *Mario AI*.

1. Puesta en marcha

1. Es necesario tener Java SDK instalado. Se puede utilizar Linux o Windows.
2. Descargar de Aula Global el fichero MarioAI.zip. Contiene el código fuente del simulador.
3. Descomprimir el fichero y compilar el código. Para ello, abrir un terminal, entrar en el directorio `src` y ejecutar el script `compilar`, o bien utilizar el comando `ant` que compilará la totalidad del código utilizando el fichero `build.xml` que se encuentra en el directorio raíz. En este punto podría ser útil importar el código del simulador a un proyecto de Eclipse o NetBeans para mayor comodidad a la hora de programar sobre todo las próximas prácticas.

4. Una vez compilado, ejecutar el simulador escribiendo en el terminal dentro del directorio `src`:

   `./ejecutar.sh human.HumanAgent`

   Aparecerá una ventana en la que podremos jugar (flechas del teclado para mover a Mario, `A` para disparar y correr más, `S` para saltar). Esta clase es un ejemplo de cómo Mario puede ser controlado por un humano. A lo largo del curso también se desarrollarán varios agentes que jugarán solos tomando como base ejemplos de entrenamiento.

   Preguntas

   a) ¿Qué elementos aparecen en la ventana de juego?

   b) ¿Qué cambios se producen al usar los distintos argumentos y atajos de teclado del Anexo 1?

2. Explorando la información disponible

1. Analizar el código y todos los comentarios de la clase `BaselineAgent.java` que se encuentra localizada en `ch.idsia.agents.controllers`. Esta clase servirá de plantilla para todos los agentes que se desarrollarán a lo largo del curso.

2. Ejecutar el anterior agente mediante el siguiente comando:

   `./ejecutar.sh BaselineAgent`

3. Consultar el Anexo 1 para una descripción de los argumentos de ejecución y probar diferentes configuraciones.

   Preguntas:

   a) ¿Qué es un tick de juego?

   b) ¿Para qué le sirven al agente los métodos `getAction()`, `integrateObservation()` y el parámetro `environment`?

   c) ¿Qué información contienen las matrices devueltas por los métodos `getLevelSceneObservationZ()`, `getEnemiesObservationZ()` y `getMergedObservationZZ()`?

   d) ¿Cuál es la posición que ocupa Mario en estas matrices?

   e) A las funciones indicadas en los puntos anteriores se les puede pasar un parámetro para indicar el nivel de detalle de la información devuelta (consultar Anexo 2). ¿Qué implica cambiar el nivel de detalle?

   f) ¿Cuál es el número que codifica las monedas, los obstáculos y los enemigos en el nivel de detalle 1?

3. Creando el fichero de entrenamiento

1. Crear dos nuevos agentes llamados `T1HumanAgent.java` basado en `HumanAgent` y `T1BotAgent.java` basado en `BaselineAgent`. Será necesario renombrar las clases y los constructores de los nuevos agentes.

2. Implementar un método nuevo (en una clase auxiliar o idéntico en ambos agentes) que escriba un mismo fichero llamado `ejemplos.txt` con una línea por cada *tick* de tiempo que contenga:

   - Toda la información del estado en el que se encuentra Mario en ese tick concreto.
   - El valor de las celdas alrededor de Mario devuelta por `getMergedObservationZ()`. Todas las celdas deben ir en la misma línea, sin saltos de línea.
   - El número de monedas recogidas y enemigos eliminados hace 5 ticks (aparte de incluir en la línea también el número de monedas y enemigos del tick actual).

3. Cada línea que se añada al fichero solo debe contener los valores de los datos pedidos separados por comas. No deben tener ningún nombre ni identificador, ni salto de línea a mitad, ni tabuladores. Solo valores y comas en una misma línea por cada tick. En prácticas posteriores se usará el índice sobre esa línea para identificar el tipo de dato y a cada línea la llamaremos un ejemplo de entrenamiento.

4. El fichero solo debe crearse si no existe. Si ya existe solo se deben seguir concatenando debajo más lineas por cada tick.

5. El fichero solo debe abrirse una vez arrancar el agente. Por el contrario, abrirlo y cerrarlo para cada línea que se añada hará que el agente sea muy ineficiente y probablemente el simulador se cerrará.

6. Ejemplo ficticio recortado del formato de línea que hay que ir añadiendo al fichero por cada tick:

   `0,0,0,0,0,0,0,true,false,false,true,34345.0345,warning,goomba,5,-23423.034`

   Pregunta:

   a) Describe brevemente cómo has programado la creación y el rellenado del fichero de registro.

4. Programación de un agente procedural simple

1. Modificar la función `getAction()` de `T1BotAgent.java` para que Mario avance hasta que se encuentre cerca de un obstáculo o enemigo y entonces lo salte.

2. Todos los agentes automáticos que se programen en este curso deben ser mejores que el `BaselineAgent` original, incluido este primero.

   Pregunta:

   a) Describe brevemente cómo has programado el agente procedural.

Normativa de Entrega

1. Se debe entregar antes de la fecha límite indicada en Aula Global:

   a) Memoria en formato .pdf (no se admite .doc, etc.) que contenga las respuestas a las preguntas que se presentan en los ejercicios.

   b) Código fuente de `T1BotAgent.java`

   c) Código fuente de `T1HumanAgent.java`

   d) Código fuente de las posibles clases auxiliares programadas

2. Es obligatorio que la entrega se haga en grupos de 2 personas. No se admiten grupos de 1 ni de más de 2.

3. La entrega debe comprimirse en un fichero .zip (no se admite .rar, .7z, etc.) y entregarse por Aula Global. El nombre del fichero debe tener un formato equivalente al del siguiente ejemplo: `t1-387633-209339.zip` Donde los números son los 6 últimos dígitos del NIA de los alumnos.

4. No se admiten entregas fuera de plazo ni por email.

Referencias

1. Julian Togelius. *The 2009 Mario AI Competition*. http://julian.togelius.com/Togelius2010The.pdf

2. Yuchen Yu. *Diseño e implementación de un agente inteligente Mario A.I.* http://www.it.uc3m.es/jvillena/irc/practicas/11-12/03mem.pdf

3. Descripción del dominio: http://www.marioai.org/

Anexo 1. Opciones de Ejecución

Argumentos para el script de ejecución `ejecutar.sh`:

```
./ejecutar.sh nombre_agente [argumentos]
```

Argumentos:

- nombre_agente: Nombre del agente en `ch.idsia.agents.controllers`
- -ls [número]: Cambia la semilla del mapa actual. Por defecto 0
- -ld [número]: Cambia la dificultad del juego (0-45). Por defecto 0
- -fps [número]: Cambia los fotogramas por segundo. Por defecto 24
- -vis off: Desactivar toda la interfaz gráfica para pruebas más rápidas
- -i on: Mario es invencible
- -lg off: Desactiva los pozos
- -le: Cambia los enemigos. Puede tomar los valores: g: para los goombas. gw: para los goombas voladores. rk: para los koopa rojos. gk: para los koopa verdes. s: para spiky. Por ejemplo, con g aparecerían solo goombas, con gw solo goombas voladores, con gwrkgk aparecerian goombas voladores, koopas rojos y koopas verdes, y así cualquier combinación

Nombres de agentes básicos:

- human.HumanAgent
- BaselineAgent

Atajos de teclado:

- Flechas: Mover a Mario
- A: Disparar/Correr/Coger caparazón
- S: Saltar
- Z: Aumentar zoom ventana
- 8: Quitar límite de fotogramas por segundo
- G: Mostrar matriz
- L: Mostrar etiquetas
- C: Mario siempre en el centro
- F: Volar

Anexo 2. Niveles de Detalle y Códigos

Existen tres niveles de mayor a menor nivel de detalle:

- Nivel 0:
  - Escenario:

Número	Código	Descripción
-20	BREAKABLE_BRICK	Ladrillo simple o con moneda o flor ocultos
-22	UNBREAKABLE_BRICK	Ladrillo irrompible con un signo de interrogación
2	COIN_ANIM	Moneda
-60	BORDER_CANNOT_PASS_THROUGH	Obstáculo a través del que no se puede pasar
-82	CANNON_MUZZLE	
-80	CANNON_TRUNK	Tronco del cañón
-90	FLOWER_POT	Flor
-62	BORDER_HILL	Obstáculo sobre el que se puede saltar y mantenerse encima

  - Enemigos:

Número	Código	Descripción
0	KIND_NONE	Ninguna criatura
-31	KIND_MARIO	Mario
80	KIND_GOOMBA	Goomba
95	KIND_GOOMBA_WINGED	Goomba con alas
82	KIND_RED_KOOPA	Koopa rojo
97	KIND_RED_KOOPA_WINGED	Koopa rojo con alas
81	KIND_GREEN_KOOPA	Koopa verde
96	KIND_GREEN_KOOPA_WINGED	Koopa verde con alas
84	KIND_BULLET_BILL	Bala
93	KIND_SPIKY	Enemigo puntiagudo
99	KIND_SPIKY_WINGED	Enemigo puntiagudo con alas
91	KIND_ENEMY_FLOWER	Flor enemiga
13	KIND_SHELL	Caparazón
2	KIND_MUSHROOM	Champiñón
3	KIND_FIRE_FLOWER	Flor
25	KIND_FIREBALL	Bola de fuego lanzada por Mario
-42	KIND_UNDEF	Tipo de enemigo indefinido

- Nivel 1:
  - Escenario:

Número	Código	Descripción
-24	BRICK	Cualquier tipo de ladrillo
2	COIN_ANIM	Moneda
-60	BORDER_CANNOT_PASS_THROUGH	Obstáculo a través del que no se puede pasar
-62	BORDER_HILL	Obstáculo sobre el que se puede saltar y mantenerse encima
-85	FLOWER_POT_OR_CANNON	Tubería con flor o parte de cañón
0		Ausencia de obstáculos

  - Enemigos:

Número	Código	Descripción
0	Sprite.KIND_NONE	Ausencia de enemigos
3	Sprite.KIND_FIRE_FLOWER	Flor que aumenta poder
2	Sprite.KIND_MUSHROOM	Champiñón que da vida
25	Sprite.KIND_FIREBALL	Bola de fuego lanzada por Mario
80	Sprite.KIND_GOOMBA	Enemigo al que se puede vencer disparándole o saltándole encima (KIND_BULLET_BILL, KIND_GOOMBA, KIND_GOOMBA_WINGED, KIND_GREEN_KOOPA, KIND_GREEN_KOOPA_WINGED, KIND_RED_KOOPA, KIND_RED_KOOPA_WINGED, KIND_SHELL)
93	Sprite.KIND_SPIKY	

- Nivel 2:
  - Escenario

Número	Código	Descripción
-60	BORDER_CANNOT_PASS_THROUGH	Cualquier tipo de obstáculo a través del que no se puede pasar (tubería con planta, cañón, cualquier tipo de ladrillo)
2	COIN_ANIM	Moneda
0		Ausencia de obstáculos, se puede pasar
1		Cualquier otro tipo de obstáculo a través del que no es posible pasar

  - Enemigos

Número	Código	Descripción
1		Cualquier tipo de enemigo
0	Sprite.KIND_NONE	Ausencia de enemigos