IMPLEMENTACIÓN EN JAVASCRIPT PARA EL MODELADO Y DESARROLLO DE SISTEMAS BASADOS EN AGENTES - PARTE I: DEFINICIÓN EN SOFTWARE DE UN AGENTE Y SU ENTORNO

Dr. Gastón Hugo Salazar-Silva M. en C. Abraham Rodríguez Galeotte

Instituto Politécnico Nacional

UPIITA

RESUMEN

Un agente autónomo es un modelo computacional que permite describir comportamientos muy complejos dentro de un entorno. Estos comportamientos se producen a partir de funciones relativamente simples que tienen como dominio una secuencia, o historia, de las percepciones del entorno. El modelado basado en agentes se aplica en inteligencia artificial, robótica, protocolos de comunicación, economía, sociología y aplicaciones web, entre mucho más.

El presente trabajo muestra una librería que permite implementar un sistema de agentes autónomos, de una manera simple, dentro de un entorno gráfico que se ejecuta en un navegador web. Esta librería también puede operar en un sistema embebido. En una segunda entrega se mostrarán dos ejemplos de aplicaciones, por ejemplo, la navegación de múltiples robots en un entorno desconocido.

1. Introducción

Un agente autónomo es un modelo computacional que permite describir comportamientos muy complejos dentro de un entorno. Estos comportamientos se producen a partir de funciones relativamente simples que tienen como dominio una secuencia, o historia, de las percepciones del entorno. El modelado basado en agentes se aplica en inteligencia artificial, robótica, protocolos de comunicación, economía, sociología y aplicaciones web, entre mucho más.

El concepto de programación basada en agentes no es nuevo (Shoham, 1993). En Niazi & Hussain (2011), los autores presentan una revisión del estado del arte del desarrollo de agentes. En Richmond & Romano (2008), los autores presentan una herramienta para el desarrollo gráfico de agentes basado en el uso de una unidad de procesamiento gráfico (GPU, por sus siglas en inglés). El problema es que no se puede portar el código fácilmente entre entornos de programación.

El presente trabajo muestra una librería que permite implementar un sistema de agentes autónomos, de una manera simple, dentro de un entorno gráfico que se ejecuta en un navegador web con ayuda del entorno THREE.js (Williams, 2013). Esta librería también puede operar en un sistema embebido, sin necesidad de disponer de un entorno gráfico. Por las restricciones del formato del documento, sólo se muestra la implementación abstracta de la clase agente. En una segunda entrega se mostrará su aplicación con dos ejemplos. Está librería se ha utilizado para simular la navegación de múltiples robots en un entorno desconocido, así como la implementación de un robot mensajero (Alvarado González, 2016). Tanto la librería como los ejemplos están disponibles en el repositorio del proyecto (Salazar-Silva&Rodríguez Galeotte, 2019).

2. Metdodología

Un agente autónomo es sistema capaz de interactuar con su entorno, y por ello de disponer de una interfaz con su entorno. Ésta se puede componer de sensores y accionadores (efectores) para el caso de un sistema físico. Por otro lado, un agente debe de actuar de forma autónoma, es decir debe capaz de tomar de decisiones de forma independiente a partir de las percepciones que realiza sobre su entorno. Además, en un agente el comportamiento debe ser teleológico, es decir está orientado a alcanzar un objetivo. Un agente puede responder que no, porque contraviene a su objetivo.

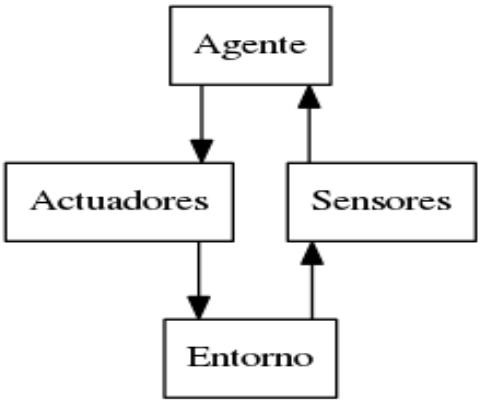

Figura 1. Descripción gráfica de un agente

Un agente se puede describir como una función f,

$$f: p^* \mapsto a$$

donde p denota la percepción realizada por el agente en momento actual, p^* representa una secuencia de percepciones realizada a lo largo del tiempo, es decir una historia de las percepciones, y a representa una acción a desempeñar por el agente. Las variables a y p pueden ser escalares, vectores o incluso funciones.

3. Resultados

A continuación, se presenta el desarrollo de las clases para definir a los agentes y su entorno.

3.1. Prototipo de agente en JavaScript

Para poder implementar los agentes, se deben poder desarrollar abstractamente los sensores, el planificador y los accionadores. En el caso de los sensores, deben ser capaces de detectar la información del entorno en que se encuentran y reaccionar a eventos en el mismo. Los accionadores deben ser capaz de afectar al entorno. Finalmente se debe poder representar el entorno para la planificación; para esto último se aprovecha la librería THREE.js, que no solo permite representar este entorno, sino también visualizarlo en 3D sobre un navegador *web*.

Para empezar, se define la función `Agent()`, que es el constructor de un agente abstracto. Un agente instanciado a partir de este constructor no tendrá realmente funcionalidad pero le permite tener una interfaz. Como se desea tener la capacidad de poder ser visualizado en 3D, se utiliza como prototipo el constructor `Object3D` de la librería THREE.js.

Listado 1. Definición de agente.

```
function Agent(x=0, y=0){
   THREE.Object3D.call(this);
   this.position.x = x;
   this.position.y = y;
}

Agent.prototype = new THREE.Object3D();
```

Las tres primitivas esenciales de un agente son percibir (`sense`), planificar (`plan`) y actuar (`act`) sobre el entorno (`environment`) sobre el cual el agente está operando. Como el constructor es para un objeto abstracto, estas primitivas se definen sin tener una implementación específica.

Listado 2. Métodos abstractos de un agente.

```
Agent.prototype.sense =
   function(environment) {};
Agent.prototype.plan =
   function(environment) {};
Agent.prototype.act =
   function(environment) {};
```

Estos métodos se implementan de manera abstracta porque existe un sinnúmero de formas de implementar estos métodos. Para una implementación específica, el comportamiento se establecerá redefiniendo estos tres métodos.

3.2. Entorno de operación

Como un agente opera sobre un entorno, se define también un constructor para los entornos. Este constructor se define como una instancia de `Scene()`, que es básicamente una colección de objetos de la clase `Object3D()`. Esto permite que los agentes puedan ser visualizados directamente en el navegador.

Listado 3. Definición del entorno.

```
function Environment() {
   THREE.Scene.call(this);
}

Environment.prototype = new THREE.Scene();
```

Después, se implementan tres métodos para las instancias de `Environment()`. En esta primera implementación se consideró un modelo síncrono de operación. Por ello, estos métodos tienen la función de enviar el mensaje a los agentes para que realicen determinadas acciones. Con el método `sense()`, el entorno envía una solicitud a los agentes para que perciban su entorno. Lo mismo ocurre con los métodos `plan()` y `act()`.

Listado 4. Definición de los métodos del entorno.

```
Environment.prototype.sense = function() {
   var c = this.children;
   for ( var i = 0; i < c.length; i++ )
      if (c[i].sense !== undefined)
         c[i].sense(this);
}

Environment.prototype.plan = function() {
   var c = this.children;
   for ( var i = 0; i < c.length; i++ )
      if (c[i].plan !== undefined)
         c[i].plan(this);
}

Environment.prototype.act = function() {
   var c = this.children;
   for ( var i = 0; i < c.length; i++ )
      if (c[i].act !== undefined)
         c[i].act(this);
}
```

Como el entorno es una instancia de `Scene()`, no todos los objetos que pueden existir dentro de éste son agentes. por lo tanto primero se debe determinar si el objeto contiene los métodos adecuados para ejecutarse. Esto permite que el entorno pueda contener objetos estáticos o de otro tipo, con los cuales los agentes pueden interactuar.

En la próxima entrega se mostrarán el desarrollo de dos aplicaciones. La primera (figura 2) es la implementación de un agente reactive, en la forma de una pelota rebotando sobre dos paredes.

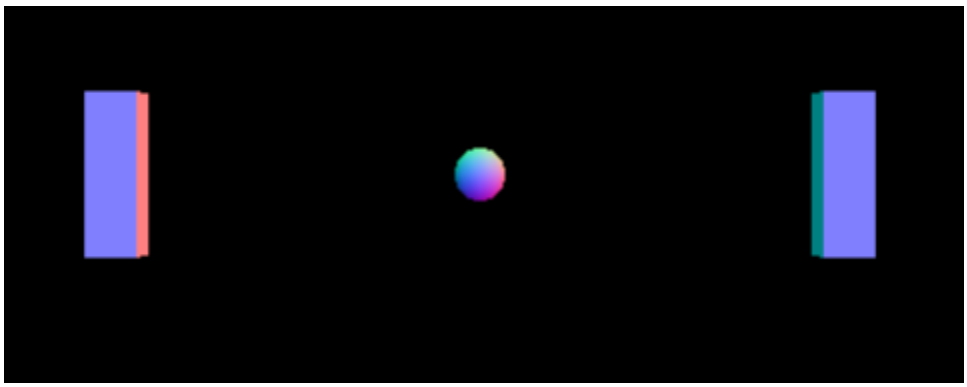

Figura 2. Ejemplo de aplicación de un agente reactivo. En este caso es una pelota que reacciona al tocar una pared.

El siguiente ejemplo (figura 3), será un conjunto de robots operando independientemente, pero logrando un comportamiento complejo por medio de reglas simples.

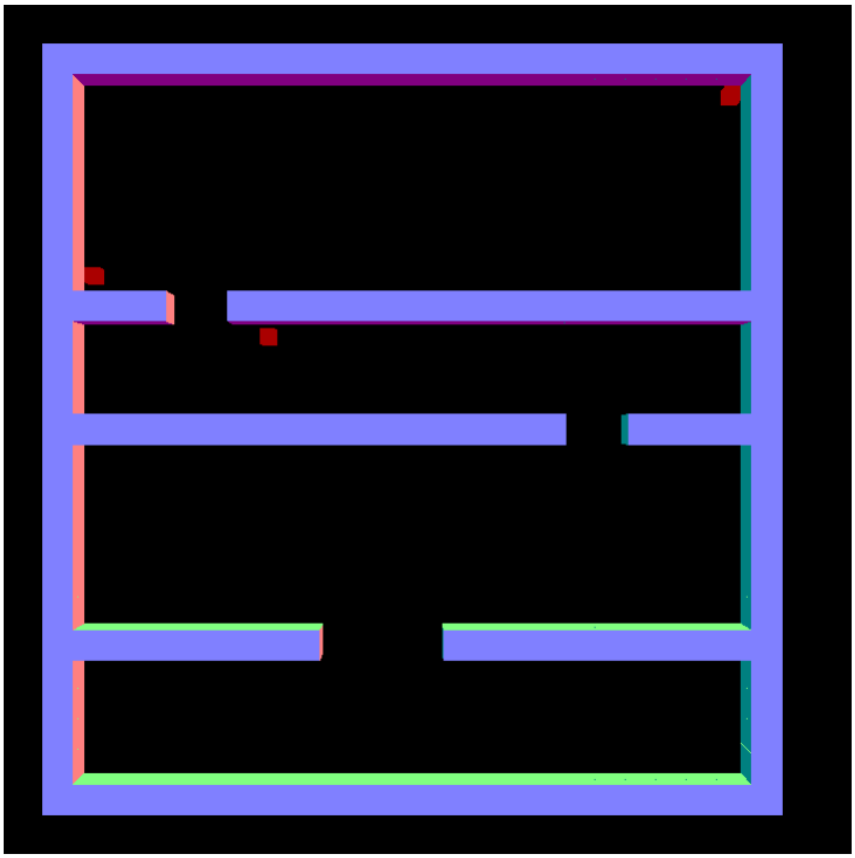

Figura 3. Ejemplo de aplicación de un sistema multi-agentes. En este caso, se modelaron un conjunto de robots que detectan los obstáculos y reaccionan de acuerdo a su programación.

Tanto la librería como los ejemplos están disponibles en el repositorio del proyecto (Salazar-Silva & Rodríguez Galeotte, 2019).

4. Conclusiones

En el presente trabajo se mostró la aplicación de una librería que permite implementar agentes autónomos, de una manera simple, dentro de un entorno gráfico, y que se puede ejecutar dentro del entorno gráfico de un navegador web con ayuda del entorno THREE.js.

Esta librería se puede operar en un sistema embebido, sin necesidad de disponer de un entorno gráfico. Está librería se ha utilizado para simular la navegación de múltiples robots en un entorno desconocido, así como la implementación de un robot mensajero.

Agradecimientos

Los autores desean agradecer al Instituto Politécnico Nacional y la COFAA-IPN por los diversos apoyos y estímulos que han proporcionado para el desarrollo de este trabajo, como el SIBE y el EDD.

Referencias y recursos electrónicos

Alvarado González, A. . (2016). *Prototipo de robot móvil omnidireccional para la repartición de documentos en un área delimitada*. Instituto Politécnico Nacional, UPIITA, Ciudad de México, CDMX.

Niazi, M., & Hussain, A. (2011). *Agent-based computing from multi-agent systems to agent-based models: A visual survey*. Scientometrics. https://doi.org/10.1007/s11192-011-0468-9

Richmond, P., & Romano, D. (2008). *Agent Based GPU, a Real-time 3D Simulation and Interactive Visualisation Framework for Massive Agent Based Modelling on the GPU*. In 1st International Workshop on Super Visualisation (IWSV08). Island of Kos, Greece. Disponible en http://www.dcs.shef.ac.uk/~daniela/Paul_abgpu_IWSV_2008.pdf

Shoham, Y. (1993). *Agent-oriented programming*. Artificial Intelligence, 60(1), 51–92. https://doi.org/10.1016/0004-3702(93)90034-9

Three.js Authors. (2019). *three.js - Javascript 3D library*. Revisado el 1 de marzo de 2019, desde https://threejs.org/

Salazar-Silva, G.H. & Rodríguez Galeotte A. (2019). *Repositorio del código de la librería de agentes*. Disponible en https://github.com/ghsalazar/agentes/tree/IPN-2018.