Transformación de Exámenes de Ambientes E-Learning en Videojuegos Web

Curras, David [1], Dabove, Mariano [1],
Sartorio, Alejandro [2], Vaquero, Marcelo [2], Diamand, Luciano [2]

[1] Grupo de desarrolladores e investigadores del proyecto [2] Grupo de docentes
Universidad Abierta Interamericana (UAI)
Centro de altos estudios en tecnología informática (CAETI)

Resumen. El objetivo de este trabajo es presentar un modelo para la creación de juegos educativos como metáforas representativas de las interfaces de la herramienta de examen utilizada en un eco-sistema de e-learning. Las plataformas de e-learning se componen de Objetos de Aprendizaje que los educadores pueden crear utilizando herramientas de software. Controlando estos objetos es posible establecer métodos de evaluación lúdicos para los estudiantes.

Palabras Clave: videojuego, educación, examen, web, objetos de aprendizaje, e-learning.

1 Introducción

Un juego educativo es un medio a través del cual los usuarios adquieren cierto conocimiento a partir de sus interacciones con el mismo. Los juegos educativos combinan la motivación necesaria para captar la atención de los usuarios, junto con los objetivos de aprendizaje que subyacen a las actividades que se realizan en el juego. Teniendo en cuenta estos aspectos, es posible facilitar este proceso para cada individuo mediante la adaptación de los juegos con los que pueda interactuar ya que ayudan a un desarrollo más amigable y sustentable en su educación y formación cognitiva puesto que a veces resulta tedioso que un estudiante aprenda con un libro [1]. Gracias a esto, nos planteamos los siguientes interrogantes: 1- ¿Podemos ayudar a los jóvenes mediante un videojuego, a que logren sostener en el tiempo y forma un proceso de aprendizaje? 2- ¿Podemos lograr que dichos jóvenes se "apropien" de las nuevas tecnologías? 3- ¿Podemos utilizar una plataforma educativa como mecanismo para que estos aprendizajes puedan ser compartidos con sus pares, y así lograr una mayor integración sin verse afectada la estructura y plan de estudios? [2].

Una de las principales actividades de los jóvenes es jugar. Desde que son pequeños, aprenden desde colorear, contar, sumar o leer ya sea con libros o juegos didácticos. Estas herramientas junto a la informática educativa surgen como estrategia para utilizar correctamente las nuevas tecnologías en forma de aprendizaje. Desde entonces han evolucionado tanto los juegos, como su metodología. De hecho, los juegos virtuales educativos generan un medio didáctico en el que se pueden crear actividades orientadas a la construcción del conocimiento en todas las disciplinas. Se

favorece un aprendizaje lúdico, alternativo y autónomo en el que el estudiante deja de utilizar un cuaderno y lápiz, para interactuar con la PC [3].

Las nuevas tecnologías comenzaron a ser utilizadas por los docentes como nuevos métodos pedagógicos que suponen un soporte interactivo a las "clases magistrales" [4]. Uno de ellos puede ser aprender con actividades lúdicas. La idea no es nueva ya que cuando jugamos construimos un espacio imaginario en el que se ensayan destrezas y capacidades, que quizás no están dentro de nuestras posibilidades inmediatas, pero que podemos utilizarlas en un futuro [3].

Los juegos educativos resultan difíciles de adaptar para los educadores que desean utilizarlos en sus cátedras. Esto se debe a que las actividades que realiza el usuario para lograr los objetivos formativos son seleccionadas previamente por el desarrollador del juego.

El contenido educativo digitalizado, puede organizarse en Objetos de Aprendizaje [12] y contener Metadatos de Objetos de Aprendizaje [13] para describir la estructura de los datos que el juego debe consumir, permitiendo la flexibilidad necesaria para adaptarlos a las cátedras. Las plataformas de e-learning se componen de Objetos de Aprendizaje, que los educadores pueden crear utilizando herramientas de software desarrolladas para tal fin [14].

Las plataformas de e-learning son una herramienta importante para la organización y distribución de contenido educativo, así como para la creación de cursos virtuales de enseñanza a distancia. Muchas de estas plataformas proveen herramientas para la evaluación de los conocimientos del alumno mediante la creación de formularios con preguntas y problemáticas redactadas por el educador. En este contexto, las plataformas de e-learning pueden convertirse en una fuente de contenido para los videojuegos educativos, aportando imágenes, documentos y textos que formen parte del juego, así como preguntas y actividades para la evaluación del aprendizaje. Las bases de datos de las plataformas de e-learning pueden ser utilizadas para parametrizar el videojuego, y así facilitar el desarrollo de un juego capaz de motivar al alumno y permitir la adaptación del mismo a los contenidos que se desea enseñar.

Estos juegos utilizan elementos lúdicos como fuente de conocimiento y motivación para el proceso de aprendizaje, y junto con la flexibilidad de las plataformas de e-learning, las cuales proveen un mecanismo de configuración, evaluación y modificación del contenido, permiten guiar a los usuarios de forma personalizada durante dicho proceso.

El objetivo de este trabajo es presentar un modelo para la creación de juegos educativos como metáforas representativas de las interfaces de la herramienta examen utilizada en un eco-sistema e-learning. Siguiendo los requerimientos descritos en la sección 3, expondremos el diseño e implementación de una interfaz que conecte el contenido generado por una plataforma de e-learning para parametrizar un juego educativo.

2 Estado del Arte

A continuación de describen partes de los trabajos utilizados como punto de partida e inspiración para este trabajo. Estas son las referencias de los seis trabajos procesados:

El trabajo 1 [1] propone un modelo para la presentación y generación de juegos adaptativos, utilizándolos como fuentes de conocimiento y motivación para el proceso de aprendizaje. Toma varios capítulos donde cuenta la necesidad de los juegos educativos adaptativos, desarrolla el modelo teórico de juego, el proceso de adaptación y la parametrización de valores y finaliza con un ejemplo de implementación del modelo.

Los trabajos 2, 3, 4 y 5 [2,4,5,6,10] muestran el diseño y desarrollo de juegos educativos, incluyendo soluciones con herramientas como Invocación Remota de Métodos (IRC), protocolo HTTP, preguntas basadas en la Especificación para la Interoperabilidad de Preguntas y Exámenes [4], y desarrollo para plataformas móviles. También exponen los resultados de implementar los juegos en torneos de matemática y cursos universitarios.

El trabajo 6 expone la evolución del proceso de aprendizaje con los cambios tecnológicos y la forma en que se fue adaptando a los diferentes avances en las TIC. También describe los distintos tipos de plataformas educativas de e-learning, sus características y herramienta.

Tomando como referencia los trabajos anteriormente expuestos, a continuación se presenta una tabla comparativa buscando los ítems más significativos y referenciales para llevar a cabo el proyecto.

Tabla 1. Comparación de proyectos anteriores con el proyecto actual.

TRABAJOS	C1	C2	C3	C4	C5	C6	C7	C8
Generación de Juegos Educativos Adaptativos	x	x	x	x	x			
Diseño y desarrollo de un juego educativo para ordenador sobre enfermedades tropicales y salud internacional	x	x	x	x				
Trivia Mathematica	x	x	x	x				
ProBot: Juego para el aprendizaje de lógica de programación	x	x	x	x	x			
Diseño del juego educativo UNA-Pregunta	x	x	x	x	x	x	x	
Plataformas abiertas de e-learning para el soporte de contenidos educativos abiertos						x		x
Trabajo actual	x	x	x	x	x	x	x	x

C1: Videojuego. C2: Juego Educativo. C3: Diseño. C4: Desarrollo.
C5: Juego Parametrizable. C6: Web. C7: Movil. C8: Plataforma e-learning.

3 Análisis

Definimos los requerimientos para la interfaz de conexión entre la herramienta examen y el videojuego educativo.

Requerimientos Funcionales. 1. La interfaz deberá conocer la estructura de datos de la plataforma de e-learning. 2. El juego deberá identificarse para acceder a los métodos de la interfaz. 3. El juego podrá obtener los datos de parametrización desde la interfaz en cualquier momento que lo requiera. 4. La interfaz deberá reconocer los parámetros que hayan cambiado en la plataforma de e-learning y tenerlos disponibles para el juego de forma instantánea. **Requerimientos No Funcionales.** 1. La interfaz

será web, para poder acceder de forma remota desde diferentes instancias del juego. 2. El juego se identificará, utilizando una clave privada generada por la interfaz, utilizando el estándar JWT (RFC 7519) [11].

4 Statechart como transformador de un examen en un videojuego

Utilizamos el lenguaje de especificación gráfico Statechart creado para modelar el comportamiento de los aspectos dinámicos que permitirán interpretar los enunciados de un examen en una instancia de un juego. Statecharts, fue introducido originalmente con D. Harel [8] como la extensión de máquinas finitas con conceptos de jerarquías y paralelismo. Aprovecharemos su representación visual para implementar una formalización del modelado de los aspectos visuales y convencionales de un examen, sus fundamentos y utilización, para usarlo como traductor de escritura de un videojuego en lenguaje de scripting de fácil utilización en los ambientes Web donde son implementados en instancias de herramientas de exámenes en ambientes e-learning modernos. Luego transformaremos los Statecharts en módulos funcionales para implementar aplicaciones de juegos Web. Usaremos JavaScript [7], un lenguaje liviano y potente, ideal para el tipo de representación que se necesita implementar. Por ejemplo, supongamos que se tiene un sitio e-learning [9] configurado con una herramienta examen en las que hay diferentes preguntas y se deben seleccionar distintas opciones. Simplificando la notación, se puede representar los elementos relacionados como: Pregunta → Opciones ← Respuestas → Caminos. Las flechas indican dependencias entre elementos. Por ejemplo, una Pregunta tiene opciones, las Respuestas son una instancia de las Opciones seleccionadas y al mismo tiempo definen un Camino de navegación hipermedial dentro del sitio. Un camino puede ser el link a la siguiente pregunta, la navegación a otra herramienta, o contenido hipermedial relacionado con el tipo de respuesta seleccionado.

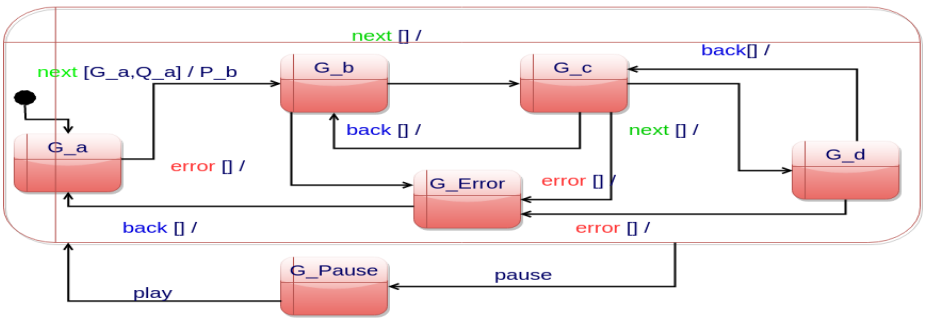

Fig. 2 (a). Statechart denominada Game para especificar los escenarios del juego.

Las estrategias de especificación están basadas en el diseño de tres máquinas principales. La máquina *Game*, está compuesta por estados que representan diferentes escenarios visuales del juego. La figura 1(a) representa estados conceptuales que identifican el inicio de un escenario de juego con el estado *G_a*. Cuando se produce un evento *next* ocurre una transición a otra instancia del escenario del juego. Por

ejemplo, se muestra por pantalla un nuevo obstáculo. De igual manera, con el evento *back* se vuelve al estado anterior. Las pausas en el juego se encuentran especificadas por medio de un super-estado que abarca estados internos (*G_a .. G_d*). Con el evento *pause* se puede transicionar desde cualquiera de estos estados al estado *G_Pause* para representar la acción de pausa en el juego. De la misma manera se especifican las distintas posibilidades de error por medio de la transición de los estados (*G_a .. G_d*) hacia *G_Error*. En este caso, para cada *G_x* (x genérico) sale una flecha para *G_Error*, permitiendo identificar diferentes tipos de errores que se pueden producir dependiendo del estado anterior.

Ahora se explican algunos aspectos conceptuales que se tuvieron en cuenta para la interpretación funcional de esta máquina implementada en módulos JavaScript que se muestra en el fragmento de código 2 de la clase *Game*. En este caso los eventos *next* y *back* son interpretados a través del método *updateGame*. Los fenómenos *winGame* y *endGame* tienen el comportamiento similar a los anteriores, denotan el comienzo del juego y la instancia de finalización en el marco de un escenario específico. La conexión de la máquina con el entorno se produce a través de las interfaces del módulo, representadas por las operaciones que lo definen. Por ejemplo, las operaciones *pauseCondition* y *pauseReceiver* implementan la transición de los estados *G_x* (x: *a..c*) hacia el estado *G_Pause*. Estas operaciones acceden al estado de un *buffer* interno que se utiliza como soporte para implementar las funcionalidades del entorno de las máquinas.

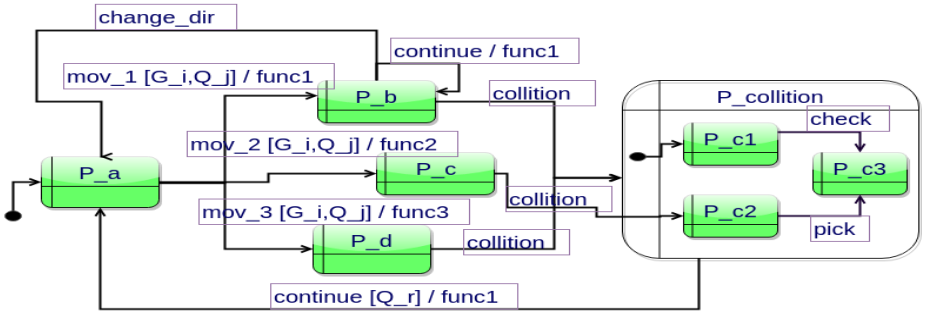

Fig. 1 (b). Statechart denominada Player para especificar las funcionalidades del juego.

La máquina *Player* se encarga de la representación funcional del juego. La figura 1 representa los movimientos que puede hacer el personaje principal del juego. En este caso se especifican cada uno de esos movimientos a través del evento *mov_i* (i: 1…N), para N casos posibles. Por ejemplo: giros, avances, saltos, cambios de velocidad, selección de caminos en lugares bifurcados, etc. De la misma forma se resuelven las funcionalidades de las colisiones. Cuando ocurre una colisión, la máquina Player adquiere conocimiento de este fenómeno si puede capturar el evento *collition*. En ese caso, se accede a una nueva máquina (*P_collition*) que se encargará de especificar el comportamiento del sistema cuando ocurra una colisión. Hay accesos desde tres tipos de estados diferentes. En la figura 1(b) hay tres estados representativos de estas posibilidades. El estado *P_b* captura una colisión y pasa la máquina de colisiones que automáticamente pasa al estado *P_c1*. Lo mismo ocurre si

desde *P_d* se captura el evento *collition*. El estado inicial de la super-máquina *P_collition* se encuentra referenciado con el círculo negro. Luego, esta máquina se encargará de especificar todas las funcionalidades necesarias para el manejo de la lógica de colisión. En este caso, a través de los eventos *check* y *pick* que permitirán sincronizar el comportamiento con la máquina *Question* encargada de especificar las estructuras y propósitos de las preguntas del examen.

Siguiendo con la misma estrategia de la máquina anterior, analizaremos la implementación realizada en JavaScript a través del módulo *Player* de la figura. Este módulo configura tres tipos de eventos: *movePlayer*, para el control de los movimientos e implementación de los eventos *mov_i* de la maquina *Player*; *checkCollision,* que implementa el evento *collition;* y *talk* que se utilizará para representar los eventos *check* y *pick* dentro de la máquina *Player*.

Por último, en la Figura 1(c) muestra las principales componentes de nuestra propuesta de Statechart para la especificación de la estructura del examen, respetando los conceptos anteriormente mencionados. Partiendo del estado inicial Q de la máquina Question hay dos posibles caminos de transición, uno para el control de una respuesta a través del evento *check* y otro para el evento *pick*. El evento *check* espera la aprobación de la validación con el evento *valid* y luego puede derivar en tres posibles estados. Hacia el estado *Predicate* con el evento *pred*, al estado *Condition* con el estado *cond* y el estado *Receive* con el evento *rec*. El otro camino comienza con el evento *pick*, iniciando la selección de una consigna para que resuelva el usuario. Pasa por el estado intermedio *Check* a través de *one* y luego hacia *Condition* y *Receive* con los eventos *cond* y *rec*, respectivamente.

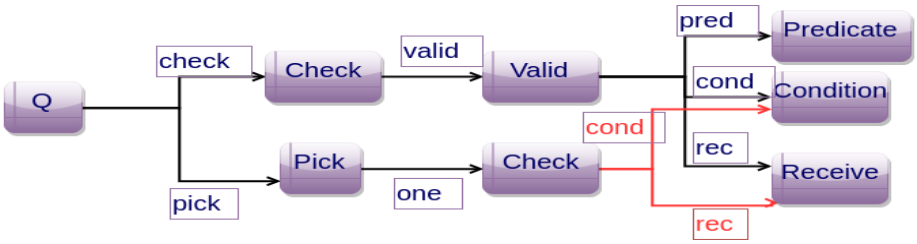

Fig. 1 (c). Statechart denominada Question para especificar el cuestionario del examen.

En cuanto a la implementación en módulo JavaScript de este Statechart, el método *checkValidPredicate,* que representa los eventos *check*, *valid*, *pred* y *pickOnePredicate,* se encarga de representar los eventos *pick* y *one*. Al igual que en los módulos anteriores, se encuentran operaciones como *pickOneCondition* encargada de conectar con el entorno, en este caso el *buffer* interno, con el propósito de verificar condiciones de posibles transiciones de estados. Por otro lado, la operación *pickOneReceiver* tendrá la responsabilidad de referenciar la pregunta adecuada y actualizar el *score* del usuario participante.

Este caso se utiliza para modelar la representación de un examen como un videojuego. La figura 1 representa tres máquinas en paralelo. Esta representación determina unos de los principales conceptos que se pretende exponer en este trabajo. Una vez entendido el propósito, se conformará una estructura eficaz para aplicar

cualquier tipo de examen (selecciones múltiples, asociar, textos libres, mensajes de aviso según respuestas, etc.) para cualquier videojuego que se pueda expresar en el grupo de máquinas de la figura 1.

Por cuestiones de espacio, concentramos en esta sección el principal concepto que se quiere trasmitir en este trabajo como una propuesta de transformación del formato de formulario web como herramienta de examen, hacia un videojuego que contenga el espíritu y los objetivos de esa instancia de evaluación.

5 Diseño

Arquitectura de la aplicación. El juego educativo obtiene los datos necesarios para adaptar su contenido a través de la interfaz de conexión web, la cual posee métodos públicos conocidos por el juego. La interfaz tiene acceso a la base de datos de la plataforma de e-learning con permisos de solo lectura, y consulta los datos cada vez que un juego se conecta para verificar si hubo modificaciones. La plataforma de e-learning se encuentra conectada a la base de datos de forma local y puede realizar modificaciones en cualquier momento. La figura 2(a) muestra de forma gráfica la arquitectura de la aplicación.

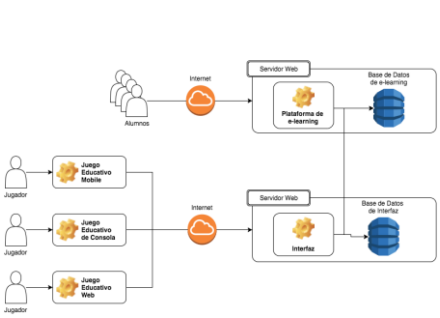

Fig. 2 (a). Arquitectura de la aplicación.

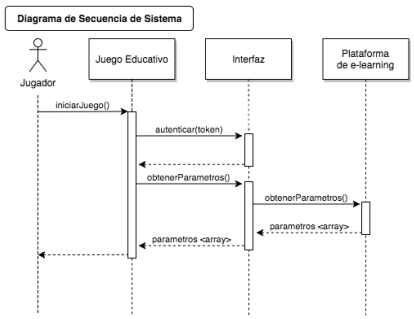

Fig. 2 (b). Diag. Secuencia del Sistema (DSS).

El Diagrama de Secuencias de Sistema expuesto en la figura 2(b) muestra las interacciones entre el jugador, el juego educativo, la interfaz de conexión y la plataforma de e-learning. También se pueden ver la firma de los métodos expuestos por la interfaz.

6 Implementación

Se ha creado un juego educativo web, el cual utiliza una interfaz de conexión para acceder a la base de datos de una plataforma de e-learning para adaptar su contenido a diferentes cursos.

Utilizando el lenguaje JavaScript [7], desarrollamos una abstracción de Statecharts [8] para implementar la interfaz, y un juego gráfico de plataformas, con preguntas y respuestas que se seleccionan de forma dinámica dependiendo del nivel y la

experiencia del jugador. De esta manera concretamos los conceptos desarrollados en la sección 4.

```
game$ ls js/abstract/
condition.js            machine.js              receiver.js             state.js
emitter.js              predicate.js            state-buffer.js         super-machine.js
game$ ls js/game
game.js         player.js       questions.js
```

Fig. 3. Estructura de archivos.

La figura 3 muestra la estructura de archivos creada, donde la carpeta *abstract* contiene una representación abstracta del modelo Statecharts, y la carpeta *game* posee las clases necesarias para crear el juego.

Cada una de las clases dentro de *game* extiende de la clase *Machine* (Máquina), la cual posee un estado de clase *State* y una lista de predicados que son instancias de la clase *Predicate*.

Fragmento de código 1. JavaScript de las clases Machine y Condition.

```
var Machine = (function() {
 'use strict';
 function Machine(state, pred) {
  var t = this;
  if (!(t instanceof Machine)) {
   return new Machine(state);
  }
  this.state = state;
  this.pred = pred;
 }
 return Machine;
}());
```

```
var Condition = (function() {
 'use strict';
 function Condition(cFn) {
  var t = this;
  if (!(t instanceof Condition)) {
   return new Condition(cFn);
  }
 }
 Condition.prototype.evaluate = cFn;
 return Condition;
}());
```

El predicado es responsable de realizar las transiciones entre estados de la máquina a la que pertenece, utilizando las instancias de *Emitter* y *Receiver*. El receptor de eventos ejecutará las acciones necesarias para transicionar al siguiente estado, solo cuando la condición (*Condition*) sea verdadera.

En el Fragmento de código 2, se puede ver que la clase *Game* extiende de *Machine (línea 50)*, y en su constructor se crean las instancias de *State (línea 10)* y *Predicate (líneas 43 y 44)* necesarias para iniciar y finalizar el juego. El código final del constructor de la clase Game es más extenso y posee 8 predicados para poder transicionar entre los diferentes estados de la máquina. De manera similar, las clases *Player* y *Question* extienden de *Machine*, y transicionan sus estados mediante predicados instanciados de la clase Predicate.

Cada máquina puede emitir eventos que serán recibidos por ellas mismas, o por otras máquinas. Para ello se utilizó la clase *Event* de JavaScript como un *EventManager*, donde todos los eventos emitidos son capturados por *Event*, y enviados al receptor que esté esperando escucharlo.

Cuando el jugador se mueve, la máquina *Player* envía el evento *movePlayer*, el cual es capturado por la instancia *Receiver* del predicado *movePlayerPredicate*. En este caso, la máquina *Player* posee los predicados para resolver la transición del

estado por sí misma, es decir, que emite y recibe el evento utilizando sus propios predicados.

Fragmento de código 2. JavaScript de la clase Game.

```
var Game = (function() {
 'use strict';
 function Game(args) {
  if (!(this instanceof Game)) { return new Game(args); }
  this.gameState = new State({ state: {}, context: {} };
  var startC = function() {
   return StateBuffer.gameState === 'idle';
  };
  var startR = function() {
   StateBuffer.gameState = 'playing';
   var rows = 10;
   var columns = 5;
   var blockWidth = (this.gameState.width / rows);
   var blockHeight = 30;
   for (i = 0; i < columns; i++) {
    for (j = 0; j < rows; j++) {
     id = 'block'+j+i;
     block.create(id);
     this.gameState.dynamicList.push(block.list[id]);
    }
   }
   window.onkeydown = keyboard.press;
   window.onkeyup = keyboard.release;
   setInterval(game.predicates.updateGame, 1000/60);
  };
  var updateC = function() {
   return StateBuffer.gameState === 'playing';
  };
  var updateR = function() {
   for (i = 0; i < this.gameState.staticList.length; i++) {
    this.gameState.staticList[i].render();
   }
   for (i = 0; i < this.gameState.dynamicList.length; i++) {
    this.gameState.dynamicList[i].update();
    this.gameState.dynamicList[i].render();
   }
   if(!Object.keys(block.list).length) {
    game.predicates.winGame();
   }
  };
  var gStart = new Predicate('startGame', startR, startC);
  var gUpdate = new Predicate('updateGame', updateR, updateC);
  var gMachine = new Machine(gameState, {
   startGame: gStart,
   updateGame: gUpdate,
  });
  this = _.extend(this, gMachine);
 }
 return Game;
}());
```

Fragmento de código 3. JavaScript de la clase Predicate.

```
var Predicate = (function() {
 'use strict';
 function Predicate(evt, evtFn, cFn) {
  if (!(this instanceof Predicate)) {
   return new Predicate(evt, evtFn);
  }
  this.receiver = new Receiver(evt, evtFn, cFn);
  this.emitter = new Emitter(new Event(evt));
 }
 return Predicate;
}());
```

La máquina *Player* también posee predicados que emiten eventos para detectar si el jugador está en un sector del juego donde es necesario mostrar una nueva pregunta, en cuyo caso se disparará el evento *pickQuestion*, que será tomado por la máquina *Question*, la cual, luego de hacer las comprobaciones necesarias, enviará el evento *showQuestion* pasando como parámetro el texto de la pregunta. Aquí la máquina *Game* lo recibe y utiliza los predicados necesarios para cambiar su estado y así mostrar la pregunta correspondiente en pantalla.

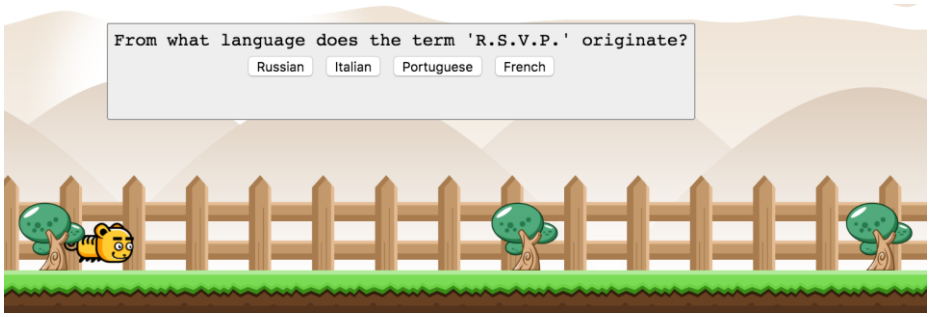

Fig. 4. Juego educativo mostrando una pregunta con sus posibles respuestas.

7 Adaptaciones

El modelo presentado es fácilmente adaptable a diferentes tipos de juegos, con nuevos desafíos para permitir evaluar al estudiante desde enfoques más dinámicos que un simple cuestionario.

Adaptación del tipo de juego. Si se pretende realizar un nuevo juego basándose en el modelo propuesto, podemos optar por modificar la apariencia gráfica y parte de la lógica, pero manteniendo la modalidad de evaluación mediante preguntas y respuestas. La mecánica del juego se encuentra definida dentro de las máquinas *Game* y *Player*. Modificando el código dentro de los predicados *startReceiver* y *updateReceiver* de la clase *Game*, y dentro del predicado *moveReceiver* de la clase

Player, se puede lograr un juego con apariencia y jugabilidad diferentes pero con el mismo sistema de preguntas y respuestas de opciones múltiples.

Adaptación del método de evaluación. También se pueden realizar modificaciones al tipo de examen. Las plataformas educativas permiten crear Objetos de Aprendizaje para evaluar al estudiante con diferentes métodos, como pueden ser las preguntas con opciones múltiples, verdadero o falso, y de redacción.

Fig. 5. Juego educativo con modificaciones en las clases Game y Player.

Para crear un juego que responda a otro tipo de método de evaluación, es necesario modificar la implementación de la máquina *Questions,* haciendo que el predicado *pickQuestion* provea el desafío acorde al nivel del jugador; y dentro del predicado *evaluateAnswer* se deben establecer las reglas que determinen si la respuesta es correcta.

8 Conclusiones y trabajo futuro

El modelo creado permite describir y generar juegos educativos para evaluar los conocimientos adquiridos por el estudiante, dependiendo del método de evaluación y los contenidos generados por el docente dentro de una plataforma de e-learning.

Este modelo ha sido aplicado a un juego de preguntas y respuestas con múltiples opciones, y en un futuro próximo se utilizará en un contexto real para obtener datos que permitan determinar bajo qué condiciones se obtienen los mejores resultados.

Se está considerando ampliar el modelo para incluir nuevos métodos de evaluación y enseñanza, aprovechando las herramientas que proveen las plataformas de e-learning mediante Objetos de Aprendizaje.

Entre los objetivos a largo plazo, se encuentra la creación de un formalismo para la descripción de los componentes de los juegos educativos y su relación con los contenidos y métodos de evaluación.

Referencias

1. Carro, R.M., Breda, A.M., Castillo, G., Bajuelos, A.L.: Generación de Juegos Educativos Adaptativos. (2002)
2. Giménez, P.C.; Arévalo, C.P.; Martínez Herráiz, J.J: Diseño y desarrollo de un juego educativo para ordenador sobre enfermedades tropicales y salud internacional: una herramienta docente más de apoyo al profesor universitario. (2011)
3. León, M.: Los métodos de enseñanza aprendizaje y la informática. Facultad de ciencias médicas, Las Tunas, Cuba. (2004)
4. Morales, R., Gallegos, E.P.: Trivia Mathematica: Una Experiencia de Desarrollo con Software Libre. (2012)
5. Moreno, J., Montaño E.A.: ProBot: Juego para el aprendizaje de lógica de programación. (2009)
6. Ramírez, E.R, Sandí, H.R.: Diseño del juego educativo UNA-Pregunta: promoviendo la participación y el aprendizaje en línea. (2014)
7. What is JavaScript? https://developer.mozilla.org//docs/Web/JavaScript/About_JavaScript
8. Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programmig 8 231-274. North-Holland. (1987)
9. Sartorio, A., & Cristiá, M.: Primera aproximación al diseño e implementación de los DHD. In XXXIV Congreso Latinoamericano de Informática. (2008)
10. Boneu, J.M.: Plataformas abiertas de e-learning para el soporte de contenidos educativos abiertos. (2007)
11. JSON Web Token (JWT). RFC 7519. https://tools.ietf.org/html/rfc7519
12. IEEE 1484.12.1 – 2002 Standard for Learning Object Metadata. http://129.115.100.158/txlor/docs/IEEE_LOM_1484_12_1_v1_Final_Draft.pdf
13. Metadatos de Objetos de Aprendizaje. http://www.asociacionelearning.com/contenidos/lom-learning-object-metadata.asp
14. Cohen, E. B., Nycz, M.: Learning Objects and E-Learning: an Informing Science Perspective. (2006)