Como citar este artículo: M. A. López, E. V. Duarte, E. C. Gutiérrez and A. P. Valderrama, "Teaching Based on Ludic Environments for the First Session of Computer Programming - Experience With Digital Natives," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 14, no. 2, pp. 34-42, May 2019, doi: 10.1109/RITA.2019.2922847.

Enseñanza Basada en Entornos Lúdicos para la Primera Sesión de Programación de Computadoras – Experiencia con Nativos Digitales

Marco Aedo López, Elizabeth Vidal Duarte, Eveling Castro Gutiérrez y Alfredo Paz Valderrama

***Title*— Teaching Based on Ludic Environments for the First Session of Computer Programming - Experience with Digital Natives**

***Abstract*— This article describes a teaching experience based on ludic environments: videogame and 3D animation environment, to provide students with a motivating environment for the first computer programming course. One problem that exists in many Latin American countries is that educators who teach the basic programming course find that for most students, computer programming is a completely new subject. We found that an approach based on ludic environments that takes into account the styles of visual and kinesthetic learning to teach basic programming concepts is more effective and motivating.**

***Index Terms*— Fundamentals of programming, Learn to program, Motivation in teaching, Ludic environments, Pedagogical tools**

I. Introducción

HAY experiencias que han mostrado que la naturaleza visual y lúdica de los videojuegos son muy útiles para el aprendizaje de los estudiantes [1],[2],[3],[4]. Incluso hay una implementación de esta aproximación en Karel the Robot [5], el cual usa clases predefinidas para introducir los conceptos fundamentales de programación.

El curso de introducción a la programación de computadoras, que en la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa – Perú [6] se denomina Fundamentos de Programación 1 (FP1), se empezó a enseñar como parte del Plan de Estudios 2013 y continúa conformando el nuevo Plan de Estudios 2017, está centrado en el lenguaje de programación Java [7] y busca enseñar los conceptos básicos de la programación de computadoras.

Debemos recalcar que el presente artículo es la extensión de un trabajo originalmente presentado en 15th Latin American and Caribbean Conference for Engineering and Technology (LACCEI 2017) [8] y que fue elegido por IEEE-RITA, por su calidad, para ser extendido; en este artículo extendido incluimos, además, los resultados actualizados de los 2 últimos años de evaluación y detallamos nuevos ejercicios aplicados que nos permitieron alcanzar nuestra meta de enseñar más conceptos básicos usando este enfoque lúdico.

Durante la primera sesión del curso, a lo largo de los años, se nota la existencia de un grupo de estudiantes bastante heterogéneo, tanto en conocimientos como en motivaciones. Así, la Tabla I muestra los resultados de los 6 años en los que se viene dictando el curso, a la pregunta realizada el primer día clases de "¿cuántos de ustedes tienen alguna experiencia en programación de computadoras?"

En la Tabla II se hace hincapié en diferentes niveles de experiencia en programación y se obtiene resultados más puntuales, descubriendo que al menos algunos conceptos simples, sin llegar al nivel de la creación de programas, son poseídos por los estudiantes del curso.

Sin embargo, en la Tabla III, ante la pregunta "¿quiénes utilizan videojuegos?", las respuestas son casi lo opuesto a las respuestas obtenidas para la primera pregunta.

TABLA I.
Respuestas a Cuestionamiento Inicial

Año	Si	No
2013	5%	95%
2014	6%	94%
2015	5%	95%
2016	7%	93%
2017	5%	95%
2018	4%	96%

M. Aedo is with Escuela Profesional de Ingeniería de Sistemas, Universidad Nacional de San Agustín de Arequipa – Perú (email: maedol@unsa.edu.pe) (https://orcid.org/0000-0001-6608-804X)

E. Vidal is with Escuela Profesional de Ingeniería de Sistemas, Universidad Nacional de San Agustín de Arequipa – Perú (email: evidald@unsa.edu.pe) (https://orcid.org/0000-0002-8367-9439)

E. Castro is with Escuela Profesional de Ingeniería de Sistemas, Universidad Nacional de San Agustín de Arequipa – Perú (email: ecastro@unsa.edu.pe) (https://orcid.org/0000-0002-0203-041X)

A. Paz is with Escuela Profesional de Ingeniería de Sistemas, Universidad Nacional de San Agustín de Arequipa – Perú (email: apazv@unsa.edu.pe)

TABLA II.
RESPUESTAS A CUESTIONAMIENTO INICIAL DETALLADO

Año	Ninguna	Crean fórmula en hoja de cálculo	Programa simple (tipo suma de 2 números)	Programa simple (tipo mayor de 2 números)	Programa medio (tipo ordenar arreglo de números)
2013	40%	55%	3%	2%	0%
2014	45%	49%	4%	2%	0%
2015	35%	60%	2%	2%	1%
2016	40%	53%	3%	3%	1%
2017	39%	57%	3%	1%	0%
2018	40%	55%	3%	1%	1%

TABLA III.
RESPUESTAS A UTILIZACIÓN DE VIDEOJUEGOS

Año	Si	No
2013	90%	10%
2014	92%	8%
2015	95%	5%
2016	97%	3%
2017	97%	3%
2018	96%	4%

La heterogeneidad mostrada en el grupo de estudiantes es tratada de homogeneizar basada en los estilos de aprendizaje de los nativos digitales y usando herramientas que ellos conocen muy bien y utilizan cotidianamente: videojuegos y otros entornos lúdicos, tales como entornos de animación en 3D.

En este artículo presentamos nuestra experiencia en el uso de las herramientas lúdicas: videojuego Lightbot [9],[10] y complementado con un entorno de animación en 3D llamado Alice [11].

El resto del artículo está organizado de la siguiente manera. En la sección II se describe el primer curso de programación de nuestro plan de estudios. En la sección III se muestra los estilos de aprendizaje de los nativos digitales en nuestro entorno en particular. En la sección IV se describe brevemente las herramientas que utilizamos, Lightbot y Alice. En la sección V se muestra el diseño de la propuesta de la primera sesión del curso, basándonos en el estilo de aprendizaje de nuestros estudiantes y usando las herramientas antes descritas. En la sección VI se muestran los resultados obtenidos del rendimiento de los estudiantes, donde se compara su rendimiento cuando utilizamos un enfoque tradicional de enseñanza con un enfoque lúdico de enseñanza. Finalmente se presentan nuestras conclusiones.

II. EL PRIMER CURSO DE PROGRAMACIÓN

A. *Generalidades*

El primer curso de programación en los Planes de Estudio 2013 y 2017 es denominado Fundamentos de Programación 1, es dictado en el primer semestre de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa - Perú. El curso tiene una duración de 17 semanas y tiene 8 horas semanales (2 horas teóricas, 2 horas prácticas y 4 horas de laboratorio). Como lenguaje de programación utiliza Java, pero desde el 2015 nos apoyamos con herramientas lúdicas para su enseñanza.

B. *Competencias*

Nuestro plan de estudios se realizó tomando en cuenta las competencias indicadas en ABET (Accreditation Board for Engineering and Technology) [12]. Donde se destaca la importancia de las habilidades profesionales y las habilidades de conciencia, además del desarrollo de las habilidades técnicas para lograr excelencia en la formación de ingenieros.

En la Tabla IV y Tabla V se muestran las competencias generales y específicas del curso de Fundamentos de Programación 1.

C. *Contenidos Conceptuales*

Los contenidos conceptuales del curso se resumen en la Tabla VI.

TABLA IV.
COMPETENCIAS GENERALES DEL CURSO FP1

c. Habilidad para diseñar un sistema, componente o proceso que satisfaga necesidades dentro de restricciones realistas tales como economía, medio ambiente, sociales, políticas, éticas, salud y de seguridad, manufacturación y sostenibilidad. k. La capacidad de utilizar las técnicas, habilidades y herramientas modernas de ingeniería y computación necesarias para la práctica de la ingeniería del software. m. Habilidad para aplicar apropiadamente matemáticas discretas, probabilidad y estadísticas, y tópicos relevantes en computación y disciplinas de apoyo a sistemas de software complejo

TABLA V.
COMPETENCIAS ESPECÍFICAS DEL CURSO FP1

1. Identifica, establece e integra los diferentes conceptos de programación reconociendo los componentes y características de un algoritmo. 2. Elabora, crea y codifica algoritmos para la solución de problemas reales en un lenguaje de programación. 3. Aplica, codifica y ejecuta sentencias de selección y de repetición apropiadas para la elaboración de programas. 4. Introduce, analiza y utiliza el concepto de métodos reconociendo su importancia en la programación. 5. Introduce, analiza y aplica los conceptos de arreglo estándar y ArrayList reconociendo su importancia en la solución de ciertos problemas. 6. Analiza, conecta e integra los conceptos de clase, objeto, atributo y método reconociendo su importancia en la programación 7. Concibe, analiza y utiliza las excepciones como una condición anormal de ejecución de un programa.

TABLA VI.
CONTENIDOS CONCEPTUALES DEL CURSO FP1

Contenidos Conceptuales
Unidad I. Conceptos Generales
Identifica los diferentes conceptos de programación reconociendo los componentes y características de un algoritmo. Elabora algoritmos para la solución de problemas reales. **Introducción** • Conceptos generales. Una aproximación con Lightbot y Alice • Algoritmo • Paradigmas de programación • Lenguajes de programación • Componentes de un algoritmo • Característica de un algoritmo • Compiladores vs intérpretes • IDEs • Pasos a seguir para crear programas • Pseudocódigo y diagramas de flujo **Manipulación de Expresiones** • Definición de programa • Valores • Variables • Introducción a tipos de datos • Expresiones, asignaciones y operadores

Iniciando con Java.
- Entrada/Salida estándar
- Primer programa en Java
- Componentes de un programa
- Convenciones estándar para nombres
- Plantilla de programas Java simples

Creación y Declaración de Objetos
- Conceptos básicos de la orientación a objetos: clase, objeto, atributo y método
- Estado de la memoria de un programa
- Envío de mensajes
- Utilizando clases estándard
  - JOptionPane
  - String
  - Date
  - SimpleDateFormat

Tipos de Datos Numéricos
- Datos numéricos.
- Expresiones aritméticas en Java.
- Evaluación de expresiones aritméticas utilizando reglas de precedencia.
- Casting implícito y explícito
- Constantes
- Asignación de memoria para los objetos y tipos de datos primitivos.
- Convertir valores String en datos numéricos.
- Standard Output/ Standard Input
- Expresiones matemáticas, utilizando la clase Math

Unidad II. Sentencias de Selección y Repetición

Aplica las sentencias de selección y de repetición apropiadas para la elaboración de programas.

Sentencias de Selección
- Sentencia **if**
- Expresiones Booleanas y variables
- Sentencia **if- anidado**
- Sentencia **if- else if**
- Comparación entre objetos
- Sentencia **switch**

Sentencias de Repetición
- Sentencia **while**
- Errores comunes al escribir sentencias de repetición
- Sentencia **do–while**
- Bucles y Control de repetición
- Dialogo de confirmación
- Sentencia **for**
- Sentencia **for-anidado**
- Bucles anidados
- Generación aleatoria de números

Unidad III. Métodos en Java

Introduce el concepto de métodos y/o funciones reconociendo su importancia en la programación

Métodos y Construcción Modular de Programas
- Construcción modular
- Declaración de métodos propios
- Invocación de métodos
- Ventajas del uso de métodos en la Ingeniería de Software
- Alcance de las variables en métodos
- Promoción de los argumentos
- Métodos de la clase Math
- Recursión
- Sobrecarga de métodos

Unidad IV. Arreglo Estándar y ArrayList

Introduce el concepto de arreglo estándar (arrays) y ArrayList reconociendo su importancia en la programación

Arreglos estándar y ArrayList
- Fundamentos de arreglos
- Declaración de arreglos
- Creación de arreglos
- Inicialización de elementos
- Valores por defecto en arreglos
- Propiedad de longitud del arreglo
- Arreglos parcialmente llenados
- Copiando un arreglo
- Histogramas
- Búsquedas en arreglos
- Ordenamiento en arreglos
- Ordenamiento por selección
- Arreglos Bi-Dimensionales
- Arreglos de Objetos

La Clase ArrayList
- Creación de objetos ArrayList
- Agregando elementos
- Accediendo a elementos en un ArrayList
- Actualizando un objeto ArrayList
- Métodos adicionales del ArrayList
- Impresión o concatenación de un ArrayList
- Primitivas de ordenamientos en un ArrayList
- Objetos ArrayList versus arreglos estándar

Unidad V. Programación Orientada a Objetos y Manejo de Excepciones

Analiza el concepto de herencia y polimorfismo interpretando el paradigma orientado a objetos
Concibe el concepto de excepciones como una condición anormal de ejecución de un programa

POO, Herencia y Polimorfismo
- Jerarquía de clases en Java
- List, LinkedList y ArrayList

Manejo de Excepciones
- Utilizando bloques try-catch para manejar llamada a métodos
- NumberFormatException
- Bloque try-catch - más detalles
- Dos tipos de excepciones - checked y unchecked
- Utilizando documentación API para el manejo de excepciones
- Cuando un bloque try-catch lanza diferentes tipos de excepciones
- La clase Exception y el método getMessage
- Multiples bloques catch
- Comprendiendo los mensajes de excepción

D. Cursos de programación en los Planes de Estudio 2013 y 2017

En la Fig. 1 se muestra la línea de programación en ambos planes de estudio, donde se muestra lo importante que es el curso Fundamentos de Programación 1 en toda la carrera profesional. Observar que el curso es prerrequisito de Fundamentos de Programación 2 (FP2) y de Programación Web 1 (PW1), los que a su vez abren ramas importantes de cursos fundamentales del plan de estudios.

Se muestra la cantidad de créditos, cantidad de horas teóricas, prácticas y de laboratorio respectivamente.

E. Forma de Evaluación del Curso

El curso se evalúa de la siguiente manera:

- Examen: netamente práctico donde solucionan problemas aplicando los conocimientos teóricos enseñados. Rinden 2 exámenes a lo largo del curso.
- Evaluación Continua: constituida por prácticas en clase individuales o grupales, tareas para la casa, participación en la solución de problemas en clase, trabajos de investigación, pequeños proyectos de desarrollo de software y lectura de artículos.
- Evaluación de laboratorio: prácticas, tareas y proyecto final.

El promedio final se calcula de la siguiente manera:

$$PF=(EC*0.20+IE*0.30+IIE*0.50)*0.70+Laboratorio*0.30$$

PF = Promedio Final
EC = Evaluación Continua
IE = I Examen
IIE = II Examen

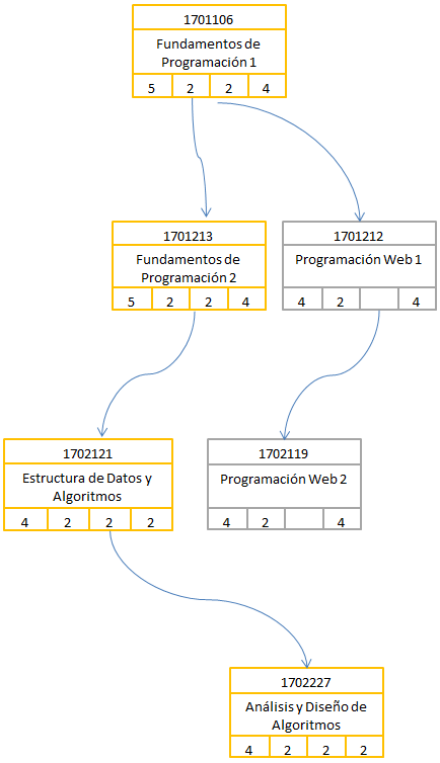

Fig. 1. Cursos de línea de Programación en los Planes de Estudio 2013 y 2017

III. Estilo de Aprendizaje de los Nativos Digitales en Nuestra Realidad

Actualmente, los estudiantes que cursan estudios universitarios son diferentes a los que lo hacían hace una década o más, la pregunta es: "¿está nuestro sistema educativo actual diseñado para este tipo de estudiantes?"

Algunos los llaman "Nativos Digitales" y otros incluso "Generación Z", individuos que nacieron junto al internet y cuya filosofía de vida y forma de aprender difieren de generaciones anteriores.

A. Los nativos digitales y cómo aprenden

El creador del término "Nativo Digital", Prensky [13], observa que los estudiantes actuales son todos "nativos hablantes" del lenguaje digital de las computadoras, videojuegos y del internet.

Se puede decir que el nativo digital construye sus conceptos en base a objetos digitales; absorben rápidamente la información de imágenes multimedia y videos; adquieren información en simultáneo de diversas fuentes, incluidas las redes sociales y esperan respuestas instantáneas [13].

Su estilo de aprendizaje ha cambiado en comparación con generaciones anteriores y se ve ampliamente influenciado por la tecnología.

B. Los estilos de aprendizaje

Se pueden definir como: "Rasgos cognitivos, afectivos, fisiológicos de preferencia por el uso de los sentidos, ambiente, cultura, psicología, comodidad, desarrollo y personalidad, que sirven como indicadores realmente estables de cómo las personas perciben, interrelacionan y responden a sus ambientes de aprendizaje y a sus propios métodos o estrategias en su forma de aprender" [14].

Es un conjunto de características cognitivas y pedagógicas que clasifican al estudiante según su forma de procesar y adquirir información [14]. De acuerdo a esa clasificación, se establece que los estudiantes pueden tener diferentes estilos de aprendizaje: visual, auditivo, kinestésico, etc.

En la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa - Perú, los docentes del curso de Fundamentos de Programación 1, primer curso de programación, se interesaron en descubrir el estilo de aprendizaje de los estudiantes denominados "nativos digitales" en nuestro entorno en particular.

El objetivo es aplicar métodos de enseñanza-aprendizaje de acuerdo a su proceso cognitivo y pedagógico. Se ha aplicado el Test desarrollado por Lynn O'Brien, a un universo de 103 estudiantes, ver Fig. 2, confirmando el resultado de los estudios de [13], los cuales determinan que los nativos digitales han desarrollado una capacidad de aprendizaje muy visual y kinestésico.

O'Brien plantea la siguiente clasificación de los estilos de aprendizaje: i) los visuales: los individuos en quienes predomina este sistema representacional captan el mundo con los ojos, se fijan mucho en los detalles visuales, recuerdan muy especialmente aquello que ven y hablan con predicados vinculados a este sentido, ii) los auditivos: experimentan el mundo a través del oído, se fijan mucho en los detalles auditivos, recuerdan lo que dice la gente y su lenguaje está muy influido por términos y expresiones vinculadas a la audición y iii) los kinestésicos: en esta tercera y última categoría tienen preferencia por la experiencia y la práctica, ya sea simulada o real; la información que se presente a los estudiantes debe corresponderse con hechos o casos de estudio concretos y significativos con el entorno en el que se desenvuelven, para que puedan poner en práctica tal información.

Los docentes del curso de Fundamentos de Programación 1 han utilizado este conocimiento para el rediseño del material de la primera sesión del curso y así lograr captar la atención, romper barreras en los estudiantes de esta generación y enseñar los conceptos fundamentales de programación de computadoras de forma efectiva.

Asimismo, el rediseño del material se basa en las sugerencias de [15], el cual determina técnicas para los estilos de aprendizaje y enseñanza en la educación de ingenieros. Se han considerado las siguientes sugerencias: i) que para utilizar un estilo de aprendizaje (deductivo/activo), se deben utilizar actividades que sean asistidas por el computador, ii) que se les dé a los estudiantes la oportunidad de trabajar de forma colaborativa, ese aprendizaje activo mejora cuando interactúan con sus pares, iii) motivar el aprendizaje, utilizando material que mejore la experiencia personal de los estudiantes, proceso inductivo/global.

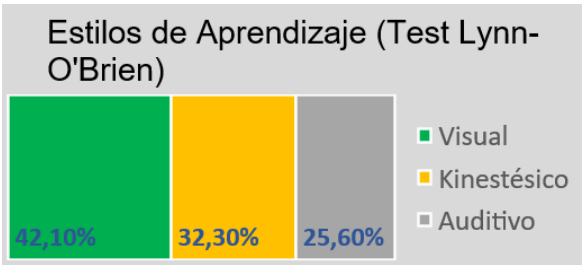

Fig. 2. Estilos de Aprendizaje – Test-Lynn O'Brien, aplicado a 103 estudiantes del curso de Fundamentos de Programación 1, en la Escuela Profesional de Ingeniería de Sistemas

IV. LIGHTBOT Y ALICE

A. *Lightbot*

Es un videojuego tipo rompecabezas (puzzle game). Sus propiedades nos llevan directamente a establecer relaciones con conceptos de programación. La meta del juego es guiar al robot a encender los cuadros azules. El jugador le debe indicar cómo lograrlo, usando un conjunto de instrucciones mostradas en la Fig. 3. Estas instrucciones son `forward()`, `turnRight()`, `turnLeft()`, `jump()`, `light()`, `funct1()` y `funct2()`.

Al comienzo sólo aparecen las dos primeras instrucciones y el resto de instrucciones van apareciendo a medida que vamos alcanzando un nuevo nivel y se constituyen en herramientas para la solución de problemas más complejos.

B. *Alice*

Es un entorno de programación diseñado específicamente como una herramienta de enseñanza-aprendizaje que permite a los programadores novatos poder crear animaciones y juegos con mundos 3D [16].

Fue desarrollado por la Universidad de Carnegie Mellon en EE.UU. como apoyo para la enseñanza de la Programación Orientada a Objetos.

Con este programa resulta más fácil comprender la programación de la orientación a objetos, por su naturaleza visual y el atractivo de las animaciones en 3D.

Alice va de la mano con Java, ya que utiliza parte de este lenguaje para las funciones de animación aplicables a objetos, tales como mover, saltar, girar, etc.

V. PROPUESTA DE LA PRIMERA SESIÓN DE FUNDAMENTOS DE PROGRAMACIÓN 1

La primera sesión se convierte en un hito temprano para el resto del curso. Después de la presentación del docente y del curso en general se realiza la encuesta cuyos resultados se muestran en las tablas I, II y III. Posteriormente se realizan las siguientes actividades:

A. *Actividades con el Videojuego Lightbot*

Se les explica que aprenderán muchos de los conceptos básicos de la programación "jugando". Inicialmente muestran sorpresa por dicha afirmación, aunque queda debidamente justificada por los resultados de la encuesta, y se despierta expectativa, capturando así su atención inmediatamente.

Se les comenta que en años superiores de estudios podrán programar Inteligencia Artificial (IA) para que permita a robots funcionar autónomamente, pero como recién acaban de ingresar a la universidad, primero programaremos robots que obedezcan nuestras instrucciones al pie de la letra.

En la Fig. 4 se les muestra el entorno del juego, la forma de interactuar con él, indicando que el objetivo es prender los cuadros azules y las instrucciones que pueden utilizar para alcanzar dicho objetivo.

Se les plantea que sólo pueden utilizar las instrucciones resaltadas `forward()` y `light()`.

La motivación es inmediata, es un reto a superar, despertando así, su interés por solucionar el problema. Se ven estimulados visual, kinestésica y auditivamente. Absolutamente todos dan la solución, ver Fig. 5.

Fig. 3. Instrucciones usadas en Lightbot

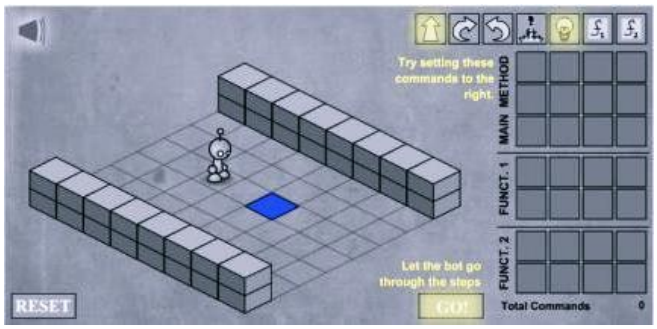

Fig. 4. Entorno de Lightbot y primer problema a resolver

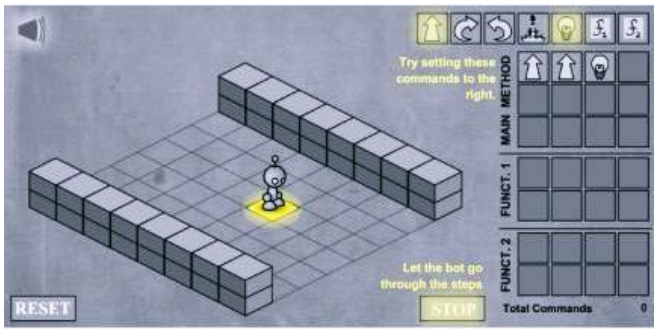

Fig. 5. Solución al primer problema de Lightbot

En este punto se les indica varias observaciones:

1) Ya estamos programando un primer "algoritmo", que es el conjunto de pasos que debe seguir el robot para alcanzar su objetivo (prender el punto azul)
2) Estamos programando en el "main method" o método principal, que siempre será el punto de partida de toda ejecución de cualquier programa
3) Estamos usando dos "instrucciones básicas", las cuales no han sido programadas por nosotros, no sabemos cómo fueron implementadas, sin embargo, las usamos y "abstraemos" cómo se han programado a más bajo nivel por alguien más, sin preocuparnos por ello
4) Los pasos que indicamos previamente de forma visual, lo podemos traducir a cierto lenguaje de programación:

```
forward()
forward()
light()
```

   Que ya es un "programa de alto nivel" compuesto por 3 líneas de código.
5) Las instrucciones que usamos las podemos "reutilizar" las veces que queramos, ahorrándonos recursos de tiempo
6) También podríamos llamar a cada una de dichas instrucciones, un "método"
7) Siempre que veamos paréntesis es un indicativo que estamos usando un método que realiza alguna acción

Se plantea el siguiente problema: Ver Fig. 6.

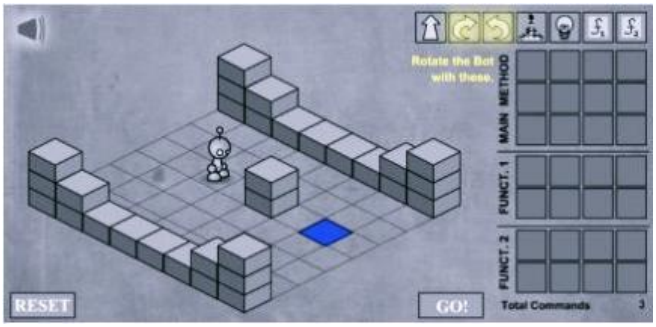

Fig. 6. Segundo problema a resolver. Mayor complejidad

En este punto se les indica varias observaciones:

1) Se muestra que la "complejidad" de los problemas puede aumentar y que es una situación muy normal en programación
2) Se resalta que para su solución se les brinda "nuevas instrucciones": `turnRight(), turnLeft()`
3) Es posible que haya varias soluciones correctas, pero hay unas soluciones mejores que otras, ya que usan menos recursos, lo que llamaremos "eficiencia"

Ver la solución en la Fig. 7.

Ahora se plantea el siguiente problema a ser resuelto, la complejidad aumenta. Ver Fig. 8.

En este punto se les indica varias observaciones:

1) Se muestra que la "complejidad" del problema puede aumentar aún más y de la misma manera será durante el curso de programación
2) A medida que la complejidad aumente, también se les brindará "nuevas instrucciones" para su solución: `jump()` y de la misma forma será durante el curso

Ver la solución en la Fig. 9.

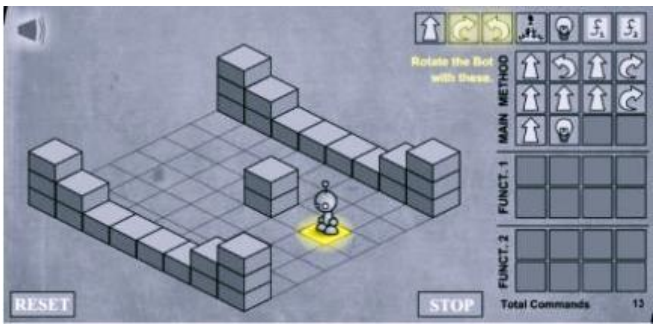

Fig. 7. Solución al segundo problema de Lightbot

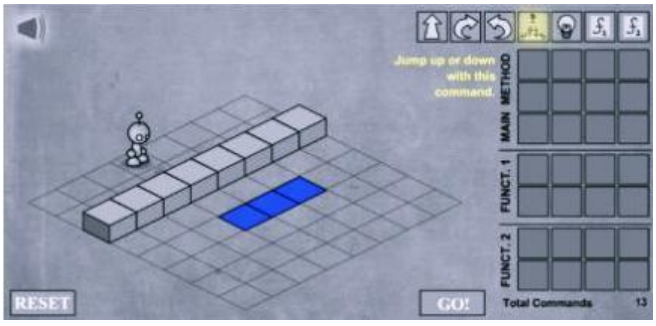

Fig. 8. Tercer problema a resolver. Mayor complejidad

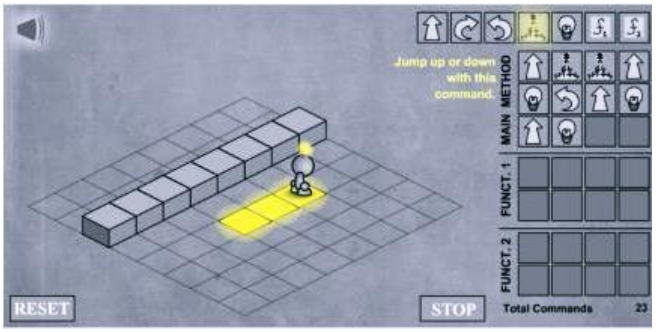

Fig. 9. Solución al tercer problema de Lightbot

Debemos notar que desde un primer momento se les hace hincapié sobre algunos conceptos básicos de la programación, tales como instrucción, algoritmo, programa, abstracción, reutilización, complejidad, método y método principal.

La complejidad continúa aumentando de problema en problema y la motivación aumenta paralelamente, cada problema es un reto concreto que estimula a los estudiantes a resolverlos. Retroalimentamos preguntando sobre los conceptos que van aprendiendo y comprobando que son comprendidos por casi todos los estudiantes. Esta experiencia después es referenciada a lo largo del curso, cuando se les enseña algún concepto de manera formal se les hace la referencia a cierta situación de esta sesión de aprendizaje. Todo lo que se aprende en esta sesión se constituye posteriormente en un "saber previo" para el resto del curso.

Aquí, se plantea un conflicto cognitivo con el siguiente ejercicio, ver Fig. 10.

Los estudiantes tratan de resolver como han estado haciéndolo, pero se encuentran con la restricción que sólo hay 12 posibles instrucciones puede tener el método principal o main. Ver en Fig. 11 un intento de solución, pero que se ve limitado por esta restricción.

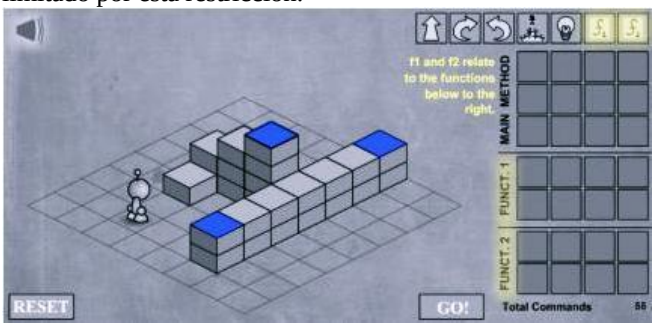

Fig. 10. Problema a resolver. Conflicto cognitivo

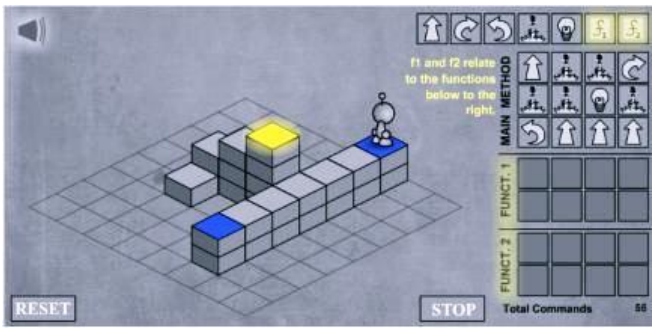

Fig. 11. Primer intento erróneo debido a la restricción

En este punto se les indica varias observaciones:

1) Es imposible resolver el problema con lo que saben hasta ahora, debido a la restricción
2) Se les brinda 2 nuevas instrucciones para la solución: `funct1()` y `funct2()`. Ahí se les enseña a crear sus propias funciones o también llamados métodos 1 y 2, mostrando el mecanismo de invocación del método principal a las funciones o métodos propios
3) La ejecución siempre empezará en el método main, pero éste podrá invocar a las 2 nuevas funciones.

Ver un típico intento fallido de solución con las nuevas funciones en Fig. 12.

En este punto se les indica varias observaciones:

1) Intenten agrupar un conjunto de instrucciones que se repiten varias veces y pónganlas dentro de una función o método
2) Las funciones se pueden invocar cualquier cantidad de veces

Ver una posible solución en la Fig. 13.

Los estudiantes en este punto están totalmente inmersos en la sesión y se muestra su interés por seguir solucionando problemas más y más complejos, ver Fig. 14 y Fig 15.

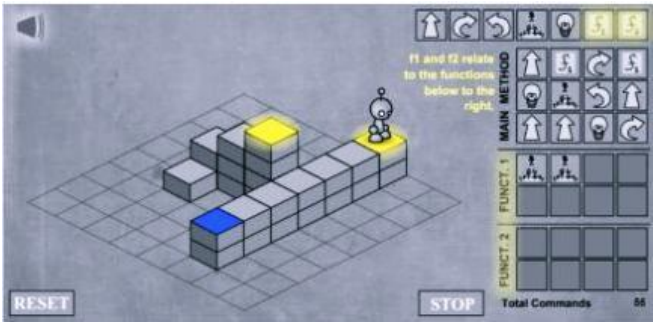

Fig. 12. Intento de solución usando func1()

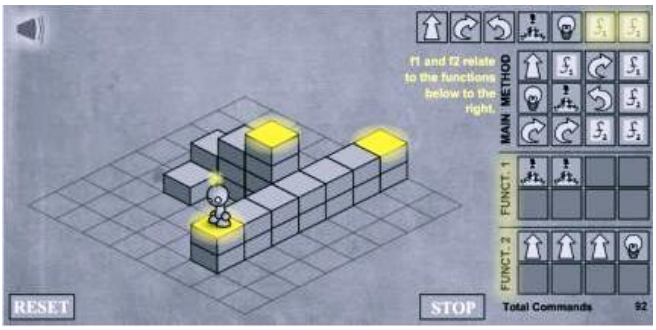

Fig. 13. Solución al problema con restricciones usando las funciones o métodos propios

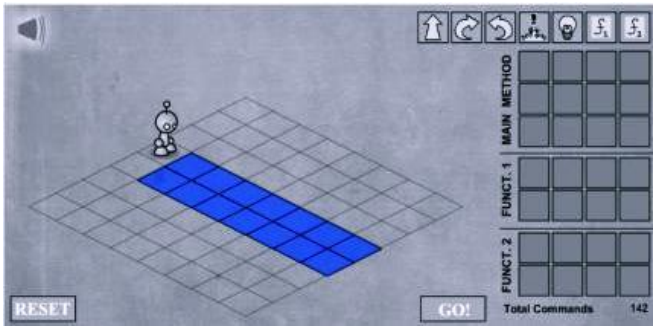

Fig. 14. Problema a resolver usando func1() y func2()

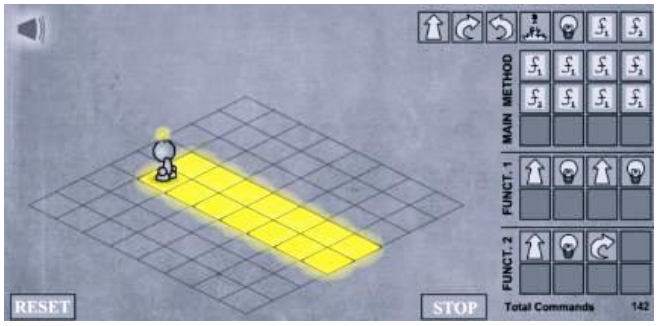

Fig. 15. Una entre varias posibles soluciones al problema

Se les hace hincapié en la metáfora que la programación va a ser un proceso muy similar a lo que están realizando con el videojuego, que la experticia que están logrando en el videojuego es por medio de la práctica y de la misma manera se lograrán la experticia para programar computadoras. Queda como actividad completar el juego en otros niveles superiores, incluso de condicionales, loops y recursión, que si bien no era el objetivo, ellos ya se encuentran con la motivación y el conocimiento para resolver los problemas por sí mismos. Además, se les estimula para jugar Lightbot 2.0, el que trae nuevos retos.

Después de retroalimentar la experiencia, ingresamos a la segunda parte de la sesión.

B. Actividades con el entorno de animación 3D Alice

Aquí, se debe aprovechar el momentum obtenido previamente y se introduce Alice, explicándoles que ahora se verá otro entorno, uno de animación; se les muestra la forma básica de interactuar con él y la forma de crear animaciones en 3D. Ver Fig.16.

Nuestro curso, al tener como lenguaje de programación a Java, necesita ciertos conceptos de orientación a objetos que de forma convencional siempre resultaron algo complejos de enseñar, especialmente al inicio del curso.

Con Alice conceptos como clase, objeto, atributo, método y mensaje, quedan bastante claros. Para nuestro objetivo planteamos una trama bélica, considerando que casi todos los estudiantes juegan videojuegos de estrategia.

Nos situamos en las tierras nórdicas de los vikingos, donde vamos a crear un ejército a punto de luchar. La pregunta es: para crear un ejército, ¿qué es lo primero que debemos tener? ¿de qué está compuesto un ejército?, la respuesta obvia es “Soldado”. Se introduce el concepto de clase, como aquella plantilla para crear soldados reales. En la galería de Alice buscamos alguna clase que se asemeje a un soldado y la arrastramos al escenario. Al hacerlo, nos pide el nombre del soldado. Ver Fig. 17.

Fig. 16. Entorno de la herramienta de animaciones 3D Alice

Fig. 17. Ejemplo de animación 3D en Alice

Indicamos que lo que hemos creado es un objeto de la clase y que podemos crear n objetos de dicha clase, cada objeto con sus propias características o atributos. Inicialmente creamos objetos estándar que tienen básicamente los mismos atributos. Los objetos se llaman Ragnar, Bjorn y Floki. Pero vamos a manipular los valores de sus atributos "altura" y "tono". Ver Fig. 18 y Fig. 19.

Ahora indicamos que se le puede dar comportamiento a cada uno de los objetos independientemente, así a Floki haremos que se mueva hacia adelante 0.5, a Bjorn que gire a la derecha 0.5 (180º) y a Ragnar que gire y apunte su mirada hacia Bjorn. Los comportamientos lo hacemos en Alice por medio de los "procedures", también llamados funciones o métodos en otros lenguajes de programación. Ver Fig. 20.

Indicamos que el hecho de invocar un método de algún objeto se llama "mandar un mensaje" a dicho objeto y que, si a dicho método se le manda información extra, la llamaremos "argumentos" del método y siempre van entre paréntesis. Así, los comportamientos que se les asignaron visualmente a los 3 objetos corresponderían a las siguientes líneas de código:

```
Floki.move(FORWARD, 0.5)
Bjorn.turn(RIGHT, 0.5)
Ragnar.turnToFace(Bjorn)
```

Fig. 18. Tres objetos estándar de la clase Soldado

Fig. 19. Tres objetos de la clase Soldado con valores diferentes en sus atributos altura y tono

Fig. 20. Tres objetos de la clase Soldado aplicándoles algún comportamiento por medio de los procedures, funciones o métodos

Debemos aclarar que, si bien podríamos entrar ya a la programación por medio de Alice, no es nuestro objetivo principal. Lo que deseamos es que los estudiantes entiendan ciertos conceptos y se sientan motivados para el aprendizaje de un lenguaje de programación orientado a objetos como Java.

Finalmente, se les motiva a los estudiantes a que creen animaciones en 3D usando su imaginación y siendo totalmente libres para realizar la actividad, inclusive colaborando entre ellos. Hay que apuntar que hubo trabajos muy sobresalientes y que aplicaron incluso conceptos y técnicas no enseñadas en clase.

VI. Análisis De Resultados

La retroalimentación que recibimos por parte de los estudiantes es muy positiva ya que realmente disfrutan las dos herramientas lúdicas que se les presentó. Incluso para la segunda sesión, al empezar se les pregunta sobre la actividad que se les dejó a desarrollar y la gran mayoría indicó que avanzaron en el videojuego de Lightbot, además unos cuantos hasta completaron todos los niveles. Sobre la actividad de Alice, todos sin excepción crearon algún mundo en 3D, unos más elaborados que otros, pero todos cumplieron satisfactoriamente.

Es destacado el hecho que al preguntar al azar sobre alguno de los conceptos de programación vistos en esta primera sesión, todos respondían correctamente.

Hay que considerar que en el curso siempre tuvimos un gran problema en el rendimiento de los estudiantes en el primer examen, el cual se realiza al acabar la segunda unidad, en dicho examen las notas fueron bajas los dos primeros años en que se dictó el curso de forma tradicional. Después de esta experiencia con entornos lúdicos se vieron favorecidos los resultados en el primer examen, tales como la cantidad de aprobados, promedio general, incluso las notas máxima y mínima aumentaron, y la desviación estándar disminuyó mostrando una mayor homogeneidad (calificación vigesimal). Tabla VII.

TABLA VII.
Resultados evaluación del Primer Examen

	2013	2014	2015	2016	2017	2018
Cantidad estudiantes	115	120	111	90	120	130
Aprobados	41.84%	44.12%	52.81%	55.02%	56.32%	55.23%
Desaprobados	58.16%	55.88%	47.19%	44.98%	43.68%	44.77%
Promedio	8.50	9.25	12.05	12.95	12.75	12.70
Desviación Estándar	4.12	4.28	3.78	3.52	3.48	3.60
Máxima	16	17	19.5	19	19	19
Mínima	1	2	4.5	7	5	3

CONCLUSIONES

En este artículo se ha compartido la experiencia de la primera sesión del curso de Fundamentos de Programación 1.

Se ha presentado las herramientas lúdicas que utilizamos para estimular la motivación de los estudiantes y enseñar de manera práctica y efectiva conceptos fundamentales de la programación, tales como: instrucción, abstracción, reutilización, complejidad, algoritmo, programa, función o método, eficiencia, clase, objeto, atributo, método, mensaje y argumento.

Se ha demostrado que una aproximación de la enseñanza basada en entornos lúdicos es efectiva para una generación de estudiantes que son denominados nativos digitales, que entre otras características presentan desinterés por lo que no les motiva visualmente ni les llama la atención.

La relación con estas herramientas no acaba ahí, siempre constituyen parte de los ejemplos al enseñar los conceptos de programación de forma convencional a lo largo de este curso y del curso que continúa Fundamentos de Programación 2, suavizando de esta forma su posterior aprendizaje.

Consideramos que esta experiencia puede ser perfectamente replicada en otras carreras profesionales, no sólo de ingeniería, y que también se puede realizar a nivel escolar primario y secundario.

REFERENCIAS

[1] E. Sweedyk, M. deLaet, M. C. Slattery, and J. Kuffner, "Computer games and CS education: why and how," in Proceedings of the 36th SIGCSE technical symposium on Computer science education , St. Louis, Missouri, USA , 2005, pp. 256-257.

[2] U. Wolz, T. Barnes, I. Parberry, and M. Wick, "Digital gaming as a vehicle for learning," ACM SIGCSE Bulletin, vol. 38, no. 1, pp. 394-395, 2006.

[3] J. D. Bayliss and S. Strout, "Games as a "flavor" of CS1," ACM SIGCSE Bulletin, vol. 38, no. 1, pp. 500-504, Mar. 2006.

[4] T. Lorenzen, W. Heilman , "CS1 and CS2: Write Computer Games in Java!" SIGCSE Bull., 34(4), 99-100, 2002.

[5] B. W. Becker. "Teaching CS1 with karel the robot in Java." ACM SIGCSE Bulletin. Vol. 33. No. 1. ACM, 2001.

[6] Escuela Profesional de Ingeniería de Sistemas. http://fips.unsa.edu.pe/ingenieriadesistemas/ Último acceso Marzo 2019.

[7] Java. https://www.oracle.com/technetwork/java/index.html Último acceso Marzo 2019.

[8] M. Aedo, E. Vidal, E. Castro & A. Paz, "Aproximación orientada a Entornos Lúdicos para la primera sesión de CS1 - Una experiencia con nativos digitales", 15th Latin American and Caribbean Conference for Engineering and Technology (LACCEI) - International Multi-Conference for Engineering, Education, and Technology, United States, 2017.

[9] Lightbot 1.0. https://armorgames.com/play/2205/light-bot Último acceso Marzo 2019.

[10] Lightbot 2.0. https://armorgames.com/play/6061/light-bot-20 Último acceso Marzo 2019.

[11] Alice. http://www.alice.org/ Último acceso Marzo 2019.

[12] ABET. Criteria for Accrediting Engineering Programs, 2015 – 2016. http://www.abet.org/wp-content/uploads/2015/05/E001-15-16-EAC-Criteria-03-10-15.pdf#outcomes. Último acceso Marzo 2019.

[13] M. Prensky, "Digital Natives, Digital Immigrants," Horiz., vol. 9, no. 5, pp. 1–6, 2001.

[14] J. García Cué, "Estilos de Aprendizaje y Tecnologías de la Información y la Comunicación en la Formación de Profesorado". Tesis Doctoral. España-UNED, 2006

[15] R. Felder, L. Silverman, "Learning and teaching styles in engineering education," Eng. Educ., vol. 78, no. June, pp. 674–681, 1988.

[16] W. Dann, "Mediated Transfer: Alice 3 to Java", SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education, 2012.

Marco Aedo es Ingeniero de Sistemas y Magister (c) en Ingeniería de Software por la Universidad Nacional de San Agustín. Es autor o coautor de artículos en conferencias internacionales y de libros. Es actualmente Docente Asociado en la Universidad Nacional de San Agustín de Arequipa – Perú y es investigador del Centro de investigación, Transferencia de tecnologías y Desarrollo de Software I+D+i, impulsando la creación de grupos en áreas: Computación gráfica, Sistemas cognitivos, Apps móviles, Seguridad de la información y Tecnologías emergentes - CiteSoft.

Elizabeth Vidal es Ingeniera de Sistemas y Magister en Economía por la Universidad Nacional de San Agustín. Es autor o coautor de artículos en conferencias internacionales. Es actualmente Docente Principal en la Universidad Nacional de San Agustín de Arequipa – Perú y es investigadora del Centro de investigación, Transferencia de tecnologías y Desarrollo de Software I+D+i, impulsando la creación de grupos en áreas: Computación gráfica, Sistemas cognitivos, Apps móviles, Seguridad de la información y Tecnologías emergentes - CiteSoft.

Eveling Castro es Ingeniera de Sistemas, Magister en Ingeniería de Sistemas por la Universidad Católica de Santa María y Doctora (c) en Ciencia de la Computación por la Universidad Nacional de San Agustín. Es autor o coautor de artículos en conferencias internacionales. Es actualmente Docente Principal en la Universidad Nacional de San Agustín de Arequipa – Perú y es investigadora principal del Centro de investigación, Transferencia de tecnologías y Desarrollo de Software I+D+i, impulsando la creación de grupos en áreas: Computación gráfica, Sistemas cognitivos, Apps móviles, Seguridad de la información y Tecnologías emergentes - CiteSoft.

Alfredo Paz es Ingeniero de Sistemas y Magister (c) en Ingeniería Informática por la Universidad Nacional de San Agustín. Es autor o coautor de artículos en conferencias internacionales. Es actualmente Docente Asociado en la Universidad Nacional de San Agustín de Arequipa – Perú y es investigador del Centro de investigación, Transferencia de tecnologías y Desarrollo de Software I+D+i, impulsando la creación de grupos en áreas: Computación gráfica, Sistemas cognitivos, Apps móviles, Seguridad de la información y Tecnologías emergentes - CiteSoft.