Transición desde programación basada en bloques a basada en texto: una revisión del campo

Jorge Rodríguez[1], Gerardo Parra[1], Daniel Dolz[1], and Rubén Ramírez[2]

[1] Grupo de Investigación en Lenguajes e Inteligencia Artificial
Departamento de Teoría de la Computación - Facultad de Informática
Universidad Nacional del Comahue
Buenos Aires 1400, Neuquén, Argentina
[2] Consejo Provincial de Educación
Ministerio de Educación de la Provincia de Neuquén
Belgrano 1300, Neuquén, Argentina
j.rodrig@fi.uncoma.edu.ar,gparra@fi.uncoma.edu.ar,ddolz@fi.uncoma.edu.ar,
rhubenariel@gmail.com

Resumen El enfoque de programación por bloques demuestra ser una opción efectiva para dar los primeros pasos en el arte de programar. Se espera que en el transcurso de la escuela secundaria los estudiantes logren desarrollar programas usando lenguajes basados en texto. El paso a un entorno de programación tradicional, como Netbeans+Java, suele resultar un proceso frustrante para la mayoría de los estudiantes.
Este artículo presenta una revisión sobre entornos diseñados para favorecer y facilitar los procesos de transición de programación basada en bloques a programación basada en texto.
Estos entornos buscan hacer más accesible la programación tradicional para estudiantes con experiencia previa en ambientes basados en bloques y posiblemente reducir la curva de aprendizaje como también los niveles de frustración.
Además, este artículo presenta un modelo que permite evaluar estos recursos y constituye una herramienta para orientar los procesos de especificación y desarrollo de este tipo de entornos.

Palabras Clave: Lenguajes de Programación Basados en Bloques, Lenguajes de Programación Basados en Texto, Escuela Secundaria, Educación en Ciencias de la Computación, Transición Bloque a Texto.

1. Introducción

La enseñanza de la computación en la escuela secundaria está transitando una profunda transformación. En muchos países se promueve el desarrollo de propuestas curriculares que adopten prácticas y conceptos fundamentales sobre Ciencias de la Computación como contenidos escolares para la educación secundaria [12,15,16].

En este contexto, las prácticas y conceptos relacionados al área de conocimiento Algoritmos y Programación reciben una atención creciente en el campo educativo. En los últimos años se realizan numerosas investigaciones que buscan definir enfoques y construir entornos destinados específicamente a la enseñanza y el aprendizaje de la programación en ambientes formales y no formales.

El enfoque y los entornos de programación basada en bloques demuestran ser una opción efectiva para introducir conceptos y prácticas fundamentales sobre algoritmos y programación a estudiantes sin formación previa en el área de conocimiento [19].

Sin embargo, existen desafíos emergentes para la enseñanza de las ciencias de la computación al momento de plantear una incorporación sostenida, rigurosa y progresiva a lo largo de los años de escolaridad [16].

Se espera que en el transcurso de la educación secundaria los estudiantes logren construir habilidades para desarrollar programas utilizando lenguajes de programación convencionales basados en texto. El paso a un entorno de programación tradicional suele resultar un proceso frustrante para la mayoría de los estudiantes [13].

En este sentido se justifica la necesidad de concretar trabajos de investigación y desarrollo sobre entornos que aborden el problema de la transición y se constituyan en soporte a prácticas pedagógicas coherentes con las necesidades de formación.

En este trabajo se presenta una revisión sobre entornos de programación que favorecen la transición de bloques a texto. Por otra parte, se expone un modelo donde se incluye la descripción de las características y componentes fundamentales de este tipo de entornos y cómo se relacionan con los procesos de transición.

El resto de este documento está organizado de la siguiente manera. En la sección 2, se describen trabajos relacionados que aportan un marco de referencia para el estudio. En la sección 3, se presenta el modelo para los ambientes de transición elaborado en el marco de este trabajo. La sección 4, muestra una revisión y análisis sobre las características de un conjunto de ambientes. Finalmente, en la sección 5, se presentan conclusiones y líneas de trabajos futuros.

2. Trabajos relacionados

Durante las últimos años, el problema de la transición desde la programación basada en bloques a la tradicional, que usa lenguajes basados en texto, está ganando atención como campo de investigación y desarrollo en el ámbito de la Educación en Ciencias de la Computación. Los ambientes desarrollados en este contexto, sus características y estrategias de enseñanza son algunas de las perspectivas en esta área emergente de investigación [11,17,13,18].

Varios estudios plantean que tanto los entornos basados en bloques como los basados en texto tienen beneficios y límites; además describen que existe una brecha importante entre los ambientes que dificulta los procesos de transición [13,14].

Las investigaciones acuerdan en relación a que los entornos de programación basados en bloques aportan ventajas significativas para construir habilidades de pensamiento algorítmico y comprender conceptos fundamentales sobre algoritmos y programación. Al mismo tiempo, establecen que no resultan adecuados para enseñar conceptos abstractos y prácticas complejas [13,18].

Por otro lado, la programación basada en texto constituye el estándar actual para la mayoría de los lenguajes y es un aprendizaje esperado en el contexto de la educación secundaria. Sin embargo, adoptar un lenguaje tradicional como primer contacto con la programación tiene varias limitaciones que causan dificultades y frustración. El hecho de producir código en un ambiente profesional implica una carga cognitiva que frecuentemente distrae de la complejidad intrínseca de la tarea de programación [13].

3. Modelo propuesto

En este trabajo, nos proponemos estudiar entornos que favorezcan y faciliten la transición de programación basada en bloques a programación basada en texto. En tal sentido, se han realizado experiencias satisfactorias con herramientas de la categoría mixta con ambientes del tipo de caja de arena.

Por herramientas de la categoría mixta se hace referencia a las que posibilitan trabajar con bloques visualizando, en forma inmediata, el código producido automáticamente, y viceversa.

En el contexto de este estudio se trabaja en la definición de un modelo que permita orientar el diseño y construcción de ambientes destinados a facilitar los procesos de transición. En este sentido, se busca sostener las ventajas de la representación gráfica y la característica de mapeo directo de las operaciones sobre el escenario con intención de posibilitar una retroalimentación ágil y la detección y corrección de errores. Al mismo tiempo que se mantiene la tarea en el plano de lo concreto se proporciona la flexibilidad, versatilidad y legibilidad del texto [19].

Se trabaja en la especificación de alguna herramienta que permita este paso. Idealmente dicha herramienta debería contener:

- Sandbox: Proponemos que el ámbito de aplicación de la herramienta sea contenido dentro de entornos definidos y aislados del sistema subyacente con límites claros pero con libertad dentro de los mismos.
- Multilenguaje: la herramienta debería proveer la posibilidad de generar e interactuar con más de un lenguaje de programación. Preferentemente, que algunos de ellos sean los mismos que luego serán utilizados en ambientes solo texto para continuar con la profundización de la disciplina.
- Modalidad dual: debería ser posible generar código a partir de bloques, y viceversa, en tiempo real. Las construcciones obtenidas a partir de los bloques, y viceversa, deben ser lo más directas posibles, para que la equivalencia entre ambas estructuras sea clara y no haya dudas que representan al mismo algoritmo de alto nivel.

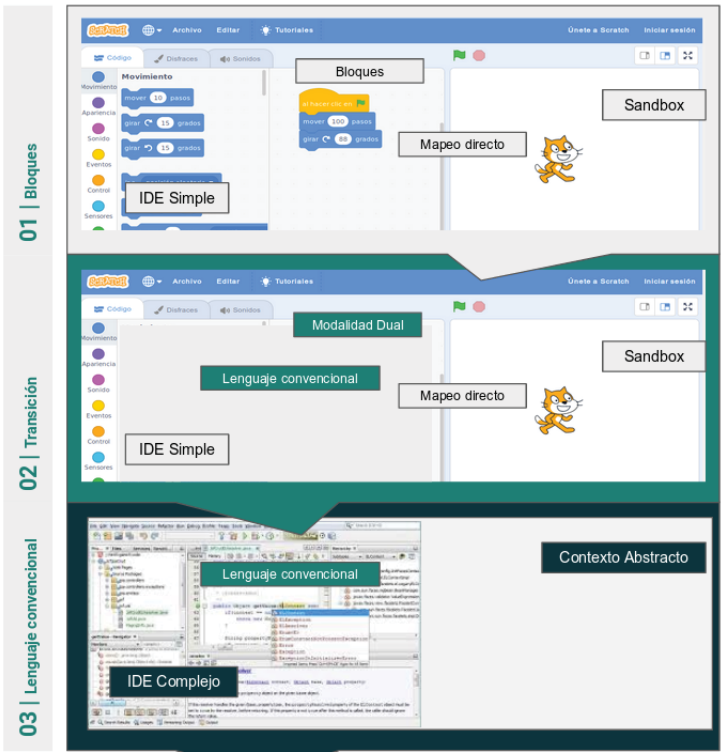

Figura 1. Modelo de ambiente de transición propuesto

- Autoevaluativa: no debería ser necesaria la ayuda de un docente para que el estudiante sepa si su algoritmo resolvió el problema o no. De todas maneras, esto no excluye una instancia de evaluación posterior de eficiencia, elegancia y posibles alternativas de solución diferentes.
- Instalable: esta línea siempre tiene en cuenta estudiantes sin acceso permanente a internet. Por lo tanto, una herramienta puramente *web-enabled* no fácilmente instalable de manera local no estaría al alcance de toda la comunidad educativa.
- Conjunto reducido del lenguaje: especialmente si uno o más de los lenguajes de traducción es de uso comercial como Java, Python, JS, etc, es importante acotar el conjunto de instrucciones para no dispersar con las mismas el objetivo del aprendizaje. Siendo las instrucciones excluidas posibles de incorporar en etapas posteriores con otros ambientes.
- Mapeo directo: Las operaciones programadas se mapean directamente sobre el mundo ofreciendo una retroalimentación instantánea de la actividad en la pantalla o el entorno físico.
- IDE simple: Los entornos de desarrollo profesionales como Netbeans o Eclipse se suelen ver en las aulas como un “entorno de aprendizaje”, aunque no se

han desarrollado con fines educativos. Estas herramientas ofrecen una amplia gama de funciones e interfaz compleja de las cuales la educación informática secundaria, en un primer momento, sólo requiere de una pequeña parte. Se propone trabajar con IDEs simples que integren algunas funciones de alerta visual de errores, reconocimiento de sintaxis y posibilidad de ejecutar el código.

4. Revisión de ambientes para la transición desde la programación basada en bloques a basada en texto

A continuación, se presentan diez ambientes de programación que tienen la posibilidad de facilitar la transición desde la programación basada en bloques a basada en texto.

En el contexto de este estudio se realizó una búsqueda en las bases de datos bibliográficas internacionales en línea, como ACM Digital Library, IEEE Xplore Digital Library y Google Scholar, con intención de identificar los principales ambientes. En un segundo momento, se realizó un análisis sobre las características de los ambientes seleccionados. Finalmente, se elaboró un estudio comparativo sobre los ambientes.

4.1. Ambientes

CodeCombat. CodeCombat[2] es un ambiente basado en juegos diseñado con el objetivo de aprender a programar. Los estudiantes escriben fragmentos de código y, de esa manera, pueden observar cómo los personajes del ambiente reaccionan en tiempo real. El ambiente consiste en un juego estilo RPG (Role Playing Game) estilo *Calabozos y Dragones*, donde en vez de jugar con los cursores o joysticks, las instrucciones se dan en Python y JavaScript mientras la ejecución se observa en un ambiente visual estilo juego de video.

CodinGame. CodinGame[4] es una plataforma online de entrenamiento para programadores y desarrolladores que les permite jugar y resolver problemas mediante la programación. El entorno provee desafíos cada vez más complejos, diseñados con el objetivo de aprender a codificar mejor con una aplicación de programación en línea que admite hasta veinticinco lenguajes de programación diferentes.

PythonTurtle. PythonTurtle[8] intenta proporcionar una de las formas más simples de aprender y enseñar el lenguaje de programación Python. Los estudiantes emplean un *shell* interactivo y utilizan las funciones de Python para mover una tortuga (en inglés, *turtle*) que se muestra en la pantalla.

El *shell* en PythonTurtle es un *shell* completo de Python. Es posible construir bucles (*loops*), definir funciones, crear clases, etc. PythonTurtle es completamente auto-contenido y no requiere tener instalado el lenguaje Python.

Processing. Processing[7] es un ambiente orientado a artistas y estudiantes. Con entorno similar a Wiring, se asemeja a lenguajes como Java y C++. Su simplicidad reside en que a partir de librerías y funciones disponibles se puede crear aplicaciones audiovisuales y juegos interactivos en muy poco tiempo.

Greenfoot. Greefoot[5], es un entorno de desarrollo interactivo diseñado específicamente con propósitos educativos. Permite, en una primera instancia, la manipulación simple de objetos sobre escenarios bidimensionales. En un segundo momento, se opera el editor de código Java, dónde se codifica el comportamiento de los actores reutilizando las librerías.

Pencil. Pencil Code[6] es un sitio de programación colaborativa para diseñar gráficos, reproducir música y crear juegos. Además, es posible experimentar con funciones matemáticas, geometría, páginas web, simulaciones y algoritmos. Los programas están disponibles para que todos puedan ver y remixar.

El sitio permite aprender y experimentar con lenguajes de programación profesionales usando un editor que admite trabajar en bloques o mediante texto. El lenguaje principal es Coffeescript.

Monkey. CodeMonkey[3] es un juego educativo en el que los estudiantes aprenden a programar en un lenguaje real de programación, sin que se requiera experiencia previa. Es un juego *online* que enseña programación por medio del lenguaje CoffeeScript. El ambiente está diseñado para niños desde los 8 años y permite introducir, de manera amena, conceptos básicos de programación como: variables, operadores lógicos, sentencias condicionales, sentencias repetitivas, argumentos, llamadas a funciones, entre otros.

Tynker. Tynker[10] es un entorno de programación híbrido, basado en bloques, Java y Python, para diferentes edades. Proporciona unos grados más de tangibilidad a la propuesta de enseñanza de las Ciencias de la Computación al permitir programar dispositivos como drones y robots. Potencia en los estudiantes el empleo de modularización y abstracción de alto nivel.

BrickLayer. BrickLayer[1] es un ambiente que permite programar manipulando bloques estilo Lego y armando objetos complejos a partir de los mismos. En este ambiente, los programas son escritos en un lenguaje de programación funcional, denominado SML.

RoboMind. RoboMind[9] es un entorno de programación educativa con un lenguaje de scripting similar a JavaScript. Presenta una metáfora de un mundo cuadriculado donde los estudiantes interactúan con un robot simulado.

4.2. Estudio comparativo sobre los ambientes

En esta sección se presenta una análisis de carácter comparativo sobre las características observadas en los ambientes revisados. El Cuadro 1 muestra una síntesis de las características presentes en cada ambiente.

	Ambiente									
Característica	01	02	03	04	05	06	07	08	09	10
Sandbox			Sí	Sí	Sí	Sí		Sí		Sí
Lenguaje	PJs	PJJs+	P	PJJs	J	Pr	Pr	PJ	Pr	Pr
Modalidad dual						Sí		Sí		
Autoevaluativa	Sí	Sí					Sí	Sí	Sí	
Instalable (INS)/On line (OLn)	OLn	OLn	INS	INS	INS	OLn	OLn	OLn	OLn	INS
Conjunto reducido del Lenguaje	Sí		Sí			Sí	Sí	Sí	Sí	Sí
Mapeo Directo	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
IDE Simple	Sí		Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí

Cuadro 1. Características observadas en los ambientes estudiados. 01: CodeCombat, 02: CodinGame, 03: PythonTurtle, 04: Processing, 05: Greenfoot, 06: Pencil, 07: CodeMonkey, 08: Tynker, 09: BrickLayer y 10: RoboMind.Lengujajes: P: Python, J: Java, Js: JavaScript, Pr: Lenguaje propio del ambiente y +: otros lenguajes

La característica de caja de arena, que ofrece un espacio libre y al mismo tiempo contenido de trabajo, está presente en todos los ambientes explorados excepto en CodeCombat, CodinGame, BrickLayer y CodeMonkey. La mayor parte de los ambientes otorga un grado de libertad amplio a los estudiantes, lo que permite diversificar el tipo y complejidad de los proyectos.

CodeCombat, CodinGame, BrickLayer, CodeMonkey y Tynker cuentan con características que mejoran las posibilidades de autoevaluación y asistencia automática, estos ambientes trabajan con desafíos de complejidad creciente con verificación automática de la solución. CodeCombat, CodinGame, CodeMonkey y Tynker incorporan además características de gamificación trasladando la mecánica de los juegos al ámbito del aprendizaje.

Todos los ambientes incluyen diferentes formas de mapeo directo, esta característica logra sostener la experiencia de aprendizaje en el plano de lo concreto y visible contribuyendo al aplanamiento de la curva de aprendizaje.

CodeCombat, CodinGame, PythonTurtle, Processing y Greenfoot utilizan uno o más lenguajes tradicionales como Python, Java o JavaScript. Esto los pone en mejor posición para transitar hacia entornos de programación reales.

4.3. Experiencia

En el contexto de este estudio, se buscó conocer la percepción de los docentes que se desempeñan en espacios curriculares destinados a la enseñanza de la computación en el escuela secundaria acerca de las posibilidades para favorecer

el proceso de transición que ofrecen dos ambientes específicos: CodeCombat y CodinGame.

La población consultada se compuso por 54 docentes de informática que poseen conocimientos y habilidades previamente adquiridos en el campo disciplinar y didáctico disciplinar. Además, cuentan con experiencia en la enseñanza de la computación en la escuela secundaria.

Se completaron 2 sesiones de 140 minutos en cada caso destinadas a la presentación, reconocimiento y estudio de los ambientes. La actividad buscó que los docentes comprendan el funcionamiento de cada ambiente y construyan una apreciación acerca de sus características y posibilidades en la escuela secundaria.

Cada sesión se organizó en tres etapas: breve presentación del ambiente, estudio utilizando el rol de estudiante y retrospectiva individual acerca de la potencialidad percibida.

Concluidas las dos sesiones de estudio, se consultó sobre la accesibilidad, los motivos de su opinión y las dificultades y utilidades percibidas. Luego, acerca de las posibilidades de participar de situaciones de enseñanza y aprendizaje en contextos reales. Finalmente, se pidió a los participantes que valoren las oportunidades con que cuenta el ambiente para ubicarse como un dispositivo útil para la transición.

La Figura 2 muestra la percepción relativa acerca la accesibilidad de los ambientes. Los docentes aprecian que CodeCombat resulta altamente accesible a estudiantes secundarios (85.19 %). En menor medida (37.04 %) perciben que CodinGame es accesible, posiblemente esta opinión se debe a que la interfaz y desafíos son más complejos.

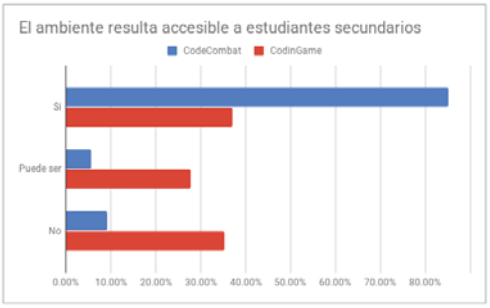

Figura 2. Accesibilidad de los ambientes CodeCombat y CodinGame

El segundo hallazgo se muestra en la Figura 3 donde, en forma coherente con el anterior, CodeCombat claramente es apreciado con mayores posibilidades (79.63 % sobre 18.52 %) de integrarse a procesos reales de enseñanza de la programación en el ámbito de la escuela secundaria.

Para cada ambiente seleccionado, se pidió a los docentes que expresaran si lo consideraban una buena opción para favorecer la transición de programación

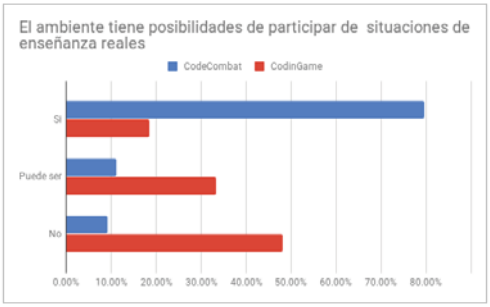

Figura 3. Posibilidades de participar contextos reales de enseñanza de los ambientes CodeCombat y CodinGame

basada en bloques a basada en texto. La Figura 4 muestra que CodeCombat es considerado un dispositivo con altas posibilidades (83.33 %) de constituirse en un puente entre la programación basada en bloques que sucede en los primeros años de la escolaridad secundaria y la programación utilizando lenguajes y entornos convencionales que se espera suceda en el tramo final de la escuela secundaria.

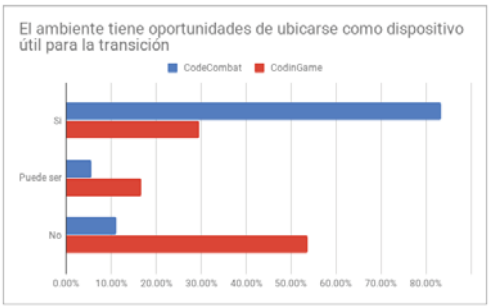

Figura 4. Oportunidad de ubicarse como dispositivo de transición de los ambientes CodeCombat y CodinGame

5. Conclusiones y trabajo futuro

Este documento busca contribuir a mejorar la comprensión acerca del problema de la transición desde la programación basada en bloques a la programación basada en texto y propone un modelo para la selección y definición de ambientes. Por otro lado, se realiza una revisión y análisis de una colección de recursos que tienen diferentes potencialidades en este contexto.

Finalmente, se resume un informe sobre las percepciones elaboradas por la docencia secundaria sobre dos ambientes en particular. Estas apreciaciones aportan

indicios preliminares que confirman el modelo propuesto, en tanto el ambiente que cubre mejor las características esperables recibió aceptación más fuerte.

El objetivo principal de esta línea de investigación es el de aportar nuevos y mejores elementos que colaboren en la definición de próximos entornos destinados a favorecer la enseñanza de las Ciencias de la Computación.

Referencias

1. BrickLayer Homepage. Último acceso Agosto 2019, website https://bricklayer.org.
2. CodeCombat Homepage. Último acceso Agosto 2019, website https://codecombat.com/.
3. CodeMonkey Homepage. Último acceso Agosto 2019, website https://www.playcodemonkey.com/.
4. CodinGame Homepage. Último acceso Agosto 2019, website https://www.codingame.com.
5. Greenfoot. Último acceso Agosto 2019, website https://https://www.greenfoot.org/.
6. Pencil Code Homepage. Último acceso Agosto 2019, website https://pencilcode.net.
7. Processing Homepage. Último acceso Agosto 2019, website https://processing.org/.
8. PythonTurtle Homepage. Último acceso Agosto 2019, website http://pythonturtle.org.
9. RoboMind Homepage. Último acceso Agosto 2019, website https://www.robomind.net.
10. Tynker Homepage. Último acceso Agosto 2019, website https://www.tynker.com/.
11. N. C. Brown, A. Altadmri, and M. Kölling. Frame-based editing: Combining the best of blocks and text programming. In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*. IEEE, 2016.
12. K-12 Computer Science Framework Steering Committee. *The K–12 Computer Science Framework*. ACM, 2016.
13. M. Kölling, N. C. Brown, and A. Altadmri. Frame-based editing. *Journal of Visual Languages and Sentient Systems*, 3(1), 2017.
14. W. Robinson. From scratch to patch: Easing the blocks-text transition. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*, pages 96–99. ACM, 2016.
15. F. Sadosky. *CC – 2016 Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas* . Fundación Sadosky, Argentina, 2013.
16. M. Webb, N. Davis, T. Bell, Y. J. Katz, N. Reynolds, D. P. Chambers, and M. M. Sysło. Computer science in k-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, 22(2):445–468, 2017.
17. D. Weintrop and N. Holbert. From blocks to text and back: Programming patterns in a dual-modality environment. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 633–638. ACM, 2017.
18. D. Weintrop and U. Wilensky. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, 18(1):3, 2017.
19. D. Weintrop and U. Wilensky. How block-based, text-based, and hybrid block/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction*, 17:83–92, 2018.