Consolidación de Conceptos de Programación en Lenguaje Ensamblador a través de la Creación de Juegos y Animaciones Gráficas en el Entorno de Simulación WinMips64

César Estrebou[1] Genaro Camele[1,3] Facundo Quiroga[1]
Horacio Villagarcía Wanza[1,2]

[1] *Instituto de Investigación en Informática LIDI, Facultad de Informática, Universidad Nacional de La Plata**

[2] *Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)*

[3] *Becario postgrado UNLP*

* *Centro asociado de la Comisión de Investigaciones Científicas de la Pcia. De Bs. As. (CIC)*

{cesarest, gcamele, fquiroga, hvw}@lidi.info.unlp.edu.ar

Resumen

Este artículo presenta la primera etapa de un proyecto que tiene como objetivo incentivar el aprendizaje de la programación en lenguaje ensamblador a través del desarrollo de juegos dentro de la asignatura Arquitectura de Computadoras de la Facultad de Informática de la UNLP. Actualmente, en la asignatura se utiliza el simulador WinMips64 para visualizar el cauce de ejecución de un procesador MIPS y ejecutar programas en lenguaje ensamblador. En esta primera etapa se agregaron funcionalidades al simulador para habilitar la programación de juegos sencillos. Para facilitar el desarrollo se creó una aplicación que transforma una imagen en una secuencia de datos estructurados que puede ser incrustada en el código y evitar problemas de codificación. Se realizó un ensayo que involucró un grupo reducido de docentes y alumnos para probar las nuevas funciones del simulador y tener una retroalimentación de la experiencia sobre el desarrollo de los juegos. Esta primera experiencia fue positiva para docentes y alumnos, alcanzando resultados satisfactorios respecto de los objetivos planteados. El código fuente con las modificaciones del simulador WinMips64, los ejemplos de los programas implementados y la aplicación de transformación de imágenes se encuentran disponibles de forma libre en Github para uso de la comunidad.

Palabras Clave: Arquitectura de Computadoras, Simulador Mips, Lenguaje Ensamblador, Programación de Juegos

1. Introducción

Según la literatura relacionada con el aprendizaje hay evidencia de los diversos factores que influyen tanto en el proceso de aprendizaje como en el compromiso cognitivo. De acuerdo con lo investigado por varios autores, algunos de estos factores están relacionados con estilos de aprendizajes diferentes, experiencias previas y motivación entre otros. El desarrollo del pensamiento algorítmico que requiere el aprendizaje de un lenguaje de programación hace que los estudiantes deban enfrentar y vencer la dificultad de comprensión de conceptos abstractos, el poco uso de la imaginación, la incapacidad para modelar los problemas y/o la falta de dominio de la lógica para encontrar soluciones. Pero sin duda uno de los obstáculos más importantes y recurrentes es la falta o pérdida de motivación de los mismos estudiantes [5].

La organización y arquitectura de compu-

tadoras tanto como la programación en lenguaje ensamblador forman parte usualmente de los contenidos formativos de las primeras asignaturas de grado de las carreras universitarias de ciencias de la computación. Normalmente, en este punto de la carrera los estudiantes han aprendido los conceptos elementales de la programación en un lenguaje de alto nivel como puede ser Pascal, C, Python o Java. En particular, la programación en lenguaje ensamblador es mucho más difícil de aprender que la de un lenguaje de alto nivel. La pérdida general de abstracción de lo aprendido por el estudiante con los lenguajes de alto nivel, lo trasladan a una situación donde el conocimiento adquirido no resulta de mucha utilidad, produciendo un efecto de frustración y desmotivación. Son muchos los problemas que debe enfrentar y solucionar el estudiante ante esta nueva situación. La naturaleza primitiva de las instrucciones del ensamblador muchas veces requiere de una buena cantidad de éstas para realizar tareas muy simples. Las interfaces de entrada/salida, intrínsecamente poco intuitivas, requieren entender y memorizar su comportamiento exacto para una programación exitosa. Los errores lógicos son muy frecuentes y provocan comportamientos poco predecibles del programa. La depuración de un programa debe hacerse normalmente mediante prueba y error, un proceso que es necesariamente tedioso debido a la naturaleza primitiva de las instrucciones. Estos aspectos de la programación del lenguaje ensamblador muchas veces aburren, abruman y/o desaniman a los estudiantes, desmotivándolos y haciendo aún más difícil el proceso de aprendizaje. Por esto es de fundamental importancia contar con herramientas que permitan aplicar de forma práctica y tangible los conceptos teóricos aprendidos y de esta forma mantener el incentivo que necesita el estudiante para avanzar. Herramientas como simuladores que permiten visualizar la interacción de los subsistemas que componen una computadora a medida que un programa se ejecuta son imprescindibles para asistir al alumno en el proceso de aprendizaje. Muchas veces es difícil encontrar recursos y estrategias para mantener la motivación del estudiante y minimizar la carga negativa que conlleva todo lo relacionado a la programación en lenguaje ensamblador.

En este sentido, la programación de juegos ofrece una serie de oportunidades y ventajas para superar el aprendizaje de los lenguajes de bajo nivel y así librar toda esta connotación negativa que los rodea. La idea de programar juegos en cualquier lenguaje de programación motiva no sólo a los alumnos durante el proceso de aprendizaje, sino también a los docentes en el proceso de enseñanza, permitiendo además, desarrollar conocimientos de manera más eficiente [9]. Existen experiencias realizadas en diferentes niveles educativos que avalan esta estrategia. Por ejemplo, en el nivel secundario hay experiencias exitosas de aprendizaje en lenguajes de alto nivel [9, 12]. Lo mismo sucede con la programación de juegos a nivel universitario tanto cuando se realiza en un lenguaje de alto nivel [8] como en lenguaje ensamblador [1, 2, 7]. En particular, en una experiencia realizada con estudiantes que participaron del desarrollo de juegos se obtuvieron resultados muy interesantes[10]. Respecto de los aspectos curriculares los estudiantes reportaron que pasaron más tiempo cursando, aprendiendo algoritmos y conceptos de programación, escribiendo comentarios en el código y mejorando sus habilidades en la resolución de problemas. Respecto de los aspectos extracurriculares reportaron que pasaron más tiempo realizando animaciones y estimulando su creatividad, pero sin duda, lo más importante de destacar fue que todos afirmaron que disfrutaron de la programación.

En este trabajo se presenta la primera parte de un proyecto que tiene como objetivo motivar a los estudiantes universitarios de primer año en la programación en lenguaje ensamblador a través del desarrollo de un juego simple. Este hilo conductor, especialmente motivador, tiene el potencial para afianzar y aumentar el conocimiento de los estudiantes sobre los temas vistos en la práctica. Por un lado, porque persiguen un objetivo atractivo y divertido para muchos programadores como es el de diseñar un videojuego y por otro lado, porque les brinda

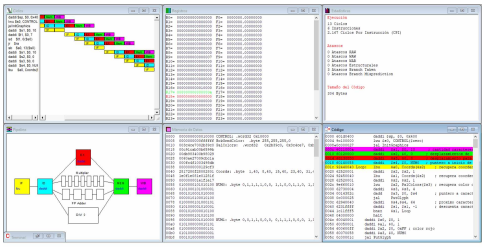

Figura 1: Simulador WinMips64

la oportunidad de desarrollar un costado creativo y los incentiva para adquirir conocimientos que trascienden la asignatura. También se describe la adaptación del software de simulación WinMips64, usado actualmente en la asignatura, que incorporó la funcionalidad necesaria para desarrollar juegos y animaciones interactivas. Finalmente, se muestran los resultados obtenidos de los cambios realizados al simulador y de la experiencia vivida por los estudiantes sobre los juegos que desarrollaron. Además, se presenta la propuesta para el desarrollo de la segunda parte del proyecto que extiende la experiencia a una parte mayor del alumnado.

2. Contexto

2.1. Arquitectura de Computadoras

Arquitectura de Computadoras es una asignatura del segundo semestre del primer año de las carreras de Licenciatura en Sistemas y Licenciatura en Informática de planes de estudio 2003, 2007, 2012 y 2015, de Analista Programador Universitario plan 2007 y 2015 y de Analista en TIC plan 2017 de la Facultad de Informática de la Universidad Nacional de La Plata, donde anualmente cursan alrededor de 800 estudiantes.

En la teoría de la asignatura se ven conceptos de organización y arquitectura de computadoras [6, 4, 13], el funcionamiento en detalle de los distintos subsistemas (CPU, memoria, entrada/salida, buses de comunicación), procesamiento paralelo y paralelismo a nivel de instrucción. En la parte práctica se aplican los conceptos vistos en teoría y se complementa con el aprendizaje de la programación en lenguaje ensamblador. Para el desarrollo de todos los trabajos prácticos se utilizan dos simuladores que visualizan la interacción de los distintos subsistemas de una computadora a partir de la ejecución de un programa. Uno de los simuladores está basado en la arquitectura CISC Von Neuman de los procesadores X86 mientras que el otro está basado en la arquitectura RISC Harvard de los procesadores MIPS.

2.2. Simulador WinMips64

La aplicación original de WinMips64 (figura 1) fue desarrollada por Mike Scott y se la puede encontrar en el sitio [11]. Es una herramienta muy completa para aprender tanto la arquitectura y el funcionamiento del procesador MIPS como la programación del lenguaje ensamblador.

De las funcionalidades que incorpora esta aplicación, una de las más interesantes es la simulación del cause del procesador. Esta visualiza la evolución de las instrucciones de un programa en cada una de las etapas en las que se divide el cauce. La ejecución de un programa puede realizarse con diferentes niveles de detalle, posibilitando el avance de la simulación de un ciclo de reloj, de un ciclo de instrucción completo o del programa completo. Posee herramientas para visualizar el estado de los registros, de la memoria de datos, de la memoria de programas, para realizar el seguimiento estadístico de ciclos de ejecución y los distintos tipos de atascos que van sucediendo. También permite establecer puntos de ruptura para detener la ejecución de los programas y así facilitar su depuración. Adicionalmente proporciona un mecanismo de invocación para la ejecución de operaciones elementales de entrada/salida que permiten leer una tecla, imprimir un número o una cadena en una consola de texto y pintar píxeles en consola gráfica. Particularmente este grupo de operaciones es la que hace que los programas desarrollados en WinMips64 sean más interesantes al permitir la interacción con un usuario.

3. Desarrollo

3.1. Motivación

El origen de este proyecto surge a partir de la inquietud por parte de varios alumnos que quisieron avanzar más allá de los contenidos de la práctica de la asignatura. Motivados por la posibilidad de dibujar en el simulador WinMips64, intentaron implementar una pequeña animación que se vio frustrada por varios problemas del funcionamiento de la precaria consola gráfica. Después de una interesante e informal charla entre docentes y estudiantes surgió la idea de adaptar el simulador de forma tal que permitiera la implementación de pequeños juegos. De esta manera se busca impulsar la motivación en los estudiantes para lograr la consolidación de los aspectos de la programación en lenguaje ensamblador vistos en la práctica, y porque no, ir aún más lejos para adquirir conocimientos más allá de los curriculares.

3.2. Propuesta

Para el desarrollo de este proyecto se planificaron dos etapas de las cuales, al momento de la redacción de este documento, se ha finalizado la primera.

Parte de esta primera etapa consistió en la adaptación del entorno de simulación WinMips64 para el desarrollo de juegos simples. Esto incluyó por un lado la depuración y mejora de las funcionalidades ya existentes y por otro lado el desarrollo de un mínimo de nuevas funcionalidades, necesarias para posibilitar la programación de juegos. Los detalles del análisis y el desarrollo de las modificaciones del software junto con el detalle de las dificultades resueltas se presentan en la siguiente sección. También se realizó una experiencia con un grupo reducido de estudiantes para tener una retroalimentación sobre el proceso del desarrollo de los juegos.

La experiencia a realizar en la segunda etapa se llevará a cabo sobre el final del cuatrimestre en una fecha cercana al cierre de la cursada de la asignatura, momento en el cual los estudiantes estarán en condiciones de haber adquirido el conocimiento necesario para desarrollar programas en WinMips64. La misma involucra la participación de un grupo importante de estudiantes que desarrollarán un pequeño programa de un videojuego o una animación. Estos contarán con una guía sobre cómo desarrollar un pequeño juego, pautas sobre la forma de escribir el programa, herramientas adicionales para el desarrollo, y un conjunto de juegos de ejemplo para analizar, consultar e inspirarse. El nivel de conocimiento de los estudiantes será evaluado antes y después del desarrollo del programa para medir el nivel de evolución del aprendizaje. Los docentes acompañarán a los estudiantes durante todo el proceso y participarán de las evaluaciones.

3.3. Adaptación del Simulador para el Desarrollo de Videojuegos y Animaciones

3.3.1. Elección de WinMips64

La decisión de utilizar WinMips64 como punto de partida para el desarrollo de juegos se basó principalmente en tres razones. La primera y principal fue que hace varios años que en la asignatura se trabajan varios temas de la práctica con éste simulador. Es una aplicación bien conocida tanto por docentes como por estudiantes y su uso es natural para ambos, por lo que no fue necesario introducir una nueva herramienta evitando de esta forma todas las desventajas que esto conlleva. La segunda razón fue que el simulador cuenta una consola gráfica y la primitiva para pintar un píxel en pantalla ya implementada, algo que resolvía una parte del trabajo que debíamos hacer y que aceleró el proceso de desarrollo. La tercera razón fue que el simulador implementa una interfaz de comunicación para las funciones de entrada/salida. Este mecanismo facilita el desarrollo de nuevas funciones siguiendo la dinámica de lo aprendido en la práctica sin agregar una nueva dificultad para los estudiantes.

3.3.2. Análisis y Desarrollo de las Funcionalidades Mínimas y Necesarias

Para la adaptación del simulador no se utilizó la versión original de Mike Scott sino que se realizó sobre una versión modificada por Andoni Zubimendi [14] que ha corregido varios errores y ha realizado la traducción al idioma español.

Primeramente se realizó un análisis de la consola gráfica con la que cuenta el simulador. Ésta tiene una dimensión de 50x50 píxeles RGB donde cada píxel lógico ocupa una área de 10x10 píxeles en la pantalla real con el fin didáctico de identificarlos fácilmente. En las pruebas realizadas se observó que la dimensión de la pantalla era algo pequeña por lo que se decidió incrementar la dimensión a 64x64 píxeles (un 60 % más) para poder incluir algo más de detalles. Por otro lado se redujo el tamaño gráfico del píxel a un área de 5x5 para una mejor percepción de la integridad de los objetos dibujados. Ambas modificaciones fueron hechas para mejorar la calidad gráfica sin perjudicar el rendimiento general de la simulación para el desarrollo del juego.

Se encontró que la implementación de la operación de entrada/salida para dibujar un píxel tenía dos problemas: uno era la lentitud con la que se dibujaba, provocando que la operación de borrado completo de la pantalla tardara decenas de segundos; el otro problema era el parpadeo (flickering) que producía la consola al dibujar cada píxel, un efecto que tornaba la animación en una experiencia tanto incómoda como poco fluida. A partir del resultado de este análisis se decidió implementar una función de pintado del píxel más eficiente para solucionar ambos problemas. Otro de los resultados del análisis encontró un inconveniente en la función de entrada/salida para leer una tecla. Esta función era bloqueante, es decir que suspendía la ejecución del programa hasta que el usuario presione una tecla. Este comportamiento imposibilita la realización de una animación en la pantalla si además se desea interactuar a través del teclado. Para resolver este problema se agregó una función adicional que retorna un valor de tecla inválido si no hay una tecla presionada y devuelve el control al programa de manera inmediata. Del análisis de funcionalidades adicionales para implementar se decidió incorporar una función de entrada/salida para generar números aleatorios. Esto es de fundamental importancia para incorporar un componente de azar en cualquier juego. Otra modificación planteada fue la incorporación de una funcionalidad para medir el transcurso del tiempo con cierta precisión. Si bien esta funcionalidad es útil para realizar rutinas de sincronización y de espera precisas, se postergó su implementación para una próxima etapa ya que no se consideró una función crítica.

En una de las varias iteraciones con los participantes, luego de haber generado los primeros programas, se encontraron problemas debido a la limitación de memoria en la simulación. Una cantidad insuficiente de bits de los buses de datos y programa que definía el simulador provocaba problemas cuando se incorporaban muchos gráficos, gráficos grandes o programas largos. Como solución de estos problemas se decidió extender el límite de la cantidad máxima de bits del bus de direcciones de los datos y de programas.

Es importante destacar que todas las modificaciones e incorporaciones de funcionalidades de entrada/salida fueron las mínimas y necesarias para incorporar al simulador la capacidad requerida por el desarrollo de juegos y animaciones interactivas. Cabe destacar que se mantuvo la misma filosofía y el mecanismo de invocación que utilizan las funciones ya definidas en el simulador, manteniendo de esta manera la transparencia en el uso de las nuevas funciones para los estudiantes.

4. Pruebas y Resultados

Finalizadas las reformas detalladas en la sección anterior, se procedió a distribuir la nueva versión del simulador entre un grupo 8 alumnos y 4 docentes para tener una retroalimentación de su funcionamiento en 4 aspectos diferentes. El primero era comprobar el correcto funciona-

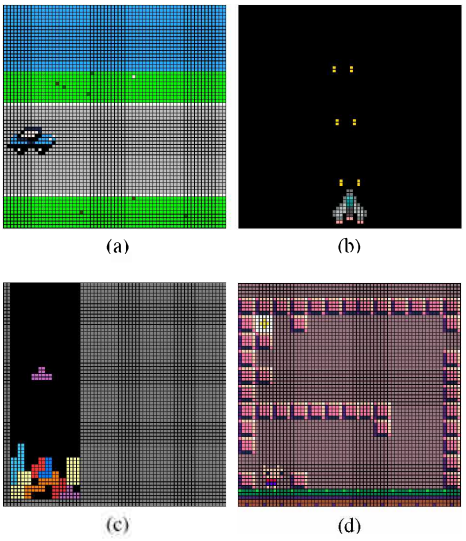

Figura 2: Juegos y animaciones implementados por los alumnos.

miento de la integración de las nuevas funcionalidades al simulador. El segundo era determinar si las nuevas funciones eran suficientes para el desarrollo de juegos o si era necesario incorporar funciones adicionales. El tercer aspecto consistió en obtener varios programas que pudieran ser utilizados por los estudiantes como ejemplos para el desarrollo de juegos en la segunda etapa del proyecto. Por último se buscó obtener una devolución sobre la experiencia desarrollada tanto por estudiantes como por docentes.

La propuesta para los participantes fue crear un programa que implementara un sencillo juego con una lógica simple o una pequeña animación interactiva que utilizara las teclas. Los estudiantes eligieron libremente el juego a programar y si lo iban a desarrollar de manera individual o grupal. La idea fue darles autonomía de acción para que no se sintieran presionados y pudieran desarrollar una aplicación acorde a sus conocimientos y tiempo disponible, intentando mantener en todo momento la motivación, el entusiasmo y el sentido de diversión que inspiraba la propuesta. Se establecieron pautas sobre la forma de escribir los programas, de acuerdo a lo visto en las clases de práctica. Entre estas figuraban no usar variables globales al menos que cumplan la función de una constante, uso de subrutinas con parámetros, uso de convención de nombres y registros según compilador de C, implementación de la pila y la recomendación de documentar y comentar el código escrito.

Respecto de los resultados obtenidos, hay varios aspectos a destacar. El comportamiento general de las nuevas funciones de entrada/salida implementadas se desarrolló sin problemas y tampoco se encontró necesario incorporar nuevas funciones para desarrollar los juegos.

Los docentes evaluaron los programas realizados por los estudiantes y destacaron la organización del código escrito en lenguaje ensamblador. Además confirmaron tanto el cumplimiento de las pautas propuestas como la confirmación del uso del conocimiento adquirido a través de las clases de práctica. Los alumnos por su parte expresaron que la experiencia fue positiva, remarcando la motivación, la diversión y la utilidad de los conceptos aprendidos. Incluso algunos se ofrecieron para desarrollar más ejemplos o extender los ya realizados luego de finalizada la cursada de la asignatura.

Respecto de los programas desarrollados por los estudiantes se obtuvieron dos programas de animaciones interactivas y dos programas de juegos cuyas capturas de pantalla pueden verse en la figura 2. En la figura 2a se muestra la animación de un auto que se mueve por una carretera mientras que en la figura 2b se muestra la animación de una nave que se mueve por el espacio. Respecto de los juegos realizados, en la figura 2c se muestra la implementación de una versión del Tetris con algunos detalles pendientes de implementación. El último de los juegos se muestra en la figura 2d y consiste en mover un personaje a través de una especie de laberinto hasta llegar a su destino. Cabe destacar que esta implementación contiene 7 escenarios diferentes que transcurren a medida que el personaje alcanza cada salida.

Algunos de los aspectos donde los alumnos tuvieron dificultad fue en la implementación de rutinas relativamente largas y/o relativamente

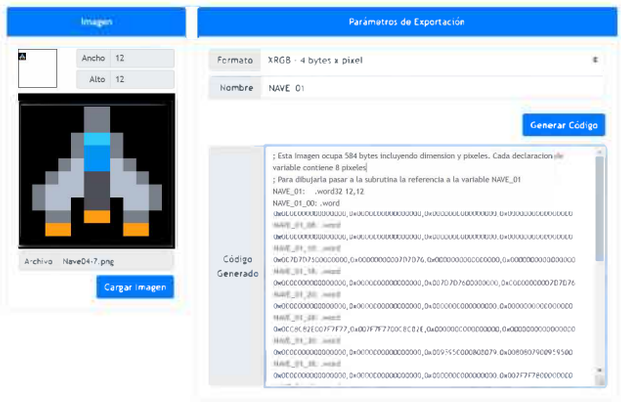

Figura 3: Conversor de Imagen a código para WinMips64

complejas. Puntualmente se presentaron dificultades al momento de dibujar una imagen en la consola gráfica, un tipo de problema que no aporta conocimientos relevantes a la experiencia y es mayormente desmotivador. A fin de minimizar estos efectos negativos, para la experiencia de la segunda etapa del proyecto se ha planificado un análisis de las rutinas complejas que requiere la implementación de los juegos. Estas rutinas serán implementadas y quedarán en un repositorio a disposición de los alumnos para puedan usarlas directamente. Otro aspecto que presentó dificultades fue la generación de la estructura que debe tener una imagen dentro del código del programa para que pueda ser dibujada en la consola gráfica. Si bien hay varias herramientas en-línea para la conversión de una imagen en código C o en una secuencia de bytes, la adaptación de este formato al que requiere el WinMips64 resultó tediosa e incorporó errores algo difíciles de depurar. Para solventar este problema se decidió desarrollar una pequeña aplicación que permite obtener a partir de una imagen el código fuente con la estructura adecuada para incrustar en el cuerpo del programa. En la figura 3 se muestra la interfaz de esta aplicación. Al momento de escribir este documento ya se encuentra disponible su primera versión.

5. Conclusiones y Trabajos Futuros

El aprendizaje de la programación a través del desarrollo de juegos ofrece a los estudiantes una oportunidad de adquirir y afianzar sus conocimientos con mayor motivación y de forma divertida y relajada. Este es el hilo conductor del proyecto que se presenta en este artículo.

Como conclusión hay varios aspectos interesantes de destacar. En esta primera etapa de desarrollo, respecto del material generado cabe mencionar cuatro cosas. En primer lugar, se ha extendido la funcionalidad del simulador WinMips64 para poder realizar programas que implementan juegos sencillos en lenguaje ensamblador. Este software con nuevas funciones fue utilizado por un grupo de docentes y estudiantes con el objeto de poner a prueba la idea propuesta y verificar la ausencia de errores. Segundo, se han desarrollado programas que implementan dos animaciones interactivas y dos juegos. Éstos quedan disponibles a modo de ejemplo para que los estudiantes puedan analizar su estructura e implementación y sirvan como inspiración y guía para desarrollar nuevos. Tercero, se ha desarrollado una pequeña aplicación que asiste a los programadores en la generación del código necesario para incrustar imágenes en el formato adecuado dentro de los programas WinMips64. Finalmente, todo el desarrollo realizado por estudiantes y docentes se encuentra disponible en un repositorio Github [3] de forma libre, incluyendo los fuentes modificados en WinMips64, los programas de ejemplo y la aplicación de conversión de imágenes.

Respecto de la experiencia realizada, tanto docentes como estudiantes se mostraron motivados y entusiasmados con el desarrollo de los juegos. Esta experiencia sirvió para detectar algunas falencias relacionadas con la dificultad intrínseca de algunas subrutinas y los problemas para incorporar la estructura de las imágenes dentro del código de los programas. El segundo problema fue resuelto con el desarrollo de la aplicación de conversión de imágenes mientras que el primero está en vías de solucionarse.

Como parte del trabajo pendiente de realizar queda planteado para la próxima etapa el desarrollo de un tutorial para mostrar tanto las nue-

vas funcionalidades del simulador como aspectos necesarios para el desarrollo de juegos sencillos. También se encuentra pendiente el análisis para determinar un conjunto de subrutinas de mayor dificultad y su implementación para que estén disponibles para los estudiantes. Finalmente, la segunda parte del proyecto queda pendiente de realización hasta el final de la cursada del próximo cuatrimestre donde se realizará una experiencia que comprenda a una mayor cantidad de alumnos. En dicha etapa se realizará un prueba diagnóstico previa al desarrollo y luego los programas realizados por los estudiantes serán evaluados por los docentes con el objetivo de medir la efectividad de los conceptos aprendidos antes y después del desarrollo del juego.

Referencias

[1] M. E. Acacio, L. Fernández-Maimó, R. Fernández-Pascual, P. González-Férez, A. Ros, and R. Titos-Gil. Proyecto tetris: aprendizaje de la programación en ensamblador por piezas. *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática*, 4:279–286, 2019.

[2] A. Burguera Burguera and J. Guerrero Sastre. Lenguaje ensamblador en el siglo XXI: Desarrollo de videojuegos como elemento motivador. *Revista de Investigación en Docencia Universitaria de la Informática*, 7, 2014.

[3] C. Estrebou. Simulador WinMips64 para juegos. `https://github.com/cesarares/WinMIPS64`, 2021. Último acceso 01/04/2021.

[4] C. Hamacher, Z. Vranesic, S. Zaky, and N. Manjikian. *Computer Organization and Embedded Systems*. Mc Graw Hill, 6th edition, 2012.

[5] S. Helme and D. Clarke. Identifying cognitive engagement in the mathematics classroom. *Mathematics Education Research Journal*, 13:133–153, 2001.

[6] J. Hennessy and D. Patterson. *Computer Architecture. A Quantitative Approach*. Morgan Kaufmann, 5th edition, 2011.

[7] J. Kawash and R. Collier. Using video game development to engage undergraduate students of assembly language programming. In *Proceedings of the 14th Annual ACM SIGITE Conference on Information Technology Education*, SIGITE '13, page 71–76, New York, NY, USA, 2013. Association for Computing Machinery.

[8] G. Kiss and Z. Arki. The influence of game-based programming education on the algorithmic thinking. *Procedia - Social and Behavioral Sciences*, 237:613–617, 2017. Education, Health and ICT for a Transcultural World.

[9] I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar, and S. Lahmine. Learning basic programming concepts by creating games with scratch programming environment. *Procedia - Social and Behavioral Sciences*, 191:1479–1482, 2015. The Proceedings of 6th World Conference on educational Sciences.

[10] D. Ozoran, N. Cagiltay, and D. Topalli. Using scratch in introduction to programming course for engineering students. In *2nd International Engineering Education Conference (IEEC2012)*, volume 2, pages 125–132, 2012.

[11] M. Scott. Winmips64 simulator. `http://indigo.ie/~mscott`, 2012. Último acceso 01/04/2021.

[12] W. Y. Seng and M. H. M. Yatim. Computer game as learning and teaching tool for object oriented programming in higher education institution. *Procedia - Social and Behavioral Sciences*, 123:215–224, 2014.

[13] W. Stallings. *Computer Organization and Architecture. Designing for Performance*. Pearson, 10th edition, 2016.

[14] A. Zubimendi. Simulador WinMips64. `https://github.com/AndoniZubimendi/WinMIPS64`, 2020. Último acceso 01/04/2021.