Trabajo de Fin de Grado

Programación en estudios pre-universitarios. Oportunidades, retos y caso de estudio

Programming in pre-university studies. Opportunities, challenges and case study

Borja Barrera Villagrasa

La Laguna, 5 de septiembre de 2016

Dª. **Coromoto León Hernández,** con N.I.F. 78.605.216-W profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

C E R T I F I C A

Que la presente memoria titulada:

"Programming in pre-university studies. Opportunities, challenges and case study."

ha sido realizada bajo su dirección por D. **Borja Barrera Villagrasa,** con N.I.F. 45.710.042-A.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firma la presente en La Laguna a 5 de septiembre de 2016.

Agradecimientos

En primer lugar a mis padres por darme la gran oportunidad de estudiar una carrera y por los ánimos que me han dado en todo momento.

En segundo lugar a mi tutora Coromoto León Hernández por su excelente trato y su gran ayuda.

Y en tercer lugar a mi pareja, por apoyarme en los buenos y malos momentos.

Licencia

Resumen

El objetivo de este trabajo es realizar un estudio comparativo de herramientas de programación visual que se utilizan en la enseñanza pre-universitaria; llevando como hilo conductor el mismo caso de estudio de un algoritmo de ordenación.

Esta memoria cuenta con siete capítulos. En el primero se realiza un análisis de la situación de la enseñanza de la programación en estudios pre-universitarios. A continuación, en el segundo capítulo se realiza una comparativa de distintas herramientas con el fin aportar distintas opciones para el aprendizaje de la programación. En el tercer capítulo se procede a realizar un caso de estudio donde se ilustrará un ejemplo sencillo de ordenación, seleccionando una serie de herramientas que van a ir incrementando el nivel de dificultad cuando se implemente el algoritmo ya que cada una de estas herramientas esta especificada para un intervalo de edades. En el siguiente capítulo, el cuarto, se mostraran los resultados obtenidos del estudio realizado a cada una de las herramientas utilizadas para la implementación del algoritmo de ordenación. En el capítulo cinco se redacta las conclusiones obtenidas de este trabajo, en el seis se traducirá al inglés esas conclusiones y en el capítulo siete se realizará un presupuesto desglosado de la realización de cada una de las tareas propuestas para la ejecución de este trabajo.

Palabras clave: Programación, educación, herramientas, enseñanza-aprendizaje

Abstract

The objective of this work is to carry out a comparative study of visual programming tools that are used in teaching pre-university education; taking as a common thread the same case study of a sort algorithm.

This memory has seven chapters. In the first it is made an analysis of the situation of the teaching of the programming in pre-university studies. Then in the second chapter is made a comparison of different tools in order to provide different options for the learning of the programming. In the third chapter proceeded to perform a case study where we illustrate a simple example of sorting, selecting a series of tools that are going to be increasing the level of difficulty when it implements the algorithm because each one of these tools this specified for a range of ages. In the next chapter, the fourth, is to show the results of a study performed on each one of the tools used for the deployment of the sort algorithm. In chapter five was drawn up the conclusions obtained from this work, in the six translated into English these conclusions, and in chapter seven there will be a budget breakdown of the performance of each of the proposed tasks for the execution of this work.

Keywords: *Programming, education, tools, teaching-learning*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Vivimos en una sociedad que tiende a confundir los conceptos de "Tecnologías de la información y la Comunicación" y "Ciencias de la computación" relacionando erróneamente sus definiciones y términos, así mismo pretenden englobar estos conceptos en un término más general como es la Informática. Es por ello fundamental aclarar estos términos con el fin de que las personas adquieran un conocimiento más amplio sobre esta rama del conocimiento.

El primer concepto que definiremos es *Informática*, que según el diccionario de la RAE (Real Academia Española)[1] se trata de "Un conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras".

Otro de los términos es el de *Ciencias de la Computación* que la vamos a definir como "Aquellas que abarcan el estudio de las bases teóricas de la información y la computación, así como su aplicación en sistemas computacionales"[2], si lo queremos expresar de otra manera, "Es el estudio de los fundamentos teóricos sobre la información y la computación y su aplicación", dicho estudio comprende entre otros conceptos la programación que la definiremos como "un conjunto de técnicas utilizadas para desarrollar programas que sean fáciles de leer, depurar y modificar o mantener"[3] y el *Pensamiento Computacional* que implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática.

Otros de los conceptos que no podemos confundir son las actividades que no pertenecen a las ciencias de la computación como pueden ser la alfabetización digital que son acciones formativas dirigidas al desarrollo de habilidades técnicas, sociales y éticas relativas al uso

[1] http://www.rae.es/
[2] Enciclopedia de las ciencias de la computación de ACM
[3] Lógica de la Programación - Efraín M. Oviedo Regino

de las TIC, la seguridad online que trata de la navegación segura, así como el intercambio de información en redes sociales, la electrónica y robótica que es el diseño y la construcción de circuitos electrónicos y/o robots para resolver problemas, y el uso de dispositivos en la educación.

Una vez aclarados estos términos vamos a empezar a argumentar porque hay que inicializar en edades tempranas el pensamiento computacional y el uso de la programación.

Últimamente ha surgido una corriente muy fuerte que defiende que, en los colegios, debería enseñarse a programar. Dicho movimiento, además, se acerca bastante a ser global, gracias a ello existe una frase que lleva mucho tiempo repitiéndose, sobre todo en aquellas casas donde hay hijos: "la informática tiene mucha salida". Estudiar alguna disciplina relacionada con la informática está razonablemente bien visto en muchas sociedades y hay numerosos informes que indican que no hay demasiado desempleo entre profesionales del ramo de las nuevas tecnologías. Es por ello que muchos padres quieren introducir a sus hijos a esta disciplina cuanto antes mejor. Además, mejorar la alfabetización digital es el sexto de los siete pilares de la Agenda Digital Europea, donde se estima que en los próximos años va a existir una gran demanda de trabajadores relacionados con las tecnologías de la información y la comunicación y que, sin la formación adecuada, quedarán sin cubrir. Es por ello importante acercar la programación a los adolescentes y comenzar a formar en ellos el razonamiento computacional, es decir, generar soluciones con pensamiento algorítmico.

Si observamos el mapa europeo, veremos que en países como Estonia nacen proyectos que tiene por finalidad enseñar a sus niños entre siete y diecinueve años a escribir código. En otros como en Finlandia será obligatorio que todos los alumnos de primaria aprendan a programar a partir del curso 2016/2017 o como en Alemania que hay tres regiones que enseñan a programar desde primaria: Saxonia, Mecklenburg-Western Pomerania y, especialmente, Bavaria, otro buen ejemplo es el Reino Unido, han empezado a repartir un millón de BBC micro::bit, los pequeños miniordenadores que permiten a los alumnos iniciarse en este mundillo. Sin

contar con las Raspberry Pi, que forman parte de diversos proyectos educativos en ese país.

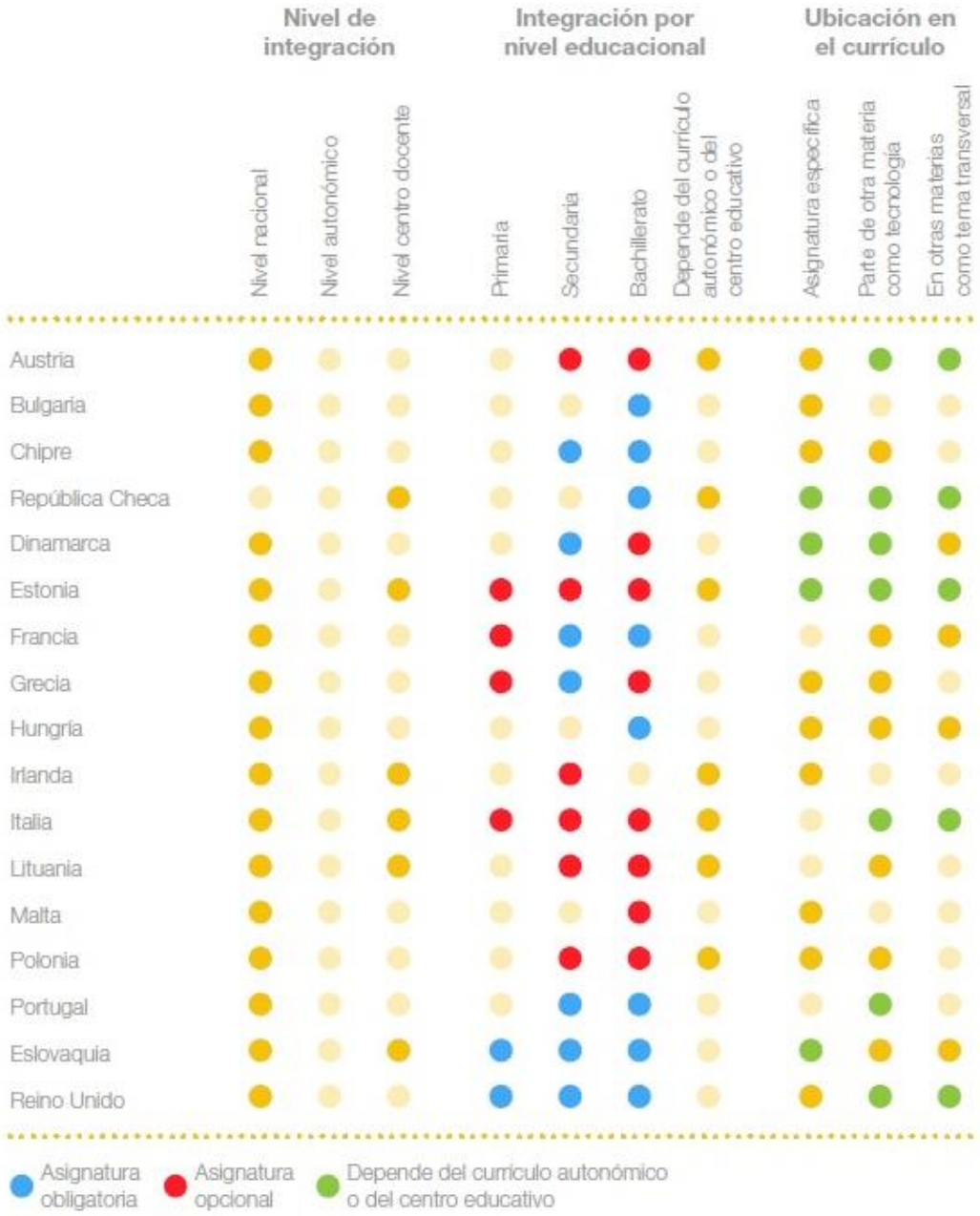

Figura 1.1. Resumen del nivel de integración de las CC en el currículo escolar de 18 países de la UE y el nivel educativo en el que se imparte esta disciplina[4][5].

[4] Computing Our Future – Computing, Programming and Coding – Priorities, School Curricula and Initiatives Across Europe, European Schoolnet 2015.

[5] Computing Our Future – Computing, Programming and Coding – Priorities, School Curricula and Initiatives Across Europe, European Schoolnet 2014

Sin embargo, ¿Cuál es la situación en España? En nuestro país hay un buen número de docentes que lleva mucho tiempo enseñando a programar a su alumnado, pero hasta el momento se trataban de iniciativas individuales que no estaban promovidas desde las administraciones públicas. A pesar de ello, se están comenzando a dar pasos en este sentido y ya hay varias Comunidades Autónomas que quieren generalizar estas enseñanzas y que han comenzado a impartir formación a sus docentes y a desarrollar distintas iniciativas.

Aun así en nuestro país las Ciencias de la Computación son un concepto desconocido en la sociedad el 82% de padres y madres no saben qué englobaban o las confundían con otros términos, algo que también les ocurría al 76% de los alumnos de entre 12 y 16 años, además la introducción de dispositivos digitales para programar es por ejemplo muy reducida: solo el 5% de los alumnos disponían de tales soluciones en cursos de primaria, mientras que en secundaria el porcentaje ascendía al 16%.

Figura 1.2. Encuesta realizada en octubre de 2015 participaron 2.324 padres y madres con hijos e hijas de distintos rangos de edad, a lo que se sumó la realización de 30 entrevistas con directores y docentes de distintos colegios de España. El informe fue elaborado en colaboración con Google y everis[6].

[6] http://www.everis.com/spain/WCLibraryRepository/References/everis_InformeGoogle_210x297_RGB_7%C2%AApres.pdf

Ante esta situación podemos preguntarnos, ¿Por qué programar desde edades tempranas?

La realización de proyectos en los que se incluya la programación permiten al alumnado aprender haciendo, desarrollar el pensamiento lógico y algorítmico, solucionar problemas de forma creativa, crear trabajos complejos a partir de sus propias ideas, trabajar de forma autónoma y en colaboración, comprender el funcionamiento del entorno tecnológico en el que vive y, en definitiva, ayuda a alcanzar las competencias básicas necesarias para desenvolverse en un mundo cambiante y tecnológicamente avanzado.

Por tanto, ¿Cómo conseguimos abordar la programación en la educación pre-universitaria?

Existen una gran diversidad de programas educativos que, a pesar de tener unos rasgos esenciales básicos y una estructura general común se presentan con unas características muy diversas: unos aparentan ser un laboratorio o una biblioteca, otros se limitan a ofrecer una función instrumental del tipo máquina de escribir o calculadora, otros se presentan como un juego o como un libro, bastantes tienen vocación de examen, unos pocos se creen expertos... y, por si no fuera bastante, la mayoría participan en mayor o menor medida de algunas de estas peculiaridades.

Las herramientas educativas, se definen de forma genérica como aplicaciones o programas computacionales que faciliten el proceso de enseñanza aprendizaje. Algunos autores lo conceptualizan como cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar.

Una de las herramientas más utilizadas es ALICE[7] que existe como soporte para la enseñanza de materias como puede ser la programación en la educación pre-universitaria.

[7] http://www.alice.org

En un artículo de Jun Hu et.al. en [1] hace referencia al programa ALICE en la cultura de la computación hablando de que el paradigma de la cultura computacional existen diferentes culturas con enfoques diferentes para abordar determinantes culturales que influyen mucho en nuestra forma de pensar.

Para la cultura occidental, una respuesta podría ser la necesidad de una instalación artística e interactiva (ALICE). Con ello, el usuario, de principio a fin, se somete a un entorno inmersivo que integra los agentes incorporados y virtuales.

Hay que destacar también unos puntos fuertes que ALICE proporciona para el aprendizaje con la intención de que los alumnos adquieran los conceptos necesarios sobre materias como la computación, es el uso de plantillas de juegos. Un método que puede llegar a ser eficaz gracias a la facilidad que aporta su aprendizaje tanto por parte del profesorado como por parte del alumnado.

En este punto hacemos referencia a otro artículo, en este caso al de Jane Nawrocki et.al. en [2] donde habla sobre las plantillas de juegos en una herramienta como ALICE para que los profesores de cualquier área del conocimiento puedan transmitir más fácilmente sus ideas y con ello los alumnos puedan consolidar dichas habilidades.

En el artículo se menciona a autores como Marzano, Pickering, y Pollock que identifican una serie de estrategias de enseñanza que promueven el aprendizaje de los estudiantes y que para cada estrategia, también sugieren formas para integrar la tecnología para aumentar el proceso de aprendizaje.

Otro autor como Marc Prensky deja claro que el juego es ahora una forma de vida. “Hoy en día los escolares, de la escuela primaria a la universidad, viajar con su propio Juego de los Niños,..., teléfonos celulares, portátiles de CD y reproductores de MP3, buscapersonas, ordenadores portátiles y conexión a Internet." Era sólo una cuestión de tiempo que los videojuegos llegaran a la escuela. Además, los juegos son versátiles y pueden ser utilizados eficazmente para enseñar, revisar, y/o reforzar cualquier materia o habilidad.

Según el autor, después de que se experimentara con la herramienta ALICE se comprobó que se podía utilizar para la enseñanza como una estrategia de enseñanza ya que es una herramienta ideal para crear plantillas de juegos educativos que pueden llegar a ser un atractivo para que el alumnado adquiera los conceptos necesarios, como por ejemplo permitiendo crear juegos en tres dimensiones que sería beneficio para la enseñanza de la programación, sin necesidad de que se tengan conocimientos de programación avanzada, también podrían utilizarse los juegos para ilustrar los conceptos de la programación en el curso.

Quizás una vez que los profesores sean testigos de la interesante mezcla de aprendizaje y diversión que los juegos pueden ofrecer, buscarán juegos adicionales cuidadosamente diseñados o desarrollar nuevos juegos en sí.

A parte de ALICE existen un sinfín de herramientas con el mismo objetivo de las cuales destacamos a parte de la mencionada a:

Scratch[8], que tiene más ventajas que inconvenientes a la hora de usarlo en la educación, ya que su aplicación puede ser amplia.

Uno de los ejemplos de uso de esta herramienta lo podemos encontrar en un trabajo de fin de master que se realizó en la facultad de educación de la Universidad Internacional de la Rioja donde nos habla de que "Los alumnos con un diagnóstico TDAH generalmente presentan serias dificultades para adaptarse a la dinámica normal de una clase, que en muchas ocasiones implica periodos de tiempo prolongados para realizar una actividad, mantener la atención o seguir indicaciones. Tareas como éstas para un alumno con TDAH son probablemente las más difíciles de realizar, por tanto, una herramienta que permita al alumno, por una parte, autonomía en la gestión del tiempo, y por otra, facilidad en su uso, además de un aporte motivacional que le permita centrar sus esfuerzos y superar las distracciones externas, podría ser una herramienta básica en las nuevas metodologías dirigidas a este colectivo de alumnos. El lenguaje de programación como parte integrante de la informática creativa permite fusionar el uso de la

[8] http://www.scratch.mit.edu

creatividad, la imaginación y las motivaciones académicas y lúdicas dando lugar a la educación personalizada y significativa. El Scratch forma parte de este lenguaje de programación."[Duby Loscari Aldana Avilés - Julio de 2015]

Como se observar, gracias a esta información, podemos comprobar lo eficiente y útil que puede llegar a ser las herramientas como Scratch, incluso para alumnos con diagnósticos como el TDAH (trastorno por déficit de atención con hiperactividad), cuando se pretende que enfoquen su atención a lo que se le pretende transmitir, motivándolos de una forma diferente a la convencional.

Los conocimientos educativos que adquieren los alumnos con esta aplicación es que el aprendizaje y experiencia educativa que aporta la herramienta es cuanto menos, enriquecedora. No sólo porque éstos aprenden el funcionamiento, aunque simple, de la programación informática, sino porque, además, aprenden a trabajar con ordenadores de una forma muy sencilla evitando, en muchas ocasiones, el aprendizaje frustrado que puede provocar el trabajar con ordenadores al principio.

En cuanto a su aplicaciones en el mundo de la educación destacamos que cuando se ha analizado y utilizado Scratch, se llega a la conclusión de que las aplicaciones que puedan desarrollarse dentro del campo de la educación están relacionadas con el juego, con "lo lúdico" y, la verdad, no hay mejor manera de ir aprendiendo que esa.

Así, esta herramienta permitirá a los niños elaborar desde simples juegos interactivos a complejas producciones artísticas, pasando por la creación de animaciones con música y llegando, incluso, a elaborar simulaciones.

De esta forma, puede deducirse que vale más que la pena empezar a utilizar esta aplicación en el ámbito de la educación, sin ningún tipo de excusa, ya que su gratuidad, facilidad de uso y posibilidades le convierten en algo más que un aliado en el mundo de la enseñanza.

Otra de dichas herramientas, enfocada a smartphones y tablets, es AppInventor[9] donde uno de sus principales destinatarios son los educadores, que pueden introducir esta herramienta para que los estudiantes den sus primeros

[9] http://appinventor.mit.edu/

pasos en el mundo de la programación. También para crear contenidos de apoyo para sus propias clases, lecciones y asignaturas.

Las apps que se consiguen con AppInventor son muy sencillas, aunque se adaptan a las necesidades básicas del aula. De esta forma, un estudiante que nunca hayan creado una aplicación lo puede hacer en relativamente poco tiempo.

El diseño de estas apps que los alumnos pueden instalar en sus dispositivos móviles representa una evolución natural en la secuencia progresiva de aprendizaje en interacción con dichos dispositivos y en sintonía con un aprendizaje orientado a la producción digital en distintos formatos.

Esta herramienta podría ser de gran interés para la enseñanza debido al gran número de usuario que disponen de dispositivos móviles, estos datos los podemos comprobar en la publicación "Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación en los Hogares. Año 2015" del Instituto Nacional de Estadística el equipamiento de las viviendas en algunos productos de tecnologías de información y comunicación indica que 96% de las personas usan teléfonos móviles.

Por último, hablaremos de Etoys[10] esta herramienta ayuda a comprender ideas, construyendo y jugando. En este aprendizaje computarizado los estudiantes encontrarán una herramienta desafiante que permite diseñar diversos proyectos de programación; al mismo tiempo el alumno, al realizar estas acciones, construye en su conocimiento integrando diversas áreas y materias. Además, el estudiante, al usar Etoys tiene un laboratorio virtual que pone retos a su creatividad.

Además de estas herramientas mencionadas existen organizaciones como Code.org que su principal objetivo es difundir la programación como parte de la educación básica de los jóvenes, para ello dispone de una amplia sección educativa en la que se pueden probar algunas técnicas de aprendizaje de la programación y practicar directamente desde el propio navegador web, también hay pistas sobre

[10] http://www.squeakland.org/

academias y cursos, software educativo y formas de aprender mediante juegos y sistemas tales como los robots de Lego o juegos para teléfonos móviles, asimismo se puede ver una amplísima selección de enlaces con tutoriales sobre lenguajes de programación obteniendo así que los jóvenes desarrollen su pensamiento computacional.

En el siguiente capítulo abordaremos el estudio de algunas de las herramientas mencionadas, además de otras de especial interés.

Capítulo 2. Herramientas

En este capítulo vamos a describir una serie de herramientas, como son ALICE, SCRATCH, SNAP, GREENFOOT, APPINVENTOR y AGENTSHEETS, que permiten hacer que la programación en edades pre-universitarias sea más sencilla de aprender.

2.1 ALICE.

Se trata de un entorno que tiene como característica principal la programación en 3D que nos permite crear una animación para relatar una historia, un juego interactivo o un video para compartir en la web. ALICE es una herramienta de enseñanza gratuita diseñada para ser el primer contacto con la programación orientada a objetos, además podemos aprender los conceptos fundamentales de la programación en el contexto de la creación de una película animada o un videojuego sencillo. Esta herramienta suministra una serie de objetos tridimensionales (muebles, vehículos, personas, etc.) que habitan un mundo virtual y permite crear con ellos un programa para animarlos.

La interfaz de usuario de ALICE permite arrastrar y soltar objetos (drag and drop) en el escenario para crear un programa donde las instrucciones corresponden a declaraciones estándar de un lenguaje orientado a objetos. El resultado se puede ver de forma inmediata, y de esta forma entenderemos la relación entre código y el comportamiento de los objetos.

2.2 SCRATCH.

En esta herramienta nos encontramos con un entorno de programación visual y multimedia basado en Squeak, de arrastrar y formar bloques al estilo lego, para crear un proyecto, en vez de la programación típica donde se debe escribir todo; y como LogoBlocks o PicoBlocks, los bloques son autoencajables, de modo que solo se ajustan si son sintácticamente correctos, esto permite concentrarse en la

lógica de programación sin perder tiempo analizando si está escrito correctamente.

A pesar de haber sido ideado como una herramienta sencilla para jóvenes, ya ha demostrado ser un instrumento valioso para la introducción a la programación tanto en educación secundaria como en universidades donde los jóvenes tienen oportunidades para aprender conceptos informáticos importantes tales como iteración, control de flujo, condicionales, variable, etc. comprobándose como estos alumnos migran fácilmente a los lenguajes tradicionales después de haberse introducido en la programación con esta herramienta.

En cuanto a los aspectos esenciales del diseño las podemos resumir en las siguientes características: la programación se basa en la metáfora de bloques de construcción, la manipulación de archivos multimedia a las que se le han añadido rutinas de manipulación de imágenes en forma de filtros y control de los mismos, además se permite compartir los proyectos a través de la plataforma web, también tiene como propósito programar objetos físicos del mismo modo que se programan objetos virtuales en pantalla usando entradas de sensores físicos para controlar el comportamiento de los objetos físicos y creaciones virtuales, y por último, la herramienta tiene soporte para múltiples lenguajes donde los bloques se pueden cambiar a diferentes idiomas.

2.3 SNAP!.

Anteriormente BYOB, Snap! es una reimplementación extendida de Scratch, por lo tanto sigue la idea de arrastrar y soltar objetos, permitiendo construir sus propios bloques. Las capacidades que se han añadido a esta herramienta, como por ejemplo las listas de primer nivel y procedimientos de primera clase, la hacen adecuada para la introducción a la programación para edades pre-universitarias.

Una de las características principales de esta herramienta es que se ejecuta en un navegador, además se implementa mediante el uso de JavaScript, el cual está diseñado para limitar la capacidad del software basado en navegador que afecten a su equipo, por lo que es seguro ejecutar proyectos de incluso de otras personas.

2.4 GREENFOOT.

Hablamos de una herramienta flexible con fines educativos y para el auto-aprendizaje como pueden ser las escuelas para aprender y enseñar los principios de la programación.

Greenfoot dispone de características más avanzadas que otras herramientas, como la edición de código fuente en Java diseñado específicamente para cubrir los conceptos y principios de la programación orientada a objetos, con el que se puede crear aplicaciones gráficas tales como simulaciones o juegos en 2D que nos permite introducir las primeras líneas de código Java y conocer los conceptos básicos de la programación orientada a objetos de una forma práctica y accesible para todos. Además los estudiantes pueden concentrarse en modificar la lógica de la aplicación, y combinar y experimentar con los objetos. Dicho entorno ha sido diseñado para atraer rápidamente a jóvenes que pueden no tener conocimientos previos o experiencia en programación donde se puede realizar animaciones simples, y es posible realizar escenarios con un aspecto profesional.

2.5 APPINVENTOR.

Se trata de una herramienta web de desarrollo para iniciarse en el mundo de la programación. Con dicha herramienta pueden crearse aplicaciones para el sistema operativo de dispositivos móviles Android, su filosofía es la de un editor que utiliza el método drag and drop para la generación de interfaces gráficas y un sistema de bloques para gestionar el comportamiento de la aplicación. Los proyectos generados a través de esta herramienta se almacenan automáticamente en los servidores de la herramienta, permitiendo llevar en todo momento un seguimiento y control de todo nuestro trabajo.

La principal característica de AppInventor es que no es necesario tener ningún conocimiento de programación para desarrollar aplicaciones, donde se ofrece un abanico amplio de posibilidades para los desarrolladores, como pueden ser los juegos que pueden ser sencillos, haciendo uso incluso del acelerómetro del dispositivo móvil o aplicaciones educativas, donde podemos desarrollar aplicaciones para los test de respuestas múltiples o preguntas directas.

2.6 AGENTSHEETS.

Esta herramienta "revolucionaria" tiene como principal característica permitir crear juegos y simulaciones basadas en agentes y publicarlas en la web a través de una interfaz fácil de usar donde sigue la filosofía drag and drop. Esta herramienta está desarrollada tanto para estudiantes de primaria, como para científicos o grandes proyectos universitarios.

Su utilización es recomendada ya que se pueden construir simulaciones complejas, comunicar ideas a los demás, o simplemente crear juegos:

- En las ciencias de la computación, por ejemplo, se pueden crear simulaciones interactivas para ayudar a comprender nuevas ideas, explorar procesos complejos en diversos campos de las ciencias.
- En los juegos, la construcción de los mismos (no solo para jugar con ellos) sino para enseñar conceptos de informática, la lógica y el pensamiento algorítmico. La escalabilidad del enfoque del diseño de juegos es ideal para equilibrar las preocupaciones motivacionales y educativas de la enseñanza de la informática.

Esta herramienta se caracteriza por la simplicidad para comenzar, al igual que las anteriores se basa en el principio de arrastrar y soltar objetos, esto permite la creación de programas sencillos. Además, gracias al uso de un lenguaje de fórmulas, visualizaciones científicas, y multimedia, los usuarios jóvenes como pueden ser los estudiantes de secundaria pueden crear sofisticadas aplicaciones y juegos que incluyan complejos algoritmos de Inteligencia Artificial.

Capítulo 3.
Caso de estudio

Cuando nos planteamos enseñar programación a un niño es necesario comentar que no vamos a enseñarles las herramientas profesionales ya que son niños, y por lo tanto sus conocimientos y capacidades en muchos ámbitos son mucho más limitados.

Es por ello por lo que nos ayudaremos de herramientas específicamente creadas para jóvenes, incluyendo niños desde los 5 o 7 años, y hasta adolescentes.

Hay organizaciones como code.org que nos enseñan los conocimientos básicos sobre programación, pero están algo limitados y al final son algo lineales, sin dejar lado a la imaginación y la creatividad. Para ello es necesario ir un paso más allá.

Es por ello que vamos a ver como una serie de herramientas como ALICE, SCRACHT y GREENFOOT que siguen el mismo principio de "drag and drop" (arrastrar y soltar), pueden ofrecer mucho más, ya que aparte de tener un entorno similar a las herramientas mencionadas anteriormente aportan muchas más funcionalidades y posibilidades a disposición del usuario.

En este capítulo vamos a ver un ejemplo sencillo sobre el método de ordenación burbuja, donde primero hablaremos un poco de cómo funciona el código y a continuación de su implementación en el software ALICE, SCRACHT y GREENFOOT.

Se han seleccionado estas herramientas porque abarcan grupos de edades diferentes y gracias e ello se va a poder comprobar, cuando se implemente el ejemplo propuesto, que el nivel de dificultad de cada una de las herramientas va aumentando dependiendo del intervalo de edades al que pertenezca.

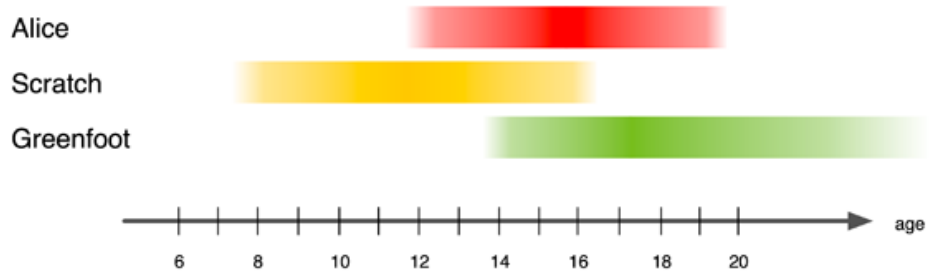

Figura 3.1. Intervalos de edades para las que están comprendidas cada una de las herramientas.

3.1 Método de ordenación burbuja (Bubble Sort)

Se trata de uno de los algoritmos más antiguos y el más sencillos e intuitivos.

Se basa en ir sistemáticamente comprando cada elemento de una lista que va a ser ordenada con el siguiente, intercambiando de posición si se encuentran en orden equivocado. Es necesario ir revisando varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista esta ordenada.

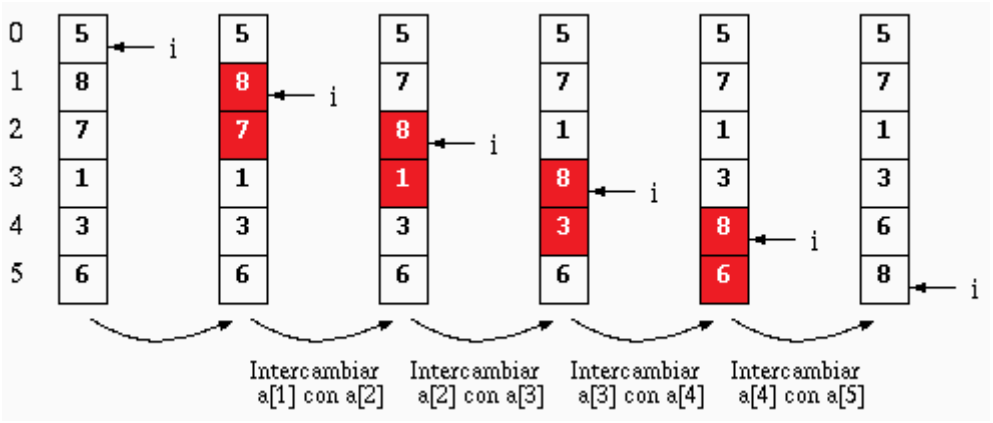

Figura 3.2. Ejemplo de primera pasada del algoritmo de ordenación burbuja.

3.2 Implementación en ALICE

Para implementar el algoritmo de ordenación burbuja en ALICE lo primero que haremos, una vez iniciada la herramienta, es elegir una plantilla para la realización de la demostración. A continuación se nos creara un entorno de desarrollo que constara, entre otras cosas, de una ventana donde se dispone la escena (aquí se añadirán los objetos que más adelante se programaran), otra donde se edita el programa y otra donde aparecen dos pestañas (una para los procedimientos y otra para la funciones).

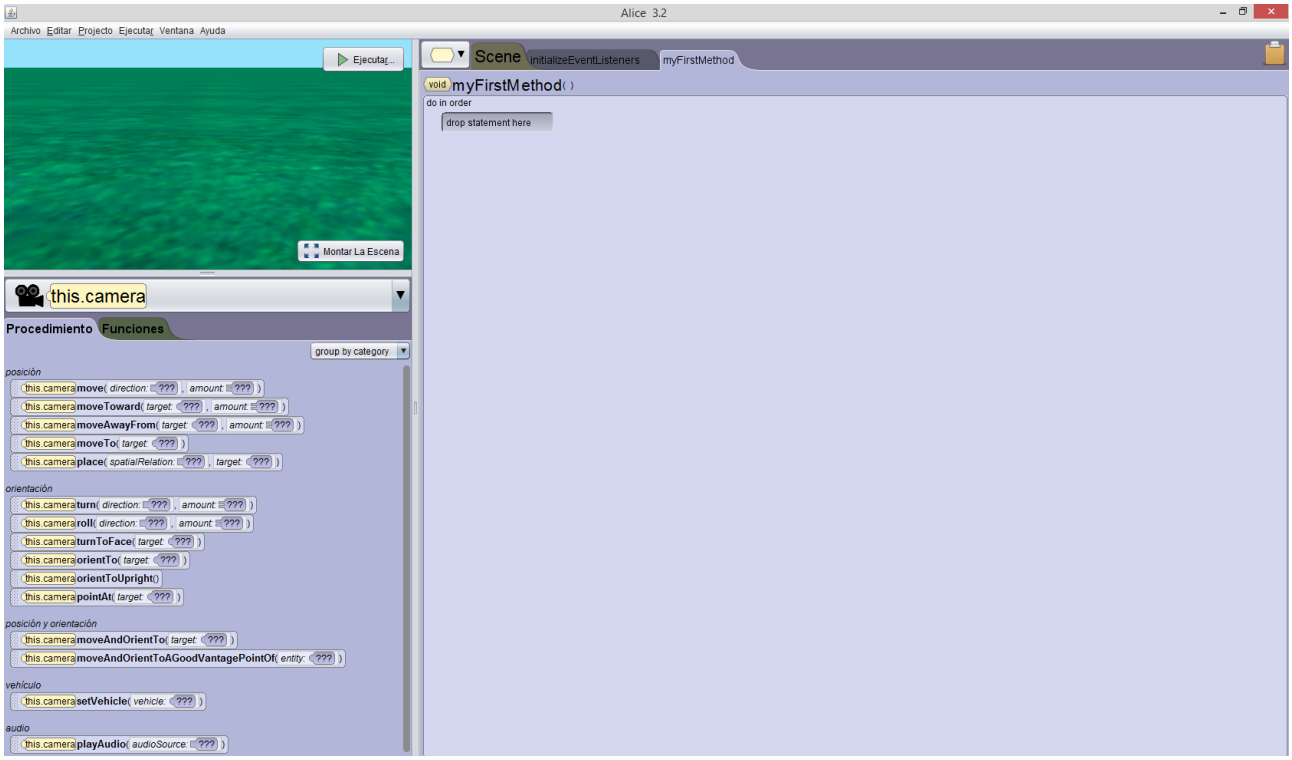

Figura 3.3. Entorno de programación de ALICE.

Lo siguiente que haremos es crear la escena, para ello clicaremos en la opción de "Montar la escena", la cual nos permitirá acceder al escenario donde podemos añadir los objetos para montar la escena.

Figura 3.4. Ventana de disposición de la escena.

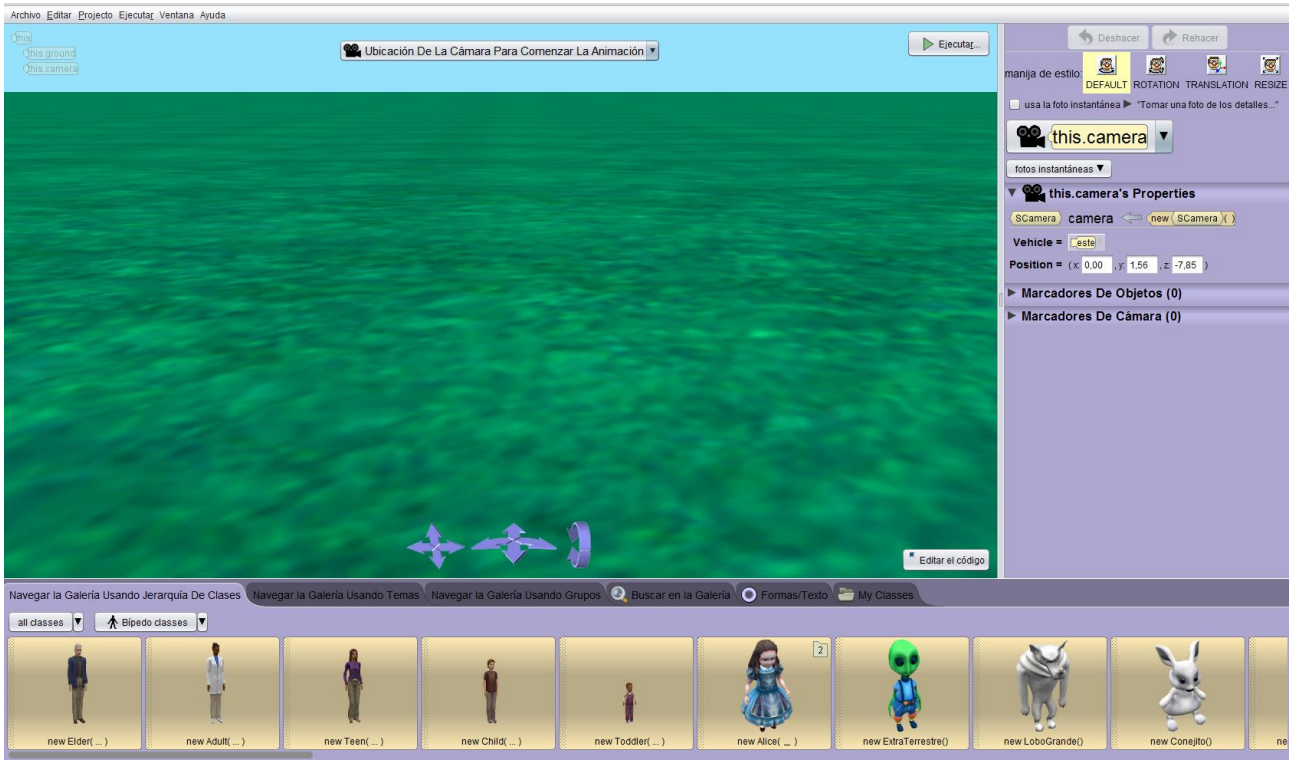

Figura 3.5. Escenario donde montar la escena.

Cuando añadimos un objeto lo primero que podremos hacer es modificarlos, como por ejemplo el color de pelo, el tipo de vestido/traje o el color de la piel, ya que nos aparecerá una ventana que nos lo permitirá. A continuación nos aparecerá otra ventana donde nos dará información del tipo de objeto que estamos utilizando y como lo vamos a inicializar, además de permitirnos cambiar el nombre del objeto.

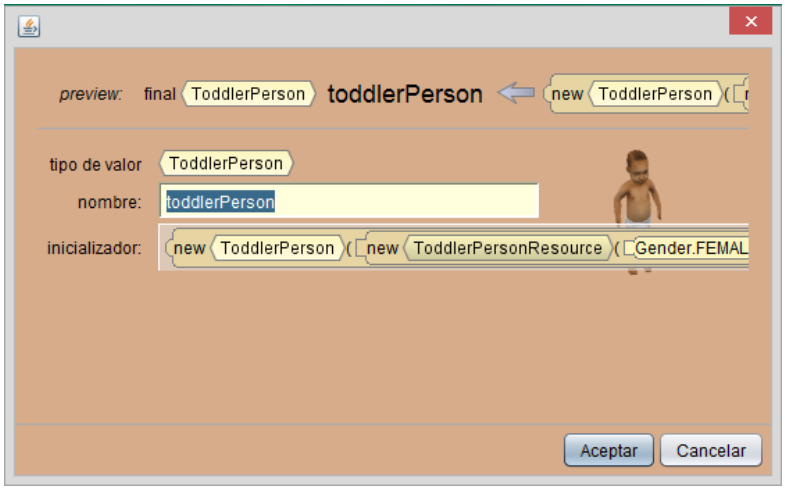

Figura 3.6. Ventana del objeto.

En este caso montaremos un escenario con cuatro objetos, tres de ellos los utilizaremos para mostrar cómo funciona el algoritmo, y el cuarto lo usaremos para que vaya narrando los pasos que se van donde en la ejecución. Además se han puesto tres círculos (uno por cada uno de los personajes que se emplearan en la demostración) para simular la posición de cada objeto y un cuarto que usaremos como si fuera una variable auxiliar para realizar el intercambio de los objetos.

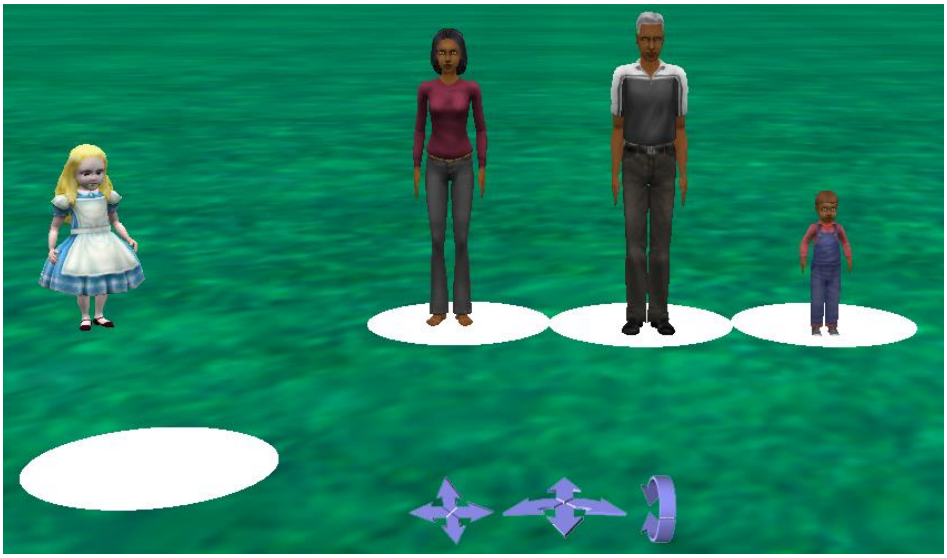

Figura 3.7. Escenario montado con los objetos.

Cuando tengamos los objetos en disposición llegará el momento de editar el código para que podamos ver en

funcionamiento este método de ordenación, hay que comentar que este ejemplo constara de una implementación suficientemente básica para que cualquier alumno de educación pre-universitaria pueda comprenderlo.

Lo primero será salir del montaje de la escena, para ello se cliquea en la opción de editar código.

Figura 3.8. Botón para pasar al modo de editar código.

Una vez cambiado de entorno, nos fijamos en la ventana de los procedimientos, ahí una vez que seleccionemos un objeto (lo podemos seleccionar de la ventana de disposición de escena) nos parecerá una serie de opciones las cual emplearemos para programar.

Para comenzar seleccionamos nuestro objeto que va a narrar los pasos del algoritmo y le vamos a hacer que diga una pequeña introducción, para ello escogemos en el panel de procedimientos el bloque que apunta al objeto y a continuación nos da la opción de decir (say, en inglés).

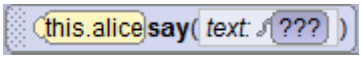

Figura 3.9. Bloque de código para dialogo.

Una vez seleccionado, lo arrastramos hasta el entorno de programación y lo soltamos, a continuación nos aparecerá una ventana que nos dará la opción de escribir una cadena de caracteres. Si queremos podemos añadirle más detalles al bloque, para ello bastara con seleccionar la opción que aparece al final de cada bloque "Añadir detalle".

Cuando hayamos hecho la introducción comenzaremos con la ordenación, vamos a ordenar los objetos por altura, así que el objeto que narra preguntara a cada uno de los otros objetos cuanto mide, utilizando el mismo bloque de código que en la introducción. Una vez le hayan respondido,

comenzara comparando cada objeto con el que tenga a su derecha, cambiando los círculos que tienen como base a un color rojo, este cambio de color se realiza con el bloque que apunta al círculo que queremos cambiar de color y la opción "setpaint" que nos permitirá realizar el cambio.

Si los dos objetos que están con bases rojas cumplen con la condición que el de la derecha es mayor al de la izquierda se realiza un intercambio sino, volveremos a poner las bases del color que estaban al principio volviendo a añadir otro bloque "setpaint" especificando dicho color.

Figura 3.10. Bloque de código para cambiar de color.

El paso anterior lo repetiremos hasta llegar al último objeto.

Para realizar los intercambios, cuando el objeto de la izquierda el mayor que el de la derecha, utilizaremos el bloque "moveTo" especificando la base a la que queremos que se mueva.

Figura 3.11. Bloque de código para mover objetos.

Por tanto, primero moveremos el objeto que se encuentre a la izquierda a la base que utilizaremos como auxiliar, a continuación el objeto que quedaba a la derecha lo movemos a la base que ocupaba el objeto que estaba a la izquierda y por último, el objeto que se encuentra en la base auxiliar lo movemos donde estaba el objeto que estaba al principio a la derecha.

Estos pasos (de comparación e intercambio) hay que repetirlos, como dice el algoritmo, tantas veces como el número de objetos menos uno, quedando al finalizar una secuencia ordenada de menor a mayor altura, empleando así el método de ordenación burbuja.

3.3 Implementación en SCRATCH

Para implementar el mismo ejemplo que en el anterior caso, pero esta vez con Scratch lo que haremos para comenzar es crear un nuevo entorno. Una creado procederemos a familiarizarnos con el entorno de trabajo, Scratch contiene entre otras cosas un escenario donde se pueden ver las historias que se van creando, el listado de objetos donde se muestran todos los objetos disponibles para un proyecto, la paleta de bloques y área de programas aquí es donde podemos programar los diferentes objetos arrastrando y soltando bloques desde la paleta de bloques al área de programas.

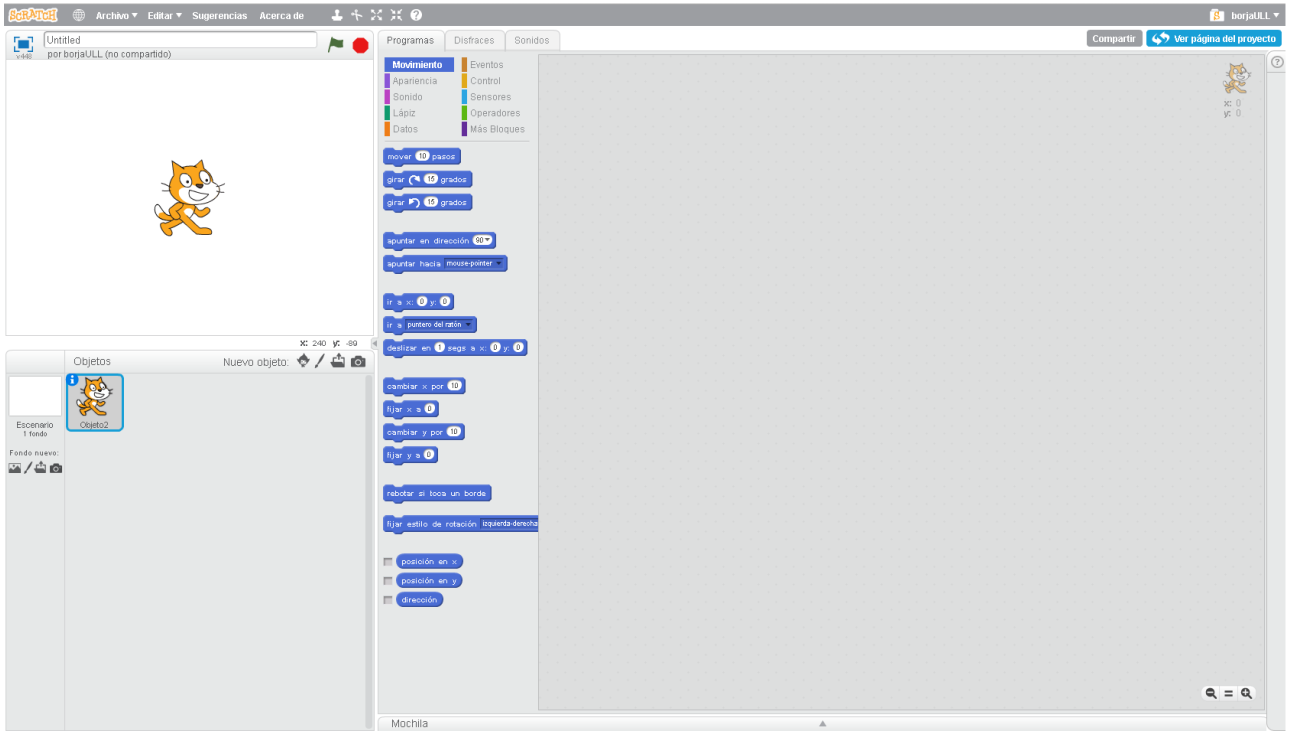

Figura 3.12. Entorno de trabajo de Scratch.

Una vez nos hemos adaptado al entorno toca programar el ejemplo propuesto. Lo primero que haremos es importar nuestros personajes de ALICE para tenerlos también en este ejemplo, para ello iremos donde se encuentra la lista de los objetos, y en la parte superior, en las opciones de "Nuevo objeto", seleccionamos la opción de "Cargar objeto desde archivo"

Figura 3.13. Opción para cargar un nuevo objeto.

Cuando tengamos los objetos cargados (deben aparecer en la lista de objetos de la herramienta) procedemos a indicar las instrucciones necesarias para que el algoritmo funcione, pero antes vamos a cambiar el fondo del escenario, para realizar este cambio hay que ir otra vez a la ventana de lista de objetos y a la derecha nos aparecen "Fondo nuevo" y en sus opciones elegiremos una que ya está en la biblioteca de scratch, para ello hay que seleccionar la opción "Seleccionar un fondo de la biblioteca" y elegir la que queramos.

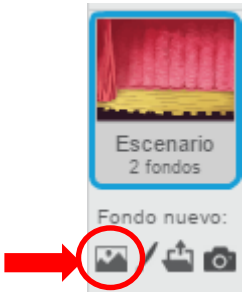

Figura 3.14. Opción para elegir un fondo de la biblioteca de scratch.

Figura 3.15. Escenario montado con los objetos y el fondo cambiado.

Cuando se cambie el fondo, tocara posicionar los objetos que deseemos ordenar dentro del escenario, en este ejemplo vamos a posicionarlos dentro de unos círculos que simularan las posiciones de un array y colocaremos un circulo de distinto color que simulara una variable auxiliar para realizar los intercambios necesarios. Estos círculos los podremos crear utilizando la opción "Dibujar nuevo objeto" de las opciones de "Nuevo objeto".

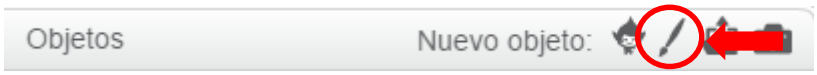

Figura 3.16. Opción para crear un nuevo objeto.

Una vez cambiado el fondo y posicionados los objetos dentro del escenario, seleccionamos, de la lista de objetos, el objeto donde aparece la mascota de la herramienta (el gato), el cual utilizaremos como objeto principal para comenzar a implementar nuestro algoritmo de ordenación. Lo que haremos una vez seleccionado es ir a la paleta de bloques y elegir la opción de "Eventos" y arrastrar el bloque de "al presionar" hasta el área del programa, con este bloque conseguiremos que el programa se inicie.

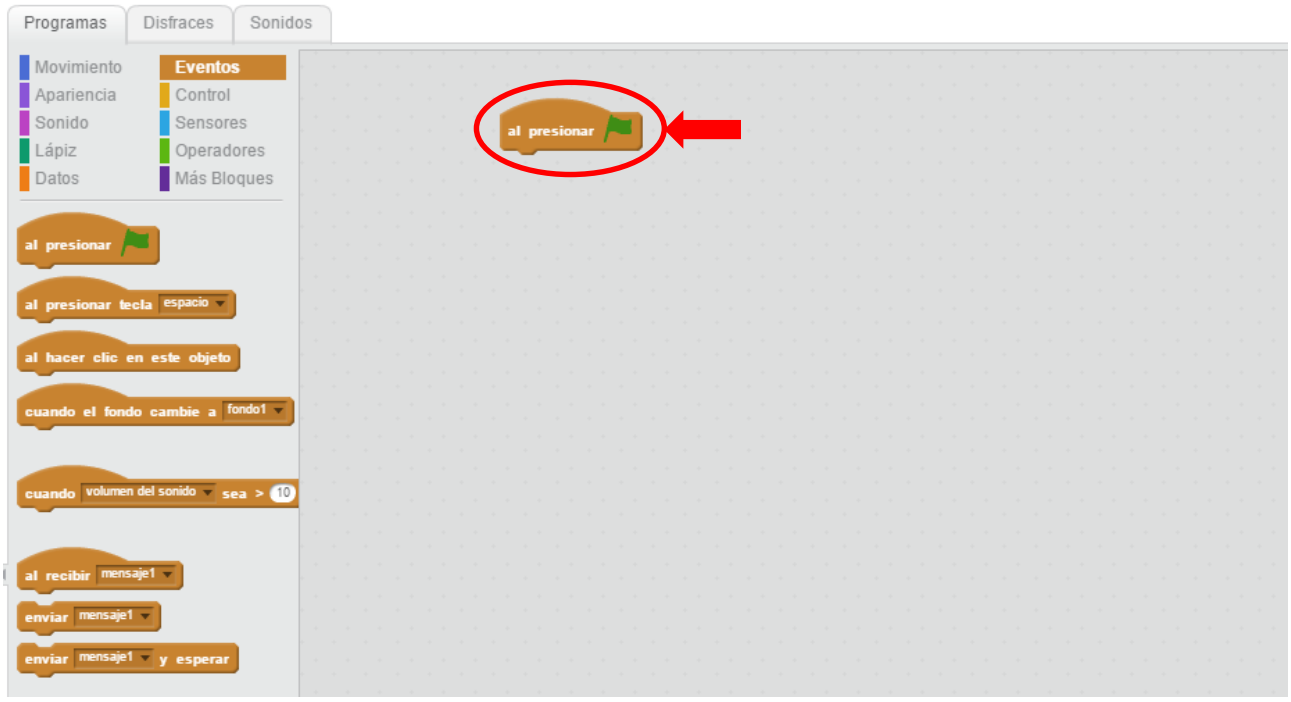

Figura 3.17. Bloque para la iniciación de un programa.

Lo siguiente que hay que hacer es arrastrar otro bloque de "Eventos", en este caso el de "enviar" donde le indicaremos un mensaje (en este caso "Res"), que al ejecutarse ira a cada uno de los objetos donde se encuentre el bloque de "recibir" con el mismo mensaje que el bloque "enviar" y modificara, en este caso, las posiciones finales de los objetos para que vuelvan a estar como al inicio.

Figura 3.18. Bloque creado para enviar un mensaje a los objetos para que vuelvan a su posición inicial.

Cuando tengamos el bloque de "recibir" en cada objeto que vamos a ordenar, hay que añadir después otros dos bloques, en este caso seleccionamos de la paleta de bloque las opción de "Movimiento" y escogemos los bloques de "fijar x a" y "fijar y a" donde los uniremos al bloque de "recibir" y fijaremos para cada objeto las posiciones iniciales que tienen.

Figura 3.19. Bloques para fijar las posiciones iniciales de cada objeto.

Una vez fijadas las posiciones volvemos al objeto principal (seleccionamos al gato) y continuamos, en este caso vamos a seleccionar de la paleta de bloque la opción de "Apariencia" y seleccionamos el bloque de "decir por segundos" y modificamos las opciones para que el personaje comente cómo funciona el algoritmo.

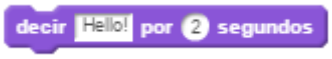

Figura 3.20. Bloque para que el personaje hable.

Una vez descrito el algoritmo por el personaje principal preguntaremos a cada personaje cuanto mide, ya que en este ejemplo vamos a ordenar por alturas (igual que el que se ha realizado en la herramienta ALICE). Por cada vez que le preguntemos a un personaje vamos a arrastrar un bloque al área del programa del objeto principal de "enviar y esperar" que encontraremos en la opción de "Eventos" de la paleta de bloques.

Figura 3.21. Bloque que usaremos para que el objeto principal espere mientras ejecuta una orden en otro objeto.

Una vez añadido este bloque iremos al objeto que queremos que nos diga la estatura y añadimos un bloque "recibir" donde, como ya mencionamos anteriormente, debe tener el mismo mensaje que el del bloque "enviar y esperar" y a continuación le añadimos otro bloque para que diga la estatura, en este caso sería el de "decir por segundos".

Figura 3.22. Secuencia de código del objeto que recibe un mensaje y devuelve la estatura.

Estos pasos lo repetiremos con cada objeto que vayamos a ordenar.

Una vez tengamos la medida de cada objeto el personaje principal va a ir comparando si el objeto que se encuentra a la izquierda es más grande que el que tiene a su derecha, si es afirmativa la premisa se utiliza un bloque "enviar y esperar" donde se indica el mensaje que ha de recibir los objetos que deberán realizar el intercambio.

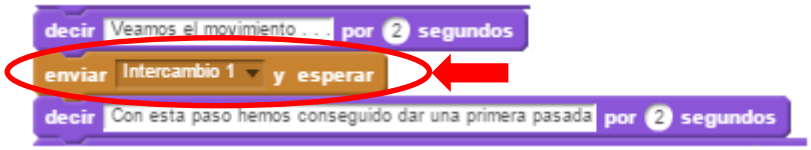

Figura 3.23. Bloque para enviar un mensaje a los objetos para que realicen el intercambio.

Para realizar un intercambio primero se ha de seleccionar el objeto de mayor tamaño (el que se encuentre a la izquierda del de menor tamaño) y moverlo a la posición auxiliar que creamos al principio, para ello se añade el bloque "recibir" con el que recibimos el mensaje del personaje principal y a continuación un nuevo bloque, en este caso el de "deslizar en segs a x y", que se encuentra en la opción de "Movimientos" de la paleta de bloques, para que el objeto se posicione en la variable auxiliar.

Figura 3.24. Bloque para posicionar el objeto.

Una vez realizado ese movimiento, dentro del objeto que posicionamos en la variable auxiliar hay que añadir un

bloque “enviar y esperar” con un mensaje que recibirá el otro objeto (el de menor tamaño) que ocupara la posición que del objeto más grande.

Figura 3.25. Conjunto de bloques para realizar un movimeinto.

A continuación seleccionamos el objeto más pequeño y le añadimos el bloque de “recibir” con el mensaje que enviamos del objeto más grande y volvemos a utilizar el bloque que nos permitió deslizar el objeto, pero esta vez para que este objeto ocupe la posición que ocupaba el mayor. Por último el objeto que se encuentra en la variable auxiliar lo deslizamos añadiendo el mismo bloque para que ocupe su posición, en este caso donde se encontraba el objeto más pequeño.

Estos pasos se van repitiendo hasta que los objetos queden ordenados de menor a mayor, utilizando así el algoritmo de ordenación por el método de la burbuja.

3.4 Implementación en GREENFOOT

Ahora vamos a ver el mismo ejemplo de ordenación burbuja pero implementado con una herramienta con un poco más de nivel (normalmente está recomendado para edades de entre 13 a 25 años).

Para comenzar, una vez tengamos instalada la herramienta y sea ejecutada es fijarnos que el entorno en el que vamos a desarrollar el ejemplo es un poco diferente a las otras dos herramientas que se han utilizado. Lo primero es fijarnos en el entorno, donde se diferencia tres áreas principales:

- El “mundo”, que es el área más grande que ocupa la mayor parte de la pantalla. Es donde el programa se ejecutará y verás lo que ocurre.

- El “diagrama de clases” es donde se nos muestra las clases que participan en este escenario
- Y los “controles de ejecución”, son los botones Accionar, Ejecutar, y Reiniciar y la barra inferior para ajustar la velocidad.

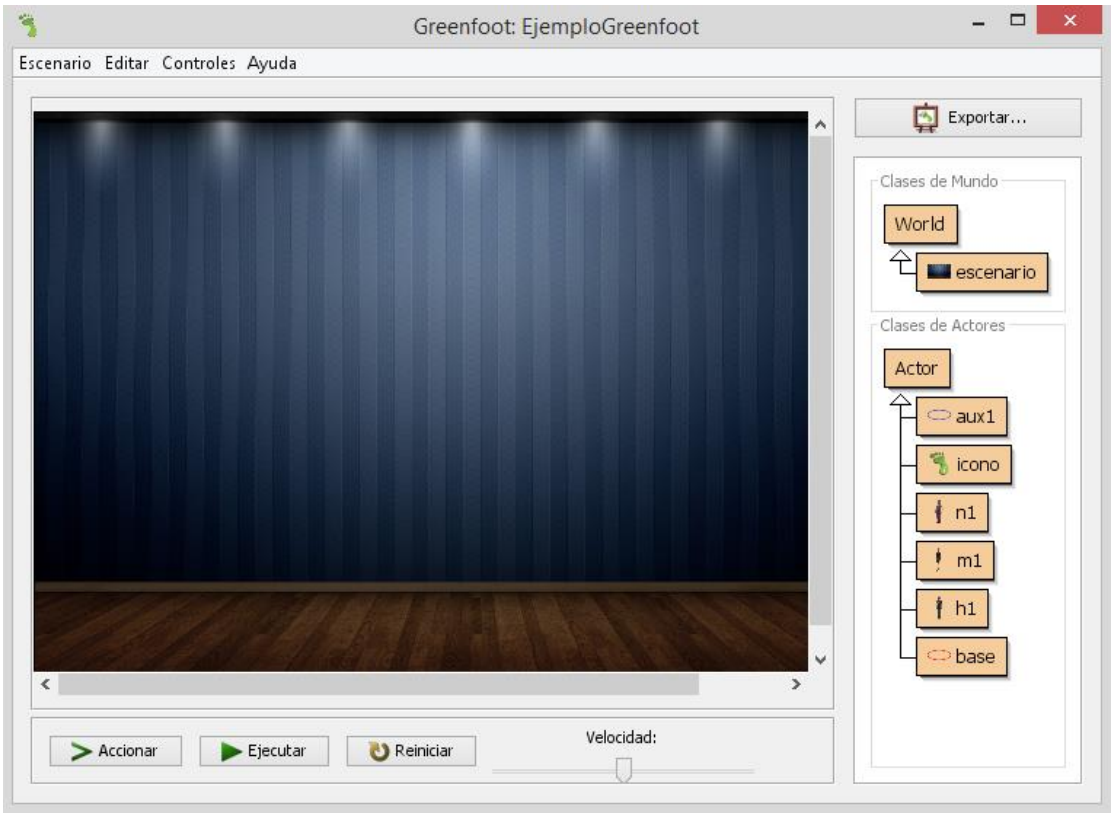

Figura 3.26. Entorno de la herramienta Greenfoot.

Una vez comprendido el entorno, se creara un nuevo escenario para poder implementar el ejemplo, para ello habrá que seleccionar la opción “Escenario” de la barra de superior y elegir “New Stride Scenario”, a continuación guardamos el proyecto con un nombre y se selecciona la ruta donde se quiera ubicar.

Concluido la creación del nuevo escenario vamos a desarrollar nuestro ejemplo, para ello se necesita crear un mundo. Esto se consigue seleccionando con el botón derecho del ratón la opción “world” del “diagrama de clases” y escogiendo “New subclass”.

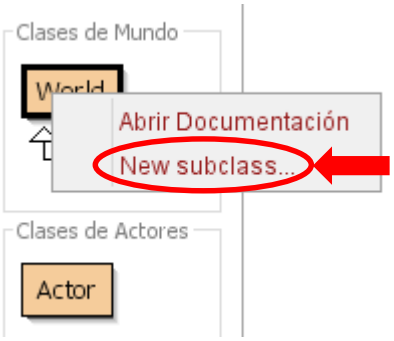

Figura 3.27. Opción para crear una nueva clase "mundo".

En el mundo, lo único que vamos a hacer es seleccionar una imagen para el fondo, la cual podemos seleccionar de la biblioteca de imágenes que nos ofrece la herramienta.

Lo siguiente que se hará es crear los objetos con los que interactuaremos para el ejemplo de ordenación. Como vamos a usar los personajes que utilizamos en las herramientas anteriores los importaremos desde donde los tenemos ubicados, el procedimiento parte de que hay que ir a la clase "Actor" del "diagrama de clases" y elegir la opción "New subclass", igual que hicimos en el paso anterior, y en la ventana que nos aparece a continuación iremos a la parte inferior donde veremos un botón, lo seleccionamos y elegimos la opción "Import from file", ello nos permitirá importar los personajes desde donde los tengamos almacenados.

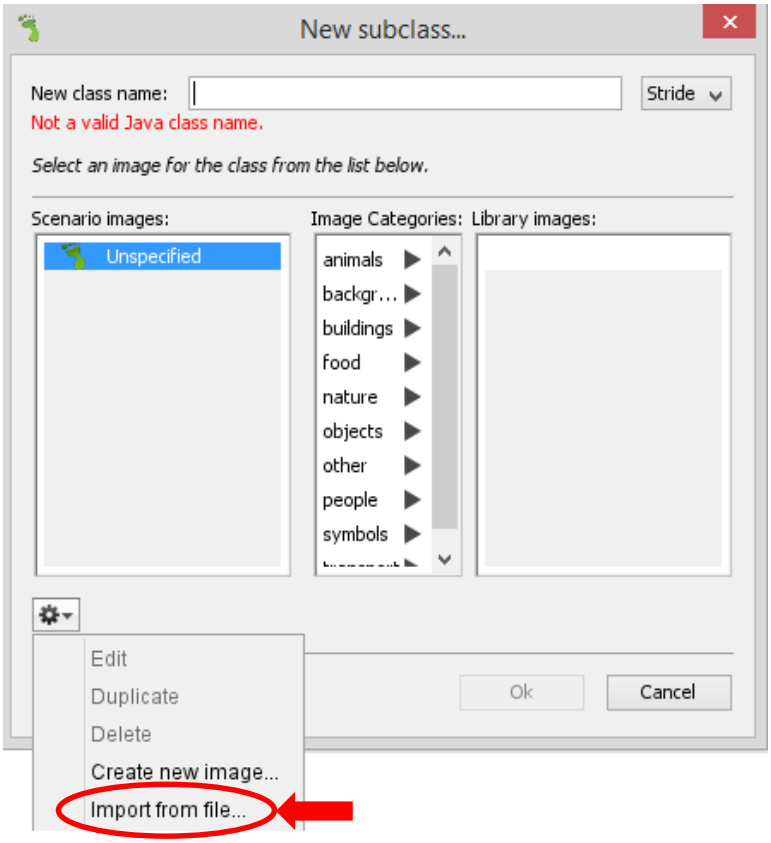

Figura 3.28. Opción para importar los personajes creados.

Es importante saber que cuando tengamos importados a los personajes, para poder añadirlos a la clase "Actor" hay que asignarles un nombre a la nueva subclase, que en este caso será cada uno de los personajes que hay que crear.

Llegado a este punto, ya tenemos el mundo y los objetos creados, ahora toca escribir el código para que se pueda realizar el método de ordenación que se ha elegido.

Figura 3.29. Escenario con los personajes.

Para poder añadir a los personajes al escenario hay que ir al "diagrama de clases" y seleccionar con el botón derecho del ratón la subclase que interese añadir, seleccionar "new" y a continuación ponerlo en el mundo que se ha creado.

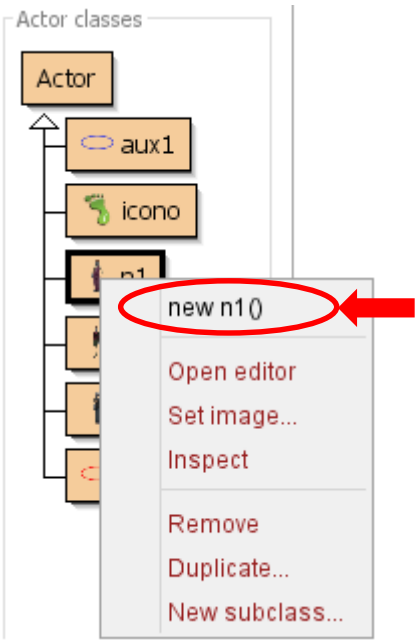

Figura 3.30. Opción para añadir un personaje al mundo.

Como comentario, antes de programar, hay que aclarar que en este ejemplo, se va a optar por una opción más dinámica, sencilla y donde se obtiene una mayor libertar para poder enseñar mejor el método de ordenación, para ello los personajes se moverán con el teclado.

Lo primero que vamos a programar son las subclases actores que se han importado, para ello la seleccionamos con el botón derecho del ratón y elegimos "open editor".

Figura 3.31. Opción para abrir el editor.

A continuación se abrirá un editor en el que se tendrá que escribir el código que se desee ejecutar en el lenguaje java. En este caso vamos a hacer que los personajes se muevan dependiendo del número que seleccionemos (1, 2, 3) para diferenciar al personaje que se quiera mover y sin soltar la tecla movemos al personaje con las flechas de dirección del teclado.

Los pasos descritos anteriormente se consiguen escribiendo en el método "public void act()" de cada subclase, que se genera por defecto cada vez que creamos un actor, un condicional en el que si presionamos la tecla uno, dos o tres del teclado seleccionamos a un actor y a continuación llamamos a otro método que pertenece a cada actor y que nos permite moverlo con las flechas de dirección.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class n1 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class n1 extends Actor
{
    /**
     * Act - do whatever the n1 wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if (Greenfoot.isKeyDown("1")){
            mover();  }
    }
    public void mover()
    {
        int y=getY();
        // Add your action code here.
        if (Greenfoot.isKeyDown("right"))
            move(1);
        if (Greenfoot.isKeyDown("left"))
            move(-1);
        if (Greenfoot.isKeyDown("up"))
            y--;
        if (Greenfoot.isKeyDown("down"))
            y++;
        setLocation(getX(), y);
    }
}
```

Figura 3.32. Código para seleccionar a cada actor y que se mueva.

Como se puede comprobar en el código de la figura anterior se observa que en el método "public void act()" escribimos el condicional y como condición escribimos "Greenfoot.isKeyDown()" esto permite detectar la tecla que esta presionada. Una vez se cumpla esta condición invoca a otro método, en este caso el de "mover()" en este punto vamos a generar una serie de condicionales en los cuales, dependiendo de la flecha que seleccionemos en el teclado el actor se moverá una posición.

Hay que tener en cuenta que los actores se mueven como en un plano cartesiano, de ahí que si se mueve a la derecha los números que se pasen como argumento sean positivos y los de la izquierda sean negativos.

En el caso de moverlos hacia arriba o hacia abajo necesitamos declarar una variable, en este caso "y", que permita devolver el valor devuelto es el índice vertical de la celda del actor en el mundo, esto se consigue con el método "getY()". A continuación, dependiendo si lo queremos mover hacia arriba incrementaremos la variable en una unidad (y++) o la decrementaremos (y--) si se quiere ir hacia abajo. Por último con la instrucción "setLocation(getX(),y)" se devuelvan las coordenadas al objeto, para que se pueda mover.

Este código se repetirá para cada subclase con la diferencia de que para cada uno hay que cambiar el número (1,2 o 3) con el que se seleccionara cada actor.

Una vez terminado con el código la idea es que se pueda ordenar a los personajes utilizando el método de ordenación burbuja, para ello hay que tener en cuenta que cada personaje está ubicado dentro de un círculo y que el conjunto de dichos círculos forman un array con posiciones que van desde el 0 al 2, además se ha añadido otro círculo distinto a los demás (en este ejemplo se ha puesto de color rojo) que actuara como variable auxiliar y se utilizara para realizar los intercambios.

Para poder ordenar los personajes tenemos que tener un criterio por el cual los vamos a ordenar, en este ejemplo será por el tamaño. Para ordenarlos empezaremos por el personaje que se encuentre más a la izquierda y nos fijaremos en el que tiene a su derecha, si este personaje es mayor no se hace nada pero si es menor se realiza un intercambio. Para realizar el intercambio se coloca al personaje de mayor tamaño (el que estaba a la izquierda) en la variable auxiliar, una vez realizado el movimiento se prosigue a mover al personaje más pequeño (el que estaba a la derecha) a la posición que ocupaba el otro personaje (el mayor) y por último se mueve al que estaba en la variable auxiliar a la posición que queda libre (la que ocupaba el personaje menor).

Este procedimiento de comparar a los personajes, intercambiándolos cuando sea necesario, se repetirá hasta que queden ordenados utilizando así el método de ordenación propuesto.

Capítulo 4. Resultados

A continuación se pretende exponer los resultados obtenidos del estudio que se ha realizado de las herramientas seleccionadas, donde se analizan características de visualización, tanto de métodos y atributos, de manipulación y creación de objetos ambos en tiempo real y código fuente, así como de la orientación del lenguaje de las herramientas, la capacidad de extender la complejidad de los programas básicos, o características básicas como, si las herramientas disponen de la opción de arrastrar y soltar objetos, el tipo de licencia que tienen o de si introducen los fundamentos de la programación orientada a objetos (POO).

	ALICE	SCRATCH	GREENFOOT
Visualización de métodos y atributos en tiempo real.	SI	SI	SI
Visualización de creación y manipulación de objetos en tiempo real.	SI	SI	SI
Visualización de código fuente de los objetos.	SI	SI	SI
Orientado al lenguaje de programación.	JAVA	Squeak / ActionScript (Scratch 2.0)	JAVA
Introduce los fundamentos de la POO (Objetos, clases, atributos, métodos).	SI	SI	SI
Opciones de "arrastrar y soltar" objetos.	SI	SI	SI
Capacidad de extender la complejidad de los programas básicos.	NO		SI
Documentación en línea	SI	SI	SI
Licencia de uso	BSD	Software libre, GPL v2	GNU/GPL

Tabla 4.1. Tabla de resultados obtenidos con cada herramienta.

Capítulo 5. Conclusiones

Se ha presentado el PROYECTO del Trabajo de Fin de Grado sobre herramientas para computación en estudios pre-universitarios.

Donde se ha realizado una Revisión Bibliográfica, Instalación de herramientas, Estudios de caso y fase de Experimentación.

Las herramientas que se han estudiado son ALICE, SCRATCH y GREENFOOT. Con las cuales se ha realizado, en el caso de estudio, el mismo ejemplo para poder experimentar y obtener así unos resultados que muestran ciertas características que poseen estas herramientas.

En cuanto a mi opinión personal me quedo con una frase que dice DAN CRAW en donde dice que

"Desconocer el lenguaje de los ordenadores será tan grave como lo es hoy no saber las letras y los números"

Con ello me hace entender lo importante que es programar, y más aún, el enseñarlo a edades en donde los alumnos vayan familiarizándose con los conocimientos básicos. Además, gracias a este proyecto he podido poner en prácticas los conocimientos que he adquirido a lo largo de la carrera como pueden ser la programación y los conocimientos de arquitectura de software.

Capítulo 6.
Summary and Conclusions

They presented the PROJECT to the End of Grade tools for computing studies pre-university.

Where we have conducted four follow-ups, and have completed the tasks of literature review, installation tools, case studies and experimentation phase in order to obtain results.

The tools that have been studied are ALICE, SCRATCH and GREENFOOT. With which it has been performed, in the case study, the same example to be able to experiment and obtain results that show certain characteristics that they possess these tools.

In my personal opinion I am left with a sentence that says DAN CRAW where he says that

"Not knowing the language of computers will be as serious as it is today does not know letters and numbers"

It makes me understand how important it is to schedule, and most importantly, teach it to ages in which the students are familiar with the basic knowledge. In addition, thanks to this project I was able to put into practice the knowledge that I have acquired along the career, such as programming and knowledge of software architecture.

Capítulo 7. Presupuesto

7.1 Presupuesto desglosado.

El presupuesto se ha calculado en base a las horas empleadas para la realización de cada una de las tareas propuestas (Revisión bibliográfica, instalación de herramientas, caso de estudio y elaboración de documentación), las cuales hemos desglosado en varias subtareas, por el precio de cada hora, obteniendo como resultado las siguientes tablas:

Tarea	Horas	Precio por hora (€)	Total (€)
Búsqueda de información	39	2	78
Redacción de la información	15	5	75

Tabla 7.1. Tabla de desglose de la revisión bibliográfica.

Tarea	Horas	Precio por hora (€)	Total (€)
Búsqueda de herramientas e información (manuales, guías de instalación)	6	2,5	15
Instalación y prueba de las herramientas	4	3,5	14

Tabla 7.2. Tabla de desglose de la instalación de herramientas.

Tarea	Horas	Precio por hora (€)	Total (€)
Búsqueda de información	7	2	14
Programación	9	10	90

Tabla 7.3. Tabla de desglose del caso de estudio.

A continuación se ha calculado los totales de horas realizadas por cada tarea y la suma de los totales, la cual mostraremos en la siguiente tabla:

Tarea	Horas	Total (€)
Revisión bibliográfica	54	153
Instalación de herramientas	5	29
Estudio de caso	16	104
Elaboración de la documentación	92	288

Tabla 7.4. Tabla de totales de horas y precio.

En la Elaboración de la documentación el precio por hora es de 3 euros

Así mismo se detalla en la siguiente tabla los gastos proporcionales de administración para la ejecución de este proyecto.

Concepto	Gasto mensual (€)	Gasto anual (€)	Horas dedicadas al proyecto	Total proporcional (€)
Alquiler	1.500	18.000	167	350,7
Acceso a internet + teléfono	66,12	793,44	167	16,7
Luz	440,5	5286	167	103,54
Material de oficina (Equipos informáticos)	250	3000	167	58,45

Tabla 7.5. Gastos de Administración.

Total de los gastos de administración es de 529,39 euros.

Total de las tareas realizadas es de 574 euros.

Total del presupuesto es **1.103,39 euros**.

7.2 Presupuesto desglosado para un autónomo.

En este apartado vamos a calcular el presupuesto, que al igual que en el apartado anterior se calculará en base a las horas empleadas para la realización de cada una de las tareas propuestas (Revisión bibliográfica, instalación de herramientas, caso de estudio y elaboración de documentación), las cuales hemos desglosado en varias subtareas, por el precio de cada hora, obteniendo como resultado las siguientes tablas:

Tarea	Horas	Precio por hora (€)	Total (€)
Búsqueda de información	39	2	78
Redacción de la información	15	5	75

Tabla 7.6. Tabla de desglose de la revisión bibliográfica.

Tarea	Horas	Precio por hora (€)	Total (€)
Búsqueda de herramientas e información (manuales, guías de instalación)	6	2,5	15
Instalación y prueba de las herramientas	4	3,5	14

Tabla 7.7. Tabla de desglose de la instalación de herramientas.

Tarea	Horas	Precio por hora (€)	Total (€)
Búsqueda de información	7	2	14
Programación	9	10	90

Tabla 7.8. Tabla de desglose del caso de estudio.

A continuación se ha calculado los totales de horas realizadas por cada tarea y la suma de los totales, la cual mostraremos en la siguiente tabla:

Tarea	Horas	Total (€)
Revisión bibliográfica	54	153
Instalación de herramientas	5	29
Estudio de caso	16	104
Elaboración de la documentación	92	288

Tabla 7.9. Tabla de totales de horas y precio.

En la Elaboración de la documentación el precio por hora es de 3 euros

Así mismo se detalla en la siguiente tabla los gastos proporcionales de administración para la ejecución de este proyecto.

Concepto	Gasto mensual (€)	Gasto anual (€)	Horas dedicadas al proyecto	Total proporcional (€)
Alquiler	1.500	18.000	167	350,7
Acceso a internet + teléfono	66,12	793,44	167	16,7
Luz	440,5	5286	167	103,54
Material de oficina (Equipos informáticos)	250	3000	167	58,45

Tabla 7.10. Gastos de Administración.

Como en este caso se trata de calcular el presupuesto para un autónomo hay que añadirle el Impuesto General Indirecto Canario (IGIC) que es el 7% del total.

Por tanto, el total bruto sería 1.103,39 euros.

Si calculamos el IGIC nos da 77,24 euros.

Quedando como total del presupuesto **1.180,63 euros.**

Apéndice A.
Ejemplo del Bubble Sort

A.1. Algoritmo en C++

```
/*****************************************************************************
*
* Bubble_Sort.cpp
*
*****************************************************************************
*
* DESCRIPCION: Algoritmo de ordenación burbuja escrito en c++
*
* void burbuja(int[ ] a, int n) {
*    int j= n-1;
*    boolean cambio= true;
*
*    while (cambio) {
*       cambio= false;
*       for (int i= 0; i<j; i++) {
*          if (a[i]>a[i+1]) {
*             // intercambiar
*             int aux= a[i];
*             a[i]= a[i+1];
*             a[i+1]= aux;
*             cambio= true;
*          }
*       }
*       j--;
*    }
* }
*****************************************************************************/
```

Bibliografía

[1] Hu, J., Bartneck, C., Salem, B. and Rauterberg, M. (2008) 'ALICE's adventures in cultural computing', Int. J. Arts and Technology, Vol. 1, No. 1, pp.102–118.

[2] Jane Nawrocki and Alka Harriger. 2009. How Alice game templates support student learning. In Proceedings of the 2009 Alice Symposium (ALICE '09). ACM, New York, NY, USA, Article 5 , 3 pages. DOI=http://dx.doi.org.accedys2.bbtk.ull.es/10.1145/1878513.1878518

[3] Randy Pausch, Matthew Conway, Robert DeLine, Rich Gossweiler, and Steve Miale. 1993. Alice and DIVER: a software architecture for building environments. In INTERACT '93 and CHI '93 Conference Companion on Human Factors in Computing Systems (CHI '93), Stacey Ashlund, Kevin Mullet, Austin Henderson, Erik Hollnagel, and Ted White (Eds.). ACM, New York, NY, USA, 13-14. DOI=http://dx.doi.org.accedys2.bbtk.ull.es/10.1145/259964.259976

[4] Joseph Shanahan and Daniela Marghitu. 2013. Software Engineering Java Curriculum with Alice and Cloud Computing. In Proceedings of Alice Symposium on Alice Symposium (ALICE '13). ACM, New York, NY, USA, , Article 4 , 6 pages. DOI=http://dx.doi.org.accedys2.bbtk.ull.es/10.1145/2532333.2532337

[5] Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, and Kevin Christiansen. 2000. Alice: lessons learned from building a 3D system for novices. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '00). ACM, New York, NY, USA, 486-493. DOI=http://dx.doi.org.accedys2.bbtk.ull.es/10.1145/332040.332481

[6] [M.D. María del Pilar Ramírez-Gil, M.D. Mariby Lucio-Castillo* y M.C. Juan José Garza-Saldaña/M.C. Lilia del Carmen García-Mundo y M.C. Juan Antonio Vargas-Enríquez - 29 de noviembre de 2011]

Título: "ALICE": Un entorno diferente para aprender programación orientada a objetos

Enlace: http://intranet.uat.edu.mx/cienciauat/ediciones/Edici%C3%B3n%20No.%2022,%20Septiembre-Noviembre%202011/Alice,%20un%20entorno%20diferente%20para%20aprender%20programaci%C3%B3n%20orientada%20a%20objetos.pdf

[7] [Duby Loscari Aldana Avilés - 10 de Julio de 2015]

Título: El Lenguaje de programación Scratch como material didáctico motivador para la aplicación y evaluación de contenidos en el área de inglés para alumnos con diagnóstico de TDAH.

Enlace: http://reunir.unir.net/bitstream/handle/123456789/3301/ALDANA%20%20AVILES%2c%20DUBY%20LOSCARI.pdf?sequence=1

[8] [Enrique Sánchez Acosta - Febrero de 2014]

Título: AppInventor: Programación para móviles al alcance de todos

Enlace: https://www.researchgate.net/profile/Enrique_Acosta2/publication/264501836_AppInventor_Programacin_para_mviles_al_alcance_de_todos/links/53e1ed930cf2235f352bf661.pdf

[9] [Gonzalo Zabala, Ricardo Morán, Sebastián Blanco - 2013]

Título: Una propuesta de enseñanza de programación en escuela media mediante el desarrollo de videojuegos con Etoys.

Enlace: http://42jaiio.sadio.org.ar/proceedings/simposios/Trabajos/SSI/19.pdf

[10] [Instituto nacional de estadística - 2015]

Título: Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación en los Hogares.

Enlace: http://www.ine.es/prensa/np933.pdf

[11] [Google, Fundación Española para la Ciencia y la Tecnología (FECYT) y everis - 2015]

Título: Educación en Ciencias de la Computación en España 2015.

Enlace:http://www.everis.com/spain/WCLibraryRepository/References/everis_InformeGoogle_210x297_RGB_7%C2%AApres.pdf

[12] [David Yan, Erin Taylor, and Alex Boldt Duke University Under the direction of Susan Rodger - 2015]

Título: An Introduction to Alice

Enlace:http://www.cs.duke.edu/csed/alice/aliceInSchools/workshop15/completeList.pdf

[13] [Lifelong Kindergarten del MIT Media Lab]

Título: Getting Started Guide Scratch

Enlace:https://cdn.scratch.mit.edu/scratchr2/static/__038df32711f6dad9a2b40bbd5714e01c__/pdfs/help/Getting-Started-Guide-Scratch2.pdf

[14] [Michael Kölling - 2011]

Título: Introducción a la programación con Greenfoot