MÓDULO DIDÁCTICO PROGRAMACIÓN JAVA UNDÉCIMO GRADO

agosto 2020

DEPARTAMENTO DE EDUCACIÓN
GOBIERNO DE PUERTO RICO

Página web: https://de.pr.gov/ **Twitter: @educacionpr**

CONTENIDO

LISTA DE COLABORADORES

Prof. Yesenia Pérez Irizarry
Maestra de Educación Comercial
Escuela Dr. Manuel de la Pila Iglesias
Ponce, Puerto Rico

Prof. Laura Álamo Serrano
Maestra de Educación Comercial
Escuela Ramón Power y Giralt
Las Piedras, Puerto Rico

Prof. Madeline Álvarez Ortiz
Maestra de Educación Comercial
Escuela Sec. Emilio R. Delgado
Corozal, Puerto Rico

Prof. Mercedes Alverio Rivera
Maestra de Educación Comercial
Escuela Rafael Cordero Molina
San Juan, Puerto Rico

Prof. Jean C. Cintrón Colón
Maestro de Educación Comercial
Escuela Dra. María Cadilla de Martínez
Arecibo, Puerto Rico

Prof. Janette González Pérez
Maestra de Educación Comercial
Escuela Jaime A. Collazo del Río
Morovis, Puerto Rico

Prof. José R. Jiménez Hernández
Maestro de Educación Comercial
Escuela Voc. Eladio Rivera Quiñones
Loíza, Puerto Rico

Prof. Mirelys Maceira Ruiz
Maestra de Educación Comercial
Escuela Fernando Callejo
Manatí, Puerto Rico

Prof. Luz Roldán Rohena
Maestra de Educación Comercial
Vocacional Ana Delia Flores Santana
Fajardo, Puerto Rico

CARTA PARA EL ESTUDIANTE, LAS FAMILIAS Y MAESTROS

Estimado estudiante:

Este módulo didáctico es un documento que favorece tu proceso de aprendizaje. Además, permite que aprendas en forma más efectiva e independiente, es decir, sin la necesidad de que dependas de la clase presencial o a distancia en todo momento. Del mismo modo, contiene todos los elementos necesarios para el aprendizaje de los conceptos claves y las destrezas de la clase de Programación Java, sin el apoyo constante de tu maestro. Su contenido ha sido elaborado por maestros, facilitadores docentes y directores de los programas académicos del Departamento de Educación de Puerto Rico (DEPR) para apoyar tu desarrollo académico e integral en estos tiempos extraordinarios en que vivimos.

Te invito a que inicies y completes este módulo didáctico siguiendo el calendario de progreso establecido por semana. En él, podrás repasar conocimientos, refinar habilidades y aprender cosas nuevas sobre la clase de Programación Java por medio de definiciones, ejemplos, lecturas, ejercicios de práctica y de evaluación. Además, te sugiere recursos disponibles en la internet, para que amplíes tu aprendizaje. Recuerda que esta experiencia de aprendizaje es fundamental en tu desarrollo académico y personal, así que comienza ya.

Estimadas familias:

El Departamento de Educación de Puerto Rico (DEPR) comprometido con la educación de nuestros estudiantes, ha diseñado este módulo didáctico con la colaboración de: maestros, facilitadores docentes y directores de los programas académicos. Su propósito es proveer el contenido académico de la materia de Programación Java para las primeras diez semanas del nuevo año escolar. Además, para desarrollar, reforzar y evaluar el dominio de conceptos y destrezas claves. Ésta es una de las alternativas que promueve el DEPR para desarrollar los conocimientos de nuestros estudiantes, tus hijos, para así mejorar el aprovechamiento académico de estos.

Está probado que cuando las familias se involucran en la educación de sus hijos mejora los resultados de su aprendizaje. Por esto, te invitamos a que apoyes el desarrollo académico e integral de tus hijos utilizando este módulo para apoyar su aprendizaje. Es fundamental que tu hijo avance en este módulo siguiendo el calendario de progreso establecido por semana.

El personal del DEPR reconoce que estarán realmente ansiosos ante las nuevas modalidades de enseñanza y que desean que sus hijos lo hagan muy bien. Le solicitamos a las familias que brinden una colaboración directa y activa en el proceso de enseñanza y aprendizaje de sus hijos. En estos tiempos extraordinarios en que vivimos, les recordamos que es importante que desarrolles la confianza, el sentido de logro y la independencia de tu hijo al realizar las tareas escolares. No olvides que las necesidades educativas de nuestros niños y jóvenes es responsabilidad de todos.

Estimados maestros:

El Departamento de Educación de Puerto Rico (DEPR) comprometido con la educación de nuestros estudiantes, ha diseñado este módulo didáctico con la colaboración de: maestros, facilitadores docentes y directores de los programas académicos. Este constituye un recurso útil y necesario para promover un proceso de enseñanza y aprendizaje innovador que permita favorecer el desarrollo holístico e integral de nuestros estudiantes al máximo de sus capacidades. Además, es una de las alternativas que se proveen para desarrollar los conocimientos claves en los estudiantes del DEPR; ante las situaciones de emergencia por fuerza mayor que enfrenta nuestro país.

El propósito del módulo es proveer el contenido de la materia de Programación Java para las primeras diez semanas del nuevo año escolar. Es una herramienta de trabajo que les ayudará a desarrollar conceptos y destrezas en los estudiantes para mejorar su aprovechamiento académico. Al seleccionar esta alternativa de enseñanza, deberás velar que los estudiantes avancen en el módulo siguiendo el calendario de progreso establecido por semana. Es importante promover el desarrollo pleno de estos, proveyéndole herramientas que puedan apoyar su aprendizaje. Por lo que, deben diversificar los ofrecimientos con alternativas creativas de aprendizaje y evaluación de tu propia creación para reducir de manera significativa las brechas en el aprovechamiento académico.

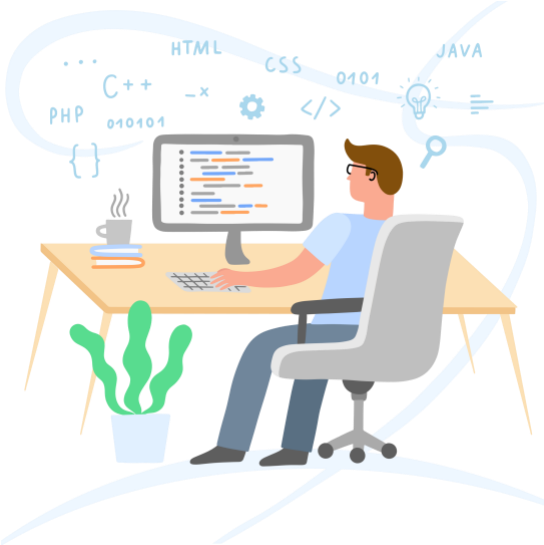
El personal del DEPR espera que este módulo les pueda ayudar a lograr que los estudiantes progresen significativamente en su aprovechamiento académico. Esperamos que esta iniciativa les pueda ayudar a desarrollar al máximo las capacidades de nuestros estudiantes.

CALENDARIO DE PROGRESO EN EL MÓDULO

DÍAS / SEMANAS	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
1	Inicio del curso	Unidad 1	Unidad 1	Unidad 1	Unidad 1
2	Unidad 1	Unidad 1	Unidad 1	Unidad 1	Unidad 1
3	Unidad 2	Unidad 2	Unidad 2	Unidad 2	Unidad 2
4	Unidad 3	Unidad 3	Unidad 3	Unidad 3	Unidad 3
5	Unidad 3	Unidad 3	Unidad 3	Unidad 3	Unidad 3
6	Unidad 4	Unidad 4	Unidad 4	Unidad 4	Unidad 4
7	Unidad 4	Unidad 4	Unidad 5	Unidad 5	Unidad 5
8	Unidad 5	Unidad 5	Unidad 5	Unidad 6	Unidad 6
9	Unidad 6	Unidad 6	Unidad 6	Unidad 7	Unidad 7
10	Unidad 7	Unidad 7	Unidad 7	Unidad 7	Unidad 7

INTRODUCCIÓN

Al utilizar teléfonos móviles o computadoras, muchas veces tenemos acceso a aplicaciones, programas, o *software*, para administrar nuestros dispositivos, y también, para completar tareas específicas. Los desarrolladores utilizan distintos lenguajes de programación para lograr que estas aplicaciones funcionen correctamente. Gracias a su versatilidad, *Java* es uno de los lenguajes de programación más populares entre los desarrolladores. Durante el año 2019, 88 por ciento de los teléfonos móviles vendidos se ejecutaba en Android, el sistema operativo móvil escrito en *Java*. Conocer más sobre *Java* puede abrirnos muchas puertas en una carrera relacionada a la informática (*Codecademy*, 2020).



Entornos de Desarrollo de Integración

Un **entorno de desarrollo integrado** (***IDE***, por sus siglas en inglés) es una herramienta que utilizan los programadores de computadoras para desarrollar aplicaciones de software. Un *IDE* incluye herramientas para escribir, editar, compilar, desplegar y depurar programas.

En el curso de un semestre presentaremos *Alice*, *Greenfoot* y *Eclipse*, que son entornos de desarrollo integrados (*IDE*), para crear aplicaciones *Java*. Las imágenes y la información en este Módulo son con fines educativos y tienen derecho de autor y representan a la empresa *Oracle* y a la organización *Alice*.org.

Alice es un entorno 3D para crear animaciones.

Greenfoot es un entorno 2D interactivo para crear juegos.

Eclipse es un entorno de desarrollo de aplicaciones *Java*.

Utilizando estos programas, aprenderemos sobre los elementos básicos de la programación informática y el lenguaje de programación *Java* para comenzar a escribir nuestros propios programas en *Java*.

¿Qué es *Java*?

Java es la base para prácticamente todos los tipos de aplicaciones de red, además del estándar global para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en web y software de empresa. Con más de 9 millones de desarrolladores en todo el mundo, *Java* nos permite desarrollar, implementar y utilizar de forma eficaz interesantes aplicaciones y servicios.

¿Por qué los desarrolladores de software eligen *Java*?

Java ha sido ajustado, ampliado y probado por toda una comunidad de desarrolladores y arquitectos de aplicaciones. Está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. Al poner a disposición de todo el mundo aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración, y reducir drásticamente el costo de propiedad tanto para aplicaciones de usuario como de empresa.

Java se posee un valor incalculable para los desarrolladores, ya que les permite:

- escribir software en una plataforma y ejecutarlo virtualmente en otra;
- crear programas que se puedan ejecutar en un explorador Web;
- desarrollar aplicaciones de servidor para foros en línea, encuestas, procesamiento de formularios HTML y mucho más;

- combinar y personalizar aplicaciones o servicios;
- codificar aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, microcontroladores, equipos inalámbricos, sensores, *gateways*, productos de consumo y prácticamente cualquier otro dispositivo electrónico. (Conozca más sobre la tecnología *Java*, s.f.)

Oracle Academy Student Hub (anteriormente conocido como *iLearning*) es un sistema de gestión de aprendizaje en el que podrá obtener planes de estudio, pruebas y exámenes. Si tenemos accesible una computadora o un teléfono móvil, el docente a cargo del curso puede generarnos un nombre de usuario y una contraseña para acceder a la programación del curso *Java Fundamentals* que nos ofrece *Oracle*. A través de esta plataforma, y con este texto como complemento, al final del curso podremos ser capaces de:

- crear animaciones y juegos,
- demostrar los conocimientos sobre la tecnología *Java* y el lenguaje de programación *Java*,
- utilizar el lenguaje de programación *Java* para crear aplicaciones, e
- integrar decisiones, bucles y otro código intermedio para crear aplicaciones.

UNIDAD 1: INTRODUCCIÓN A *ALICE*

Estándar: Programa de Aplicación

Objetivos: 1. Identifica, evalúa, selecciona, instala, utiliza, actualiza, soluciona problemas y personaliza las aplicaciones.

1.1 - Escena Inicial

Alice es un innovador entorno de programación en bloques que facilita la creación de animaciones, narraciones interactivas y la programación de juegos simples en 3D. Es una aplicación de codificación que motiva el aprendizaje a través de la exploración creativa. *Alice* es un programa diseñado para enseñarnos habilidades de pensamiento lógico y computacional, y principios fundamentales de programación. Además, nos sirve como una primera exposición a la programación orientada a objetos.

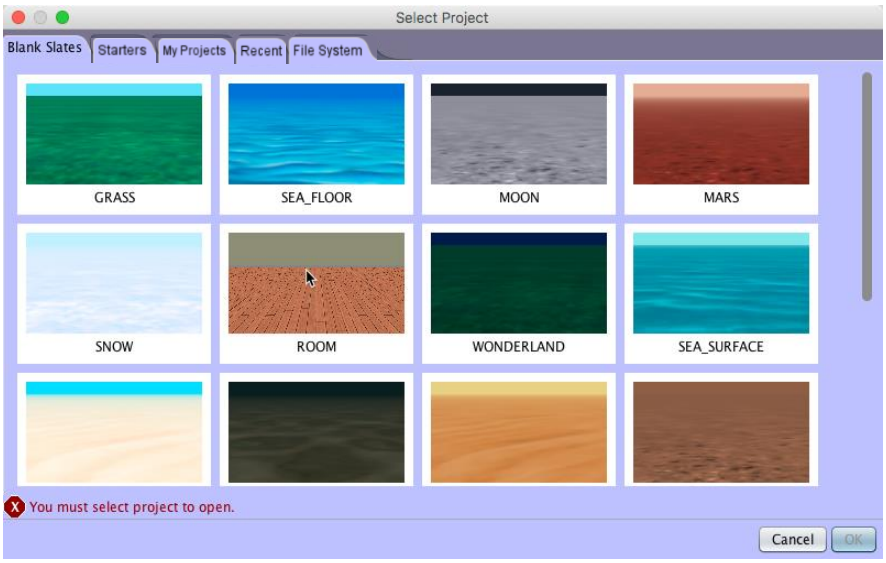
Al ejecutar el programa *Alice*, encontraremos en la pantalla la escena inicial. La escena inicial es el punto de partida de la animación. La escena es el primer paso donde podremos seleccionar una plantilla de fondo y podremos colocar objetos. La escena inicial consta de tres componentes:

- Una plantilla de fondo que proporciona el cielo, la tierra y la iluminación.
- Objetos de escenario estáticos que proporcionan el decorado.
- Objetos en movimiento que proporcionan la acción.

Pasos para crear un nuevo proyecto sin objetos en el escenario:

1. Acceda al programa *Alice*.

   Al acceder al programa, aparece una caja de diálogo para crear un nuevo proyecto o acceder a un proyecto ya creado. La caja de diálogo tiene cinco pestañas.



   - *Blank Slates*: plantillas de superficies o atmósferas sin objetos, es la pestaña que aparece por defecto al acceder al programa
   - *Starters*: plantillas de mundos (ya vienen con objetos incluidos en el escenario de acuerdo al tema)
   - *My Proyects*: muestra los proyectos existentes guardados en la carpeta de proyectos de *Alice*
   - *Recent*: muestra proyectos guardados recientemente en cualquier lugar de la computadora
   - *File System*: proporciona la alternativa para buscar un proyecto guardado en cualquier lugar de la computadora o dispositivo de almacenamiento

2. Selecciona la pestaña *Blank Slates* (si no está seleccionada).
3. Haga clic en la plantilla deseada.
4. Haga clic en el botón *Ok*.

Pasos para guardar un proyecto:

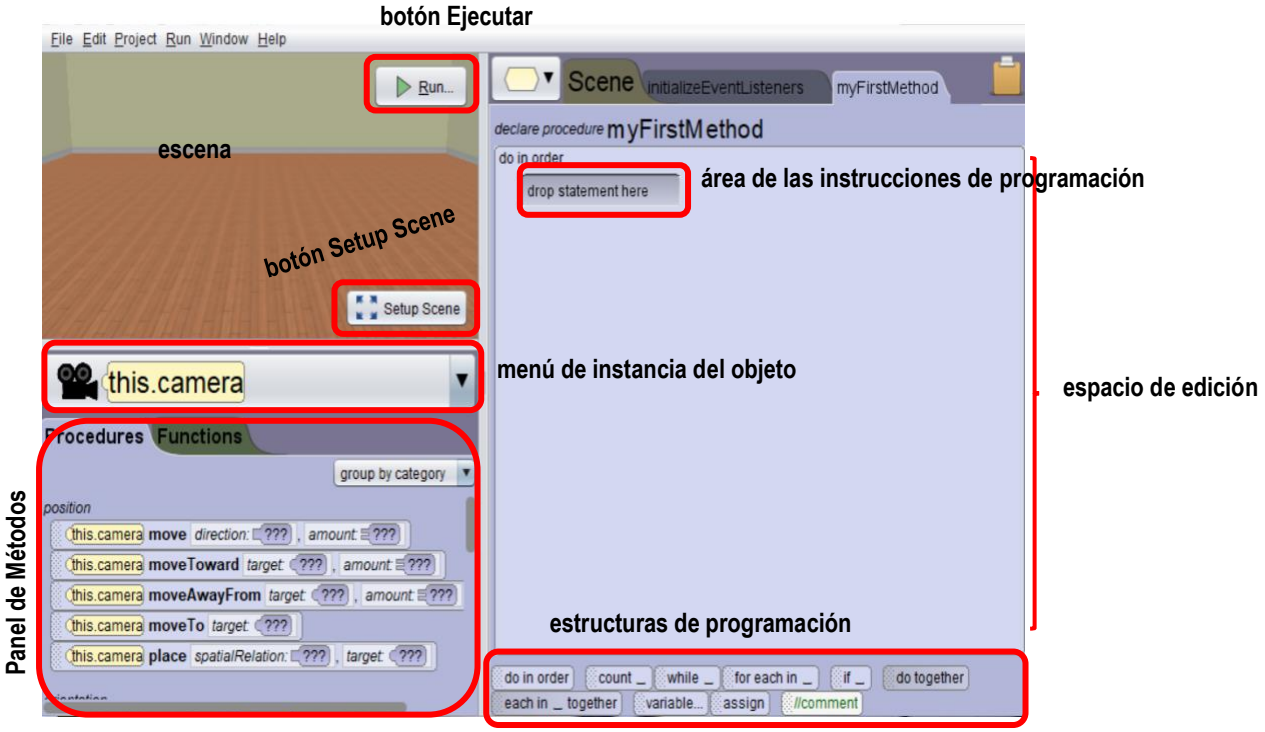
1. En la barra de menú, haga clic en *File*.
2. Haga clic en la opción *Save As*.
3. Busca y haga clic en la ubicación para guardar el proyecto.
4. Escriba el nombre del proyecto.
5. Haga clic en el botón *Save*.

Debemos guardar los proyectos con frecuencia para evitar perder el trabajo.

1.2 - Navegando entre los Editores

Alice brinda dos editores de entorno de trabajo diferentes, llamados **perspectivas**, que intercambiaremos con frecuencia mientras construyamos el proyecto. Los dos editores son:

- Editor de Código (*Edit Code*): es donde se agregan las instrucciones de programación.



- **botón *Run***: reproducir la animación
- **escena**: muestra las instancias de objetos agregados
- **botón *Setup Scene* (editor de escena)**: cambiar al editor de escena
- **menú de instancia del objeto**: aparecen todas las instancias del objeto que están en la escena, muestra la instancia del objeto seleccionada en la escena y podemos seleccionar la instancia del objeto
- **área de instrucción de programación**: donde se coloca la instrucción de programación
- **espacio de edición**: proporciona un entorno de arrastrar y soltar los procedimientos, funciones y sentencias de control
- **Panel de Métodos**: se encuentran las instrucciones de programación
- **estructuras de programación**: se encuentran las sentencias de control que permiten controlar el orden en que se ejecutan las instrucciones y crear bloques de programación

- Editor de Escena (*Setup Scene*): crear un mundo virtual agregando y organizando los objetos en una escena.

- **árbol de objetos (*object tree*)**: contiene la lista de todos los objetos que están en el escenario, se puede seleccionar la instancia del objeto
- **cámara**: actúa como un objeto en la animación, permite visualizar los escenarios y personajes desde distintos puntos de vista
- **botón *Run***: reproducir la animación
- **flechas de navegación**: mover el ángulo de la cámara sobre el escenario
- **escenario**: donde se agregan, colocan los objetos
- **Galería**: objetos tridimensionales para agregar, colocar en el escenario
- **botón *Edit Code* (editor de código)**: cambiar al editor de código
- **Panel de Propiedades**: atributos para aplicar a la instancia del objeto

El intercambio entre los dos editores es posible con el botón *Edit Code* o el botón *Setup Scene*. Nos encontramos en el **Editor de Código** cuando vemos las instrucciones de programación en el panel izquierdo de la ventana. Nos encontramos en el **Editor de Escena** cuando vemos la galería de objetos en la parte inferior de la pantalla. Por defecto, *Alice* se inicia en el Editor de Código.

1.3 - Agregar un Objeto a un Escenario

En términos de programación, una **clase** es un plano que se utiliza para crear un objeto, mientras que un **objeto** es una instancia de una clase. Después de agregar un objeto al escenario, se hace referencia a él como **instancia del objeto**. Podemos agregar varias instancias del mismo objeto a un escenario (por ejemplo, varios objetos de coral en el agua). Cada instancia del objeto debe tener un nombre único.

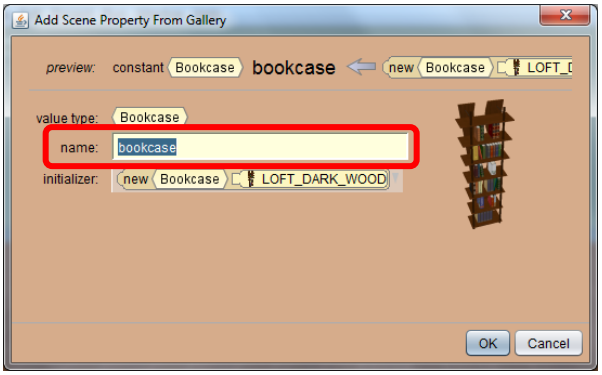
Pasos para agregar un objeto al escenario:

1. Haga clic en el botón *Setup Scene*.
2. En el Editor de Escena, podemos agregar un objeto (instancia de una clase) al escenario de una de estas dos formas:
   a. lugar en específico: haga clic en el objeto deseado y arrastra al escenario en el lugar específico que lo desea,

o

   b. centro del escenario: haga clic en el objeto deseado.

Cuando un objeto se agrega al escenario, se muestra una caja de diálogo con el nombre establecido por el programa; y el nombre del objeto se agrega automáticamente al árbol de objetos.



3. En la opción *name*, revisa el nombre del objeto que tiene por defecto. Modifica el nombre si deseas: en el espacio de *name*, escriba el nuevo nombre.
4. Haga clic en el botón *Ok*.

Siempre debemos verificar el nombre proporcionado para el objeto y así poder identificarlo al aplicar atributos a la instancia o declarar un procedimiento de programación.

1.4 - Editor de Escena

En el Editor de Escena de *Alice*, podemos:

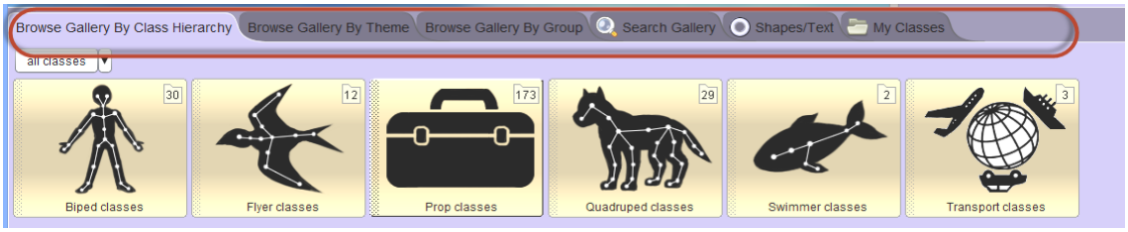
- Seleccionar objetos de la galería para agregarlos a la escena.
- Colocar los objetos en la escena con la paleta de manejadores.
- Editar las propiedades de un objeto en el panel *Properties*.
- Acceder al Editor de Código para agregar sentencias de programación.
- Ejecutar la animación después de agregar las sentencias de programación al espacio de edición.

El Editor de Escena contiene dos paneles: la Configuración de escenas (*Scene Setup*) en la parte superior y la Galería (*Gallery*) en la parte inferior.

Galería

La Galería es una recopilación de objetos tridimensionales que se pueden insertar en la escena y se organiza por pestañas. Para buscar objetos, examinamos las pestañas de la galería o utilizamos la función *Search Gallery* para buscar por palabra clave. Al seleccionar las diferentes clases, se muestran menús de navegación.

La galería tiene seis pestañas:



- *Browse Gallery by Class Hierarchy* - Organiza los objetos por movilidad.
- *Browse Gallery by Theme* - Organiza los objetos por región y contexto cultural.
- *Browse Gallery by Group* - Organiza los objetos por categorías.
- *Search Gallery* - Permite buscar un objeto por nombre.
- *Shapes/Text* - Organiza las formas de los objetos, el texto en 3D y la cartelera.
- *My Classes* - Permite agregar clases externas al proyecto.

1.5 - Selección de una Clase

Una **clase** contiene las instrucciones que definen la apariencia y el movimiento de un objeto. Todos los objetos de la misma clase tienen propiedades comunes. La clase proporciona instrucciones a *Alice* para crear y mostrar el objeto al agregarlo al escenario. En la pestaña *Class Hierarchy* agrupa los objetos por tipo de movilidad (bípedo, volador, etc.).

Las clases pueden contener subclases, como podemos observar en la siguiente imagen:

La clase *Alice* es una subclase de la clase de bípedos. Hay dos subclases de *Alice* en la galería: *Carnegie_Mellon* y *Wonderland*. Cada objeto *Alice* agregada al escenario hereda las propiedades que todos los objetos bípedos tienen en común: dos piernas, articulaciones móviles, etc.

1.6 - Guardar una Nueva Versión del Proyecto

Una vez colocados los objetos en la escena inicial, podemos ahorrar tiempo guardando varias versiones del proyecto y asignando un nombre diferente a cada versión. Una de las ventajas de guardar varias versiones de los proyectos es que podemos utilizar la misma escena para crear animaciones diferentes. Además, este paso nos permite ahorrar tiempo a la hora de crear la escena si se producen errores de programación. Asimismo, guardar los proyectos con frecuencia nos ayudará a recuperar nuestro trabajo en caso de un fallo en el suministro de alimentación o un fallo informático.

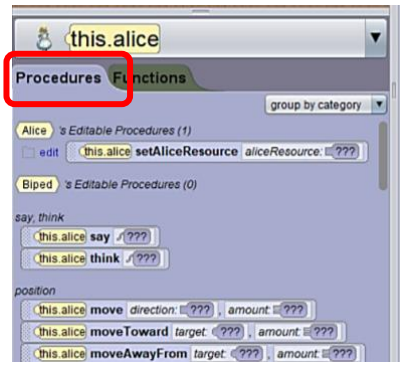
Pasos para guardar una nueva versión del proyecto:

1. Con el archivo en pantalla, en la barrar de menú, haga clic en *File*.
2. Haga clic en la opción *Save As*.
3. Busca y haga clic en la ubicación para guardar el proyecto.
4. Escriba el nuevo nombre del proyecto.
5. Haga clic en el botón *Save*.

Guarda los proyectos con frecuencia en caso de fallas eléctricas o de computadora.

1.7 -Editor de Código

En el Editor de Código es donde se agregan las instrucciones de programación para programar la animación. Hacemos clic en el botón *Edit Code* para mostrar el Editor de Código. En la pantalla del Editor de Código encontraremos el **Panel de Métodos** y en este panel está ubicada la pestaña ***Procedures* (Procedimientos)**. La pestaña Procedimientos muestra las instrucciones de programación creadas (editables) y las instrucciones de programación por defecto.

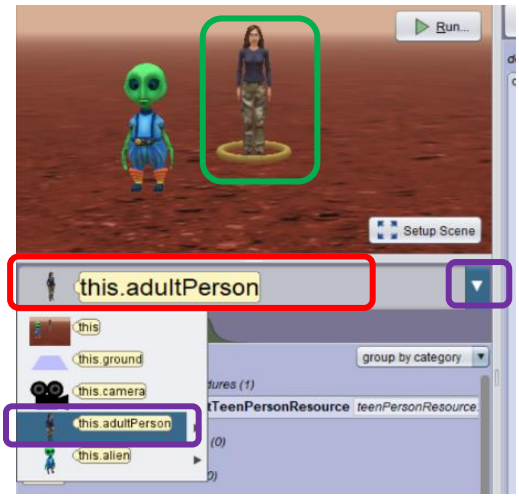


Un **procedimiento** es una parte del código de programación que define cómo el objeto debería ejecutar la tarea. Alice tiene un conjunto de procedimientos para cada clase; sin embargo, los usuarios pueden crear o "declarar" nuevos procedimientos

1.8 - Creación de un Procedimiento de Programación

Antes de declarar un procedimiento de programación, debemos asegurarnos ir al menú de instancia y seleccionar la instancia del objeto que deseamos animar. Hay dos maneras de seleccionar la instancia a la que desea crearle un procedimiento de programación:

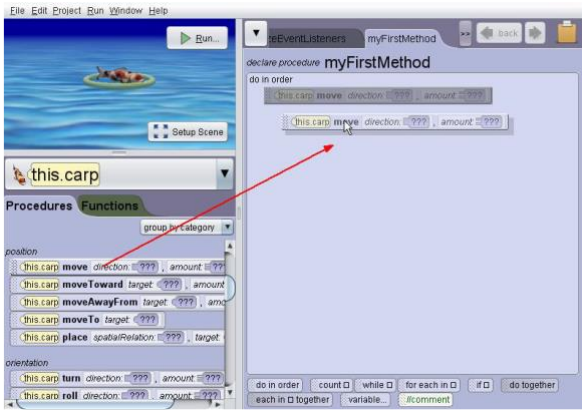
- Hacer clic en la instancia en la escena.
- Hacer clic en la flecha del menú de instancia y hacer clic en la instancia deseada.



Una vez hemos seleccionado la instancia, ésta se rodea con un anillo amarillo y se muestra en el menú de instancia.

Pasos para declarar un procedimiento de programación:

1. Haga clic en la instancia del objeto deseado.
2. En el Panel de Métodos, haga clic en la instrucción de programación deseada y arrastra al espacio de edición.



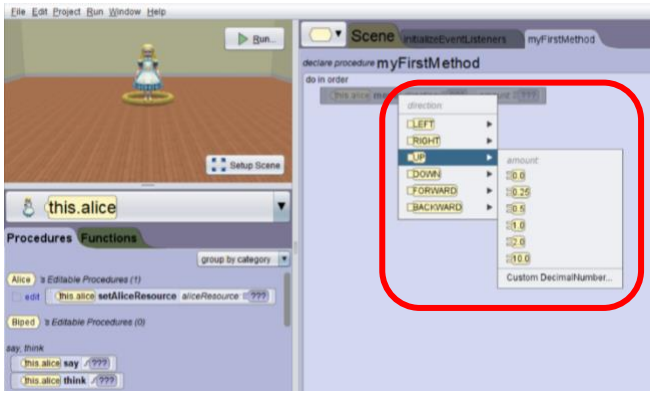
Después de arrastrar la instrucción de programación al espacio de edición, se muestra la lista desplegable para los valores para cada argumento.

3. Haga clic en cada argumento que desea dependiendo de la instrucción agregada.

Los tipos de argumentos pueden incluir: dirección, cantidad, duración y texto, etc. *Alice* reconoce cuántos argumentos son necesarios para cada instrucción de programación. Le presenta la cantidad correcta de lista desplegable para especificar los valores para cada uno de esos argumentos. Además, también se puede añadir otros argumentos al procedimiento de programación utilizado. Estos otros argumentos añaden otras características al procedimiento añadido.

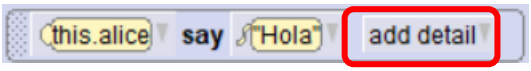
Pasos para cambiar el valor al argumento:

1. Haga clic en el botón de la flecha del argumento al que desea cambiar un valor.
2. Haga clic en el valor deseado. Si desea otro valor, haga clic en Custom… y escriba el valor deseado)



Pasos para agregar otro argumento en el procedimiento de programación:

1. Haga clic en el botón *add detail* en el procedimiento de instrucción al que desea aplicar otro argumento.



2. Haga clic al argumento deseado.

Dependiendo del procedimiento de instrucción, se mostrará los diferentes argumentos que se pueden aplicar para ese procedimiento.

3. Haga clic en el valor deseado. Si desea otro valor, haga clic en Custom… y escriba el valor deseado)

Copia de un procedimiento de programación

Para copiar un procedimiento de programación, podemos utilizar cualquiera de estos dos métodos:

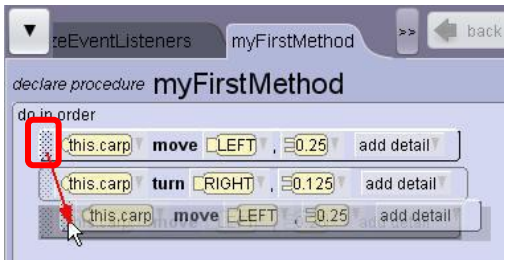
- tecla *Ctrl* + arrastrar
- tecla *Ctrl* + arrastrar el procedimiento al portapapeles
- botón derecho y la opción *Copy to Clipboard* (copiar al portapapeles)

Pasos para copiar el procedimiento de programación con la tecla Ctrl + arrastrar:

1. Presiona la tecla *Ctrl* y manténgala presionada.
2. Haga clic en el manejador de instrucción del procedimiento que desea copiar y arrastra el manejador de instrucción:
   a. al lugar que desea que se repita el procedimiento en el espacio de edición. Suelta el botón del *mouse* y luego la tecla *Ctrl*.

o

manejador de instrucción



   b. al portapapeles. Haga clic en el portapapeles y arrastra el procedimiento al espacio de edición en el lugar donde desea repetir el procedimiento. Suelta el *mouse*.

portapapeles (*clipboard*)

Pasos para copiar procedimiento de programación con el botón derecho del mouse:

1. Haga clic con el botón derecho del *mouse* sobre el manejador de instrucción del procedimiento que desea copiar.
2. Haga clic en la opción *Copy to Clipboard*.
3. Haga clic en el portapapeles y arrastra al espacio de edición en el lugar donde desea repetir el procedimiento.
4. Suelta el *mouse*.

El portapapeles cambia de color verde cuando el puntero del mouse hace contacto con el icono del portapapeles.

El portapapeles puede almacenar varios procedimientos de programación al mismo tiempo. Se arrastran desde el portapapeles en el orden opuesto al que estaban colocados. El portapapeles muestra el número de elementos que tiene almacenados.

1.9 - Prueba y Depuración de su Animación

Una vez que hayamos agregado o creado los procedimientos de programación para la animación, debemos probar la animación. Para probar la animación, hacemos clic en el botón *Run* (Ejecutar). Ejecutaremos la animación para probar que todo funciona correctamente, según lo previsto y sin errores. Debemos probar la animación con frecuencia durante el desarrollo.

Probar los límites del programa es una parte importante del proceso. Por ejemplo, cambiar el valor de un argumento en un procedimiento para "romper" intencionalmente el código demuestra si el código funciona en condiciones extremas. ¿Qué ocurre si un número es muy grande o negativo? Debemos probar los límites de la animación con frecuencia durante el desarrollo.

La **depuración** del programa se refiere al ciclo que implica: probar el programa, identificar errores o resultados no deseados, reescribir el código y volver a evaluar. Los programas de *software*, como las animaciones, se prueban mediante la introducción de comandos imprevistos para intentar "romper" el código. Cuando algo se "rompe" o no funciona según lo previsto en un programa de *software*, se denomina normalmente ***bug***. La depuración es el proceso por el que se buscan *bugs* en un programa de *software*.

Podemos utilizar las siguientes técnicas para programar la animación en *Alice*:

- Ajustar los argumentos que especifican la dirección, distancia y duración del movimiento de los objetos.
- Ajustar las expresiones matemáticas que manipulan la dirección, distancia y duración del movimiento de los objetos.
- Acotar o sustituir las instrucciones en el código que no funcionan como se esperaba.
- Resolver errores creados por el programador.

Actividad de Avalúo 1: Unidad 1

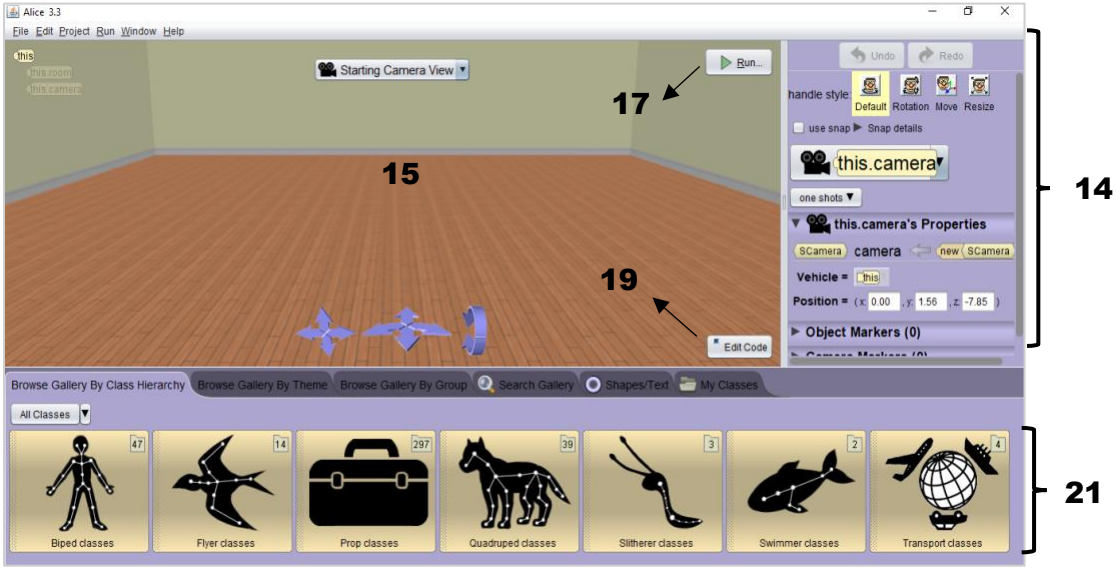
Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

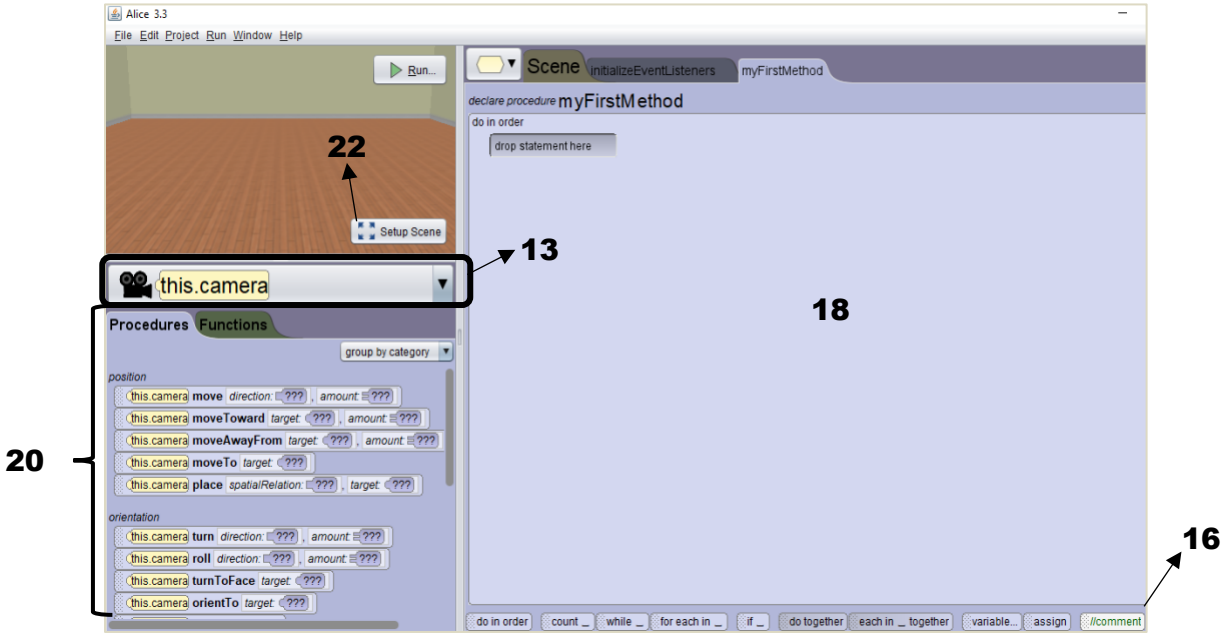
Ejercicio 1: Circule la(s) respuesta(s) correcta(s).

1. Un método para copiar un procedimiento de programación es utilizar la tecla de Ctrl +:
   a. portapapeles
   b. manejador de instrucción
   c. Copy to Clipboard
2. Una _____ es como se le conoce cuando se agrega el objeto al escenario.
   a. instancia del objeto
   b. instancia de una clase
   c. clase
3. Al arrastrar un objeto desde la galería con el mouse al escenario, ¿dónde se ubicará el objeto?
   a. lugar deseado en el escenario
   b. al centro del escenario
   c. a la izquierda del escenario
4. ¿Cuáles son las razones por las que debemos grabar frecuentes versiones de nuestro trabajo en Alice?
   a. Podemos utilizar la misma escena para crear animaciones diferentes.
   b. Ahorramos tiempo a la hora de crear la escena si se producen errores de programación.
   c. Podemos usar los menús en cascada para seleccionar el valor de cada uno de los argumentos.
5. Cuando vemos las instrucciones de programación en el panel izquierdo de la ventana, estamos en el:
   a. Editor de Escena
   b. Editor de Programación
   c. Editor de Códigos
6. Al buscar un proyecto que he trabajado días pasados, ¿en qué pestaña lo busco para abrirlo?
   a. My Proyects
   b. File System
   c. Recent

7. ¿Qué opciones se encuentran en el Editor de Escena?
    a. editar las propiedades de un objeto, ejecutar la animación, acceder al editor de códigos
    b. colocar objetos en el escenario, agregar, procedimientos de instrucción, acceder al editor de códigos
    c. ejecutar la animación, seleccionar objetos de la galería, colocar objetos en el escenario
8. Al hacer clic en un objeto para agregarlo al escenario, ¿qué ocurre?
    a. se ubica al centro del escenario
    b. se ubica en el lugar deseado en el escenario
    c. se muestra caja de diálogo para confirmar nombre del objeto
9. Es el ciclo que implica probar el programa, identificar errores o resultados no deseados, reescribir el código y volver a probar.
    a. procedimiento
    b. depurar
    c. ejecutar
10. Si de seo comenzar un proyecto con una plantilla que tenga varios objetos por defecto, voy a la pestaña:
    a. My Proyects
    b. File System
    c. Starters
11. Una plantilla de fondo, objetos estáticos y objetos en movimiento son los componentes de:
    a. la galería
    b. el menú de instancia
    c. la escena inicial
12. Al colocar el objeto en el escenario aparece un cuadro de diálogo para:
    a. confirmar la propiedad del nombre del objeto
    b. colocar la posición del objeto al centro del escenario
    c. adjudicar características al objeto

Ejercicio 2: Identifica cada nombre y su función o contenido según se especifique.





13. nombre, contenido y función
14. nombre y contenido
15. nombre
16. nombre
17. nombre y función
18. nombre
19. nombre y función
20. nombre
21. nombre y contenido
22. nombre y función

UNIDAD 2: ADICIÓN Y COLOCACIÓN DE OBJETOS

Estándar: Programa de Aplicación

Objetivos:
1. Identifica, evalúa, selecciona, instala, utiliza, actualiza, soluciona problemas y personaliza las aplicaciones.

2.1 - Abrir un Proyecto Existente

Los archivos guardados de proyectos con *Alice* se pueden abrir y editar. Hay tres maneras de abrir un archivo existente después de iniciar el *IDE*:

- Seleccionando el proyecto en la pestaña *My Projects*.
- Seleccionando el proyecto en la pestaña *Recent*.
- Buscando el proyecto mediante la pestaña *File System*.

Pasos para abrir un proyecto existente mediante la pestaña My Projects:

1. Acceda al programa *Alice*.
2. En el cuadro de diálogo *Select Project*, haga clic en la pestaña *My Projects*.
3. Busca y haga clic en el nombre o en la vista en miniatura del proyecto deseado.
4. Haga clic en el botón *Ok*.

Pasos para abrir un proyecto existente mediante la pestaña Recent:

1. Acceda al programa *Alice*.
2. En el cuadro de diálogo *Select Project*, haga clic en la pestaña *Recent*.
3. Busca y haga clic en el nombre o en la vista en miniatura del proyecto deseado.
4. Haga clic en el botón *Ok*.

Pasos para abrir un proyecto existente mediante la pestaña File System:

1. Acceda al programa *Alice*.
2. En el cuadro de diálogo *Select Project*, haga clic en la pestaña *File System*.
3. Haga clic en el botón *Browse*.
4. Busca el lugar donde guardó el proyecto.
5. Haga clic en el nombre del proyecto deseado.
6. Haga clic en el botón *Ok*.

2.2 - Posicionamiento de la Instancia del Objeto en el Escenario

El posicionamiento de la instancia en *Alice* se debe realizar desde el Editor de Escena. Cuando indicamos que vamos a posicionar las instancias, nos referimos a recrear cómo van a estar las instancias en el escenario para la historia que vamos a

contar. El posicionamiento de la instancia dentro del escenario incluye la selección de:

- Dirección que la instancia debe tener.
- Orientación de las instancias respecto de las demás instancias.
- Posición de las instancias.
- Posición de las sub-partes de la instancia (brazos, piernas, etc.).

Funciones de posicionamiento de las instancias del objeto

Todas las instancias de *Alice* comparten las mismas características de posicionamiento:

- Coordenadas 3D sobre ejes x, y & z.
- Un punto central, donde los ejes se cruzan (generalmente en el centro de masa).
- Sub-partes que se pueden mover.

Orientación de las instancias del objeto

Las instancias y sus sub-partes se mueven en relación con su propia orientación o sentido de dirección. Una instancia que mira hacia la parte de atrás de la escena, programado para avanzar 2 metros, se moverá 2 metros más hacia el fondo de la escena. Dado que las instancias no se mueven según la perspectiva del usuario, si vamos a codificar una instancia orientada hacia la cámara, imaginamos que se trata del reflejo en un espejo.

Maneras de posicionar una instancia

Hay dos formas de posicionar una instancia:

- Posicionamiento **preciso** mediante uno de estos dos métodos:
  - Utilizar un procedimiento *one-shot* (de instantánea)
  - Ingreso de los valores para las coordenadas x, y & z
- Posicionamiento **impreciso**
  - Método de arrastrar y soltar

Seleccionar la instancia para posicionar

Antes de posicionar la instancia del objeto, debemos seleccionar la misma. De esta manera, *Alice* puede aplicar las características de posicionamiento a la instancia que hayamos seleccionado. Hay tres maneras de seleccionar la instancia que desea posicionar:

- Hacer clic en el nombre de la instancia en el árbol de objetos.
- Hacer clic en la instancia en el escenario.
- Hacer clic en la flecha del menú de instancia en el panel de Propiedades y hacer clic en la instancia deseada.

Una vez hemos seleccionado la instancia, ésta se rodea con anillos o flechas.

Posicionamiento preciso: utilizando el procedimiento one-shot

Los **procedimientos *one-shot* (de instantánea)** se utilizan para realizar ajustes en la escena y posicionar las instancias. No se ejecutan cuando se selecciona el botón *Run* para reproducir la animación. Los procedimientos de instantánea son los mismos que se utilizan en el Editor de Código, sin embargo, solo se ejecutan una vez para reubicar la instancia, a diferencia de lo que ocurre en el Editor de Código, en donde se ejecutan cada vez que se hace clic en el botón *Run* para reproducir la animación. Tenemos dos opciones para abrir el menú de procedimientos de instantánea. Estas son:

- Hacer clic con el botón derecho (*right click*) en el objeto en el escenario y colocar el puntero en la opción procedimientos (*procedures*).

- En el Panel de Propiedades, hacer clic en en la flecha del botón *one shots* y colocar el puntero en la opción procedimientos (*procedures*).

Pasos para aplicar el posicionamiento preciso utilizando el procedimiento one-shots:

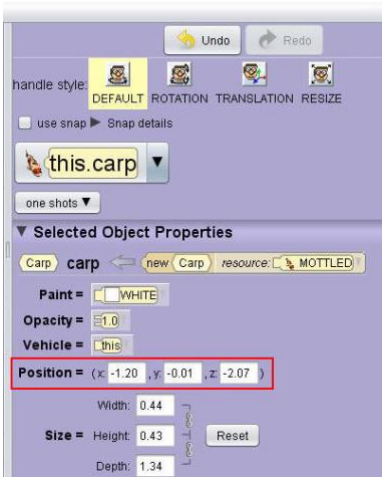
1. Haga clic en la instancia a posicionar.
2. Coloca el puntero en la opción *procedures*.
3. Coloca el puntero en la instrucción de programación deseada.
4. Haga clic en los argumentos y valores deseados.

La instancia se reubicará automáticamente según el procedimiento seleccionado y los argumentos especificados.

Podemos colocar las instancias en el centro del escenario y luego moverlas con procedimientos de instantánea a la ubicación y posición deseada.

Posicionamiento preciso: mediante las coordenadas x, y & z

En el panel de Propiedades se encuentra la propiedad *Position*. La propiedad ***Position*** indica el lugar donde está la instancia posicionada en el escenario en los ejes x, y & z. Las coordenadas ingresadas en los ejes determinarán la posición de la instancia del objeto de la siguiente forma:



- x: hacia la izquierda y hacia la derecha
- y: hacia arriba y hacia abajo
- z: hacia delante y hacia atrás

Pasos para aplicar el posicionamiento preciso mediante coordenadas x, y & z:

1. Haga clic en la instancia a posicionar.
2. En la propiedad *Position*, escriba la(s) coordenada(s) en el(los) eje(s) deseado(s).
3. Cada vez que escriba una coordenada, presiona la tecla *Enter*.

La instancia se reubicará automáticamente en las coordenadas introducidas en los campos de los ejes x, y & z. Es sumamente importante presionar la tecla *Enter* después de introducir cada valor, pues, de lo contrario, la instancia no se moverá a la posición especificada.

Posicionamiento impreciso: método de arrastrar y soltar

Este método suele ser utilizado para las instancias de escenario o instancias no importantes de la animación. El posicionamiento impreciso, no especifica la posición exacta de la instancia. Se utilizan los manejadores que se encuentran en el Panel de Propiedades. Cada manejador realiza una acción específica en el objeto seleccionado.

Estilos de estilos de manejadores:

- **Rotación simple y movimiento.**
- Permite arrastrar hacia adelante, hacia atrás, hacia la izquierda y hacia la derecha en el plano horizontal de la escena.

- **Girar alrededor de los ejes x, y, z.**
- Permite arrastrar los tiradores de torneado que rodean el objeto para girarlo a la izquierda y a la derecha, hacia delante y hacia atrás, y rodar el objeto hacia la izquierda y la derecha.

- **Desplazarse por los ejes x, y, z.**
- Permite arrastrar las flechas que aparecen para mover el objeto en la dirección indicada.

- **Cambiar el tamaño del objeto y extenderlo por los ejes x, y, z.**
- Permite arrastrar la flecha que aparece para cambiar el tamaño del objeto. Al hacer clic y arrastrar el propio objeto, también se cambiará el tamaño del objeto cuando se seleccione este estilo de control.

Cada estilo de manejador presenta anillos o flechas para ayudarnos con el posicionamiento.

Pasos para aplicar el posicionamiento impreciso mediante el método de arrastrar y soltar

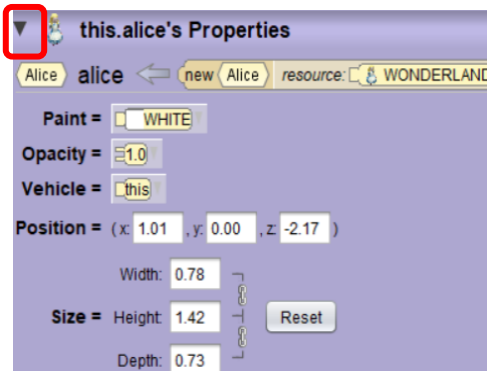
1. Haga clic en la instancia a posicionar.
2. En el Panel de Propiedades, haga clic en el estilo de manejador deseado.
3. Posiciona el objeto con el cursor seleccionado y arrastrando los anillos o flechas que lo rodean al lugar deseado.

Luego de utilizar el estilo de manejador, tiene que volver a dar clic en el estilo Default. De esta manera se restablece las propiedades de la instancia.

2.3 - Panel de Propiedades

En el Panel de Propiedades nos ofrece la posibilidad de cambiar las propiedades de la instancia seleccionada durante la configuración de la escena. En la flecha de *Selected Object Properties* podemos ocultar o mostrar las propiedades de una instancia. Si no se muestran los campos de propiedades de una instancia, hacemos clic en la flecha para mostrarlos.

Selected Object Properties



Otras propiedades que podemos utilizar, además, de cambiar la posición de la instancia son:

- *Paint*: aplicar un color a la instancia
- *Opacity*: aclarar los colores a la instancia hasta que haga el efecto de desaparecer (0.0)

- *Vehicle*: permite que una instancia sea el transporte de otra instancia (une dos instancias)
- *Size*: cambiar el tamaño a la instancia a una medida en específico (alto, ancho, profundidad)
- *Show Joints*: mostrar las articulaciones

Pasos para cambiar las propiedades de una instancia

1. Haga clic en la instancia deseada.
2. En el Panel de Propiedades, haga clic en la propiedad que desea cambiar.
3. Haga clic en la opción deseada o escriba los valores deseados. Recuerda que, si escribes el valor, debes presionar la tecla *Enter* para su aceptación.

2.4 - Posicionamiento de Sub-partes de una Instancia

El movimiento de las sub-partes de las instancias puede aportar a una instancia un aspecto mucho más realista al comienzo de la animación. Lograr la perfección en la manipulación de las sub-partes puede tomar algún tiempo, pero merece el esfuerzo, ya que produce una animación más natural. Podemos utilizar los estilos de manejador para posicionar las sub-partes de una instancia durante la configuración de la escena. Por ejemplo, puede que deseemos que la cabeza de una instancia mire a la derecha cuando comience la animación. Puedes hacer que las articulaciones se posicionen hacia la izquierda o derecha, hacia arriba o abajo, inclinarse a la izquierda o a la derecha.

Pasos para posicionar las sub-partes de la instancia:

1. En el Panel de Propiedades, haga clic en la flecha donde aparece el nombre de la instancia seleccionada.
2. Coloca el puntero sobre la flecha de la instancia deseada.
3. Haga clic en la sub-parte que desea modificar.

En el Panel de Propiedades aparecerá ahora el nombre con la sub-parte seleccionada. La instancia en el escenario aparecerá con tres aros (rojo, azul y verde) alrededor de la sub-parte seleccionada. Cada color de aro representa un posicionamiento diferente.

4. Haga clic en el aro correspondiente y posiciona la sub-parte con el cursor seleccionado y moviendo el aro correspondiente al posicionamiento deseado.

Ejemplo: Posicionar cabeza de la instancia *CardOne* hacia la derecha.

Paso núm. 1

Paso núm. 2

Paso núm. 3

Paso núm. 4

Resultado

Actividad de Avalúo 2: Unidad 2

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Selecciona la descripción para cada término

23. desplazarse por los ejes x, y & z	a. resize
24. posicionamiento de la instancia	b. cambiar alto, ancho y profundidad
25. ingreso de valores para las coordenadas x, y & z	c. move
26. estilos de manejadores	d. posicionamiento impreciso
27. rotación simple	e. Editor de Escena
28. propiedad size	f. posicionamiento preciso
29. posicionamiento one-shot	g. default
30. girar alrededor de los ejes x, y & z	h. rotation
31. cambiar tamaño de la instancia	
32. método de arrastrar y soltar	

Ejercicio 2: Coloca los pasos en el orden en que se deben llevar a cabo

33. Cambiar las propiedades de una instancia
    a. En el Panel de Propiedades, haga clic en la propiedad que desea cambiar.
    b. Haga clic en la instancia deseada.
    c. Haga clic en la opción deseada o escriba los valores deseados.

34. Pasos para posicionar las sub-partes de la instancia
    a. Haga clic en la sub-parte que desea modificar.
    b. Haga clic en el aro correspondiente y posiciona la sub-parte con el cursor seleccionado y moviendo el aro correspondiente al posicionamiento deseado.
    c. Coloca el puntero sobre la flecha de la instancia deseada.
    d. En el Panel de Propiedades, haga clic en la flecha donde aparece el nombre de la instancia seleccionada.

35. Pasos para aplicar el posicionamiento preciso utilizando el procedimiento one-shots
    a. Haga clic en la instancia a posicionar.
    b. Haga clic en los argumentos y valores deseados.
    c. Coloca el puntero en la instrucción de programación deseada.
    d. Coloca el puntero en la opción procedures.

36. Pasos para abrir un proyecto existente mediante la pestaña Recent
    a. Busca y haga clic en el nombre o en la vista en miniatura del proyecto deseado.
    b. Acceda al programa Alice.
    c. Haga clic en el botón Ok.
    d. En el cuadro de diálogo Select Project, haga clic en la pestaña Recent.

37. Pasos para aplicar el posicionamiento preciso mediante coordenadas x, y & z
    a. Cada vez que escriba una coordenada, presiona la tecla Enter.
    b. En la propiedad Position, escriba la(s) coordenada(s) en el(los) eje(s) deseado(s).
    c. Haga clic en la instancia a posicionar.

38. Pasos para abrir un proyecto existente mediante la pestaña File System
    a. En el cuadro de diálogo Select Project, haga clic en la pestaña File System.
    b. Haga clic en el botón Browse.
    c. Haga clic en el nombre del proyecto deseado.
    d. Busca el lugar donde guardó el proyecto.
    e. Acceda al programa Alice.
    f. Haga clic en el botón Ok.

UNIDAD 3 PROCEDIMIENTOS Y ARGUMENTOS

Estándar: Programa y desarrollo de aplicaciones

Objetivos: 1. Diseña, desarrolla, prueba e implementar programas y aplicaciones.

3.1 - Visualización del Editor de Código

En el material anterior discutido, aprendimos que *Alice* le permite trabajar con dos perspectivas distintas: un Editor de Escena y con un Editor de Código. En el **Editor de Código** es donde se programa la animación. Por defecto al acceder al programa y/o a un proyecto ya existente, *Alice* nos muestra la perspectiva del Editor de Código. Si estamos en la perspectiva del Editor de Escena, hacemos clic al botón *Edit Code*.

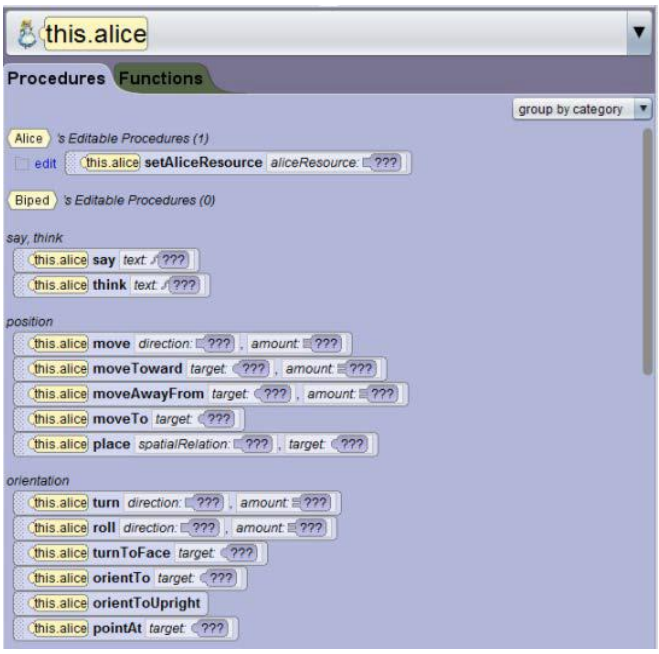
También aprendimos que siempre debemos seleccionar la instancia del objeto antes de poder modificar cualquiera de sus propiedades de posicionamiento. Esto también aplica cuando deseamos programar una instancia. Para seleccionar la instancia que desea programar, podemos utilizar cualquiera de las maneras ya discutidas anteriormente. Para asegurarnos que la instancia deseada es la que está seleccionada, debemos observar que tenga un círculo color amarillo debajo de la instancia. Este paso es sumamente importante, pues de esta forma, nos aseguramos de que estamos creando una instrucción de programación para la instancia correcta.

3.2 - Panel de Métodos

Al seleccionar una instancia del objeto en el Editor de Código, podremos acceder al Panel de Métodos. En el **Panel de Métodos** se encuentran todas las sentencias de programación de *Alice*, las cuales incluyen procedimientos para las acciones y funciones de información. El Panel de Métodos se divide en dos pestañas:

- ***Procedures* (Procedimientos)**: Todos los procedimientos predefinidos para un objeto.
- ***Functions* (Funciones)**: Todas las funciones predefinidas para un objeto.

Pestaña *Procedures* (Procedimientos)



Recordamos que un **procedimiento** es una parte del código del programa que define la forma en que se debe ejecutar el objeto. *Alice* tiene un juego de procedimientos para cada clase; sin embargo, los usuarios pueden crear (declarar) nuevos procedimientos. La pestaña ***Procedures*** muestra los procedimientos predefinidos para la instancia seleccionada, así como los procedimientos creados por el propio usuario.

Pestaña Functions (Funciones)

Una **función** calcula y responde a una pregunta sobre un objeto, como, por ejemplo:

What is its width or height? (¿Cuál es su ancho o altura?), o *What is its distance from another object?* (¿Cuál es su distancia desde otro objeto?). *Alice* dispone de un juego de funciones para cada clase; sin embargo, los usuarios pueden declarar nuevas funciones. La pestaña ***Functions*** muestra las funciones predefinidas para la instancia seleccionada, así como las funciones creadas por el propio usuario.

3.3 - Editor de Código

En el espacio de edición, en la parte superior, encontraremos el botón de jerarquía de clases, y las pestañas *Scene*, *initializeEventListeners* y *myFirstMethod.*

En la pestaña *myFirstMethod* podemos agregar las instrucciones de programación. Por defecto, *Alice* crea una sentencia de control *do in order* en el espacio de edición. El área denominada ***drop statement here*** es la ubicación en la que debemos colocar las instrucciones de programación. *Alice* ejecuta los procedimientos de programación en un orden procesal. El procedimiento de programación del principio se ejecutará en primer lugar y, a continuación, los siguientes procedimientos hasta que alcance el final de los procedimientos de programación agregados en el espacio de edición.

Sentencias de control

En la parte inferior de la pestaña *myFirstMethod* en las estructuras de programación, encontraremos las sentencias de control de *Alice*. Para colocar estas sentencias de control al espacio de edición, podemos hacer clic y arrastrarlas a la posición que deseemos.

3.4 - Movimiento de Objetos

El movimiento de las instancias es egocéntrico. Lo que significa que las instancias se mueven en función de la dirección hacia la que están colocadas. Una instancia, así como sus sub-partes, se pueden mover en seis direcciones: hacia arriba, hacia abajo, hacia adelante, hacia atrás, hacia la derecha y/o hacia la izquierda.

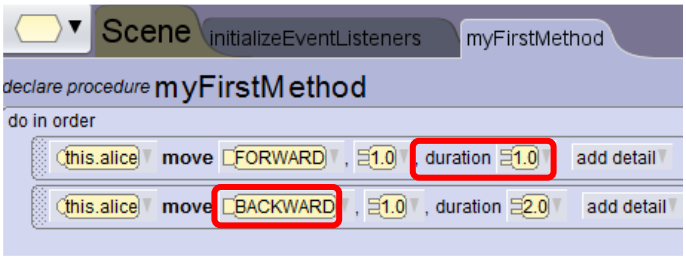
Algunos ejemplos de procedimientos de movimiento son:

Procedimiento	Descripción
Move	La instancia se desplaza en cualquiera de sus seis direcciones.
Move Toward	La instancia se desplaza hacia otra instancia.
Move Away From	La instancia se aleja de otra instancia.
Move To	La instancia se desplaza de su posición actual al punto central de la instancia de destino.

Procedimiento	Descripción
Move and Orient To	La instancia se desplaza de su posición actual al punto central de la instancia de destino y ajusta la orientación de la instancia para que coincida con la orientación de la instancia de destino.
Delay	Se detiene el movimiento de una instancia durante un determinado número de segundos. *Delay* se puede utilizar para ralentizar la ejecución de una animación.
Say	Crea una burbuja de llamada con texto que hace que parezca que el objeto habla.

3.5 - Argumentos

Aprendimos en el material discutido los pasos para declarar un procedimiento de programación. Una vez declaramos un procedimiento de programación, utilizamos la lista desplegable para definir los valores de cada argumento. Un **argumento** es un valor que utiliza el procedimiento para completar su tarea. Los valores de argumentos



de cualquier procedimiento de programación agregada al editor de códigos se pueden cambiar en cualquier momento. Los tipos de argumentos pueden incluir: privilegios, dirección, cantidad de dirección, tiempo de duración, texto.

La lista desplegable de argumentos ofrece valores por defecto que se pueden seleccionar. Si ninguno de los valores por defecto es adecuado, podemos seleccionar otros valores para especificar un valor de argumento más preciso.

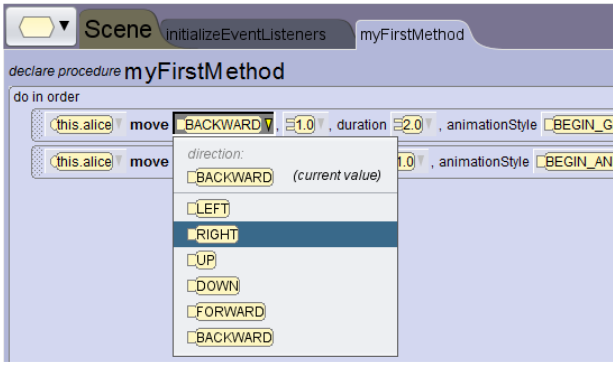
Argumentos como marcadores de posición

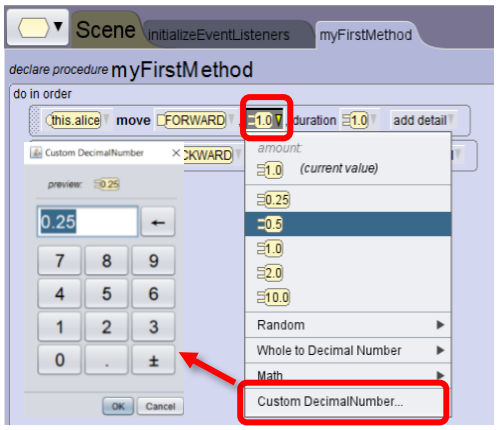
Al colocar un procedimiento de programación en el espacio de edición, se deben especificar todos los valores del argumento. Puede ocurrir que un valor de argumento seleccionado como valor de marcador de posición temporal se sustituya más tarde. Por ejemplo, puede que deseemos que un objeto se mueva hacia delante, pero que no estemos seguros de la distancia. En otras palabras, seleccionamos un valor de marcador de posición de 2 metros, ejecutamos la animación y determinamos que se necesita un valor diferente. Procederemos entonces a modificar el valor siguiendo

los pasos para especificar un valor de argumento. También podemos especificar un valor de marcador de posición que sustituiremos más tarde por una función o una variable.

Pasos para cambiar un valor de argumento:

1. Haga clic en la flecha del argumento deseado. Se mostrará una lista desplegable con los valores por defectos.
2. Haga clic en el valor deseado. Si desea un valor en específico:
   a. Haga clic en *Custom Decimal Number*
   b. Haga clic o escriba el valor deseado.
   c. Haga clic en el botón *Ok*.





3.6 - Ejecución del Programa

Al igual que en todos los entornos de programación, es recomendable codificar pequeñas animaciones y probarlas de forma periódica. No debemos declarar, crear una gran cantidad de procedimientos para luego probar que funciona. Para ejecutar las instrucciones de programación, hacemos clic en el botón *Run*. Debemos ejecutar el programa con frecuencia para probar si se obtienen los resultados deseados y modificar los valores de los argumentos según sea necesario.

3.7 - Procedimientos de Programación

Los procedimientos de programación son las distintas instrucciones que daremos a la instancia del objeto. Las mismas combinan procedimientos y funciones que arrastramos desde el panel de métodos y sus respectivos argumentos definidos en la sentencia *do in order*. Una vez colocadas en el espacio de edición, los

procedimientos de programación se pueden reordenar, es decir, reubicar, cambiar de orden, mover a otro lugar.

Pasos para reordenar procedimientos de programación:

1. Presiona la tecla *Ctrl* y manténgala presionada.
2. Haga clic en el manejador de instrucción del procedimiento que desea reordenar y arrastra el manejador con el mouse a su nueva posición.
3. Suelta el botón del *mouse* y luego la tecla *Ctrl*.

Se mostrará un indicador de posición verde para ayudarnos a alinear la sentencia de programación con la posición deseada.

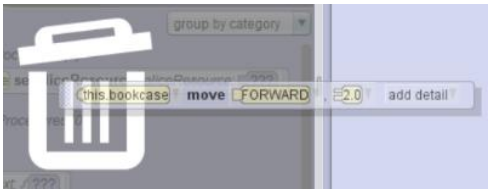
3.8 - Eliminar y/o Desactivar Procedimientos de Programación

Una vez creado (declarado) el procedimiento de programación podemos eliminarlo. También podemos eliminar sólo parte del procedimiento de programación. La eliminación de partes del procedimiento evita tener que volver a crear el procedimiento desde el principio. Esta es una herramienta muy útil en *Alice* que ahorra gran cantidad de tiempo. Recordamos que, si eliminamos un procedimiento de programación y después cambiamos de opinión, siempre podemos utilizar la opción de deshacer (CTRL+Z) para volver a activarlo. Hay dos maneras de eliminar todo el procedimiento de programación:

- clic y arrastre
- botón derecho del mouse (*right click*)

Pasos para eliminar el procedimiento de programación mediante el clic y arrastre:

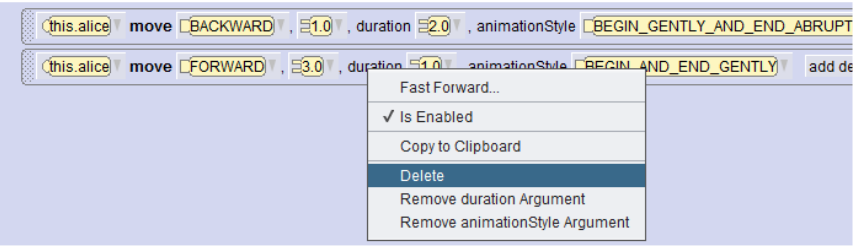
1. Haga clic en el manejador de instrucción del procedimiento que desea eliminar y arrastra con el mouse el manejador hasta el Panel de Métodos. Aparecerá la imagen del zafacón.



2. Suelta el botón del *mouse*.

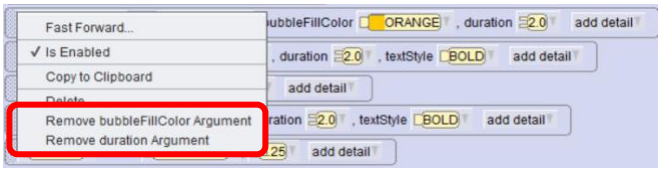
Pasos para eliminar el procedimiento de programación mediante el right click:

1. Haga *right click* en el manejador de instrucción del procedimiento que desea eliminar.
2. En la lista desplegable, haga clic en la opción *Delete*.



Pasos para eliminar parte del procedimiento de programación:

1. Haga *right click* en el manejador de instrucción del procedimiento que desea eliminar.
2. Haga clic en la opción *Remove* con el nombre del argumento en el procedimiento que deseas eliminar.



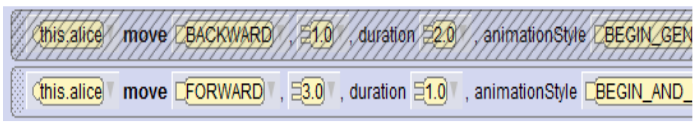
Desactivación de procedimientos de programación

La desactivación de bloques de código permite ahorrar mucho tiempo, debido a que no tendremos que estar constantemente probando procedimientos que ya se han probado. Los procedimientos de programación se pueden desactivar en el Editor de Código. Desactivaremos procedimientos de programación para:

- Ayudar a aislar partes del procedimiento durante la prueba.
- Ayudar a centrarnos en la programación, pruebas y depuración de un procedimiento específico.

Pasos para desactivar procedimientos de programación:

1. Haga *right click* en el manejador de instrucción del procedimiento que desea desactivar.
2. Haga clic en la opción *Is Enabled*.



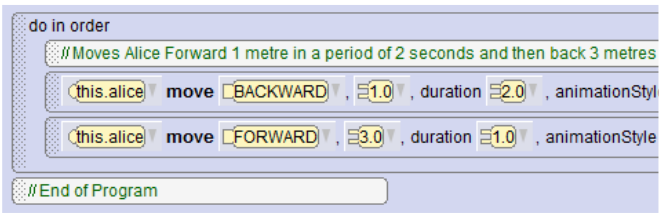
El color del procedimiento cambiará a marcas *hash* grises para indicar que está desactivado.

Si deseo volver a activar los procedimientos, repita los pasos. El procedimiento de programación aparecerá sin las marcas grises. Luego volvemos a activar todos los procedimientos para las pruebas finales.

3.9 - Depuración

Recordamos que, luego de reordenar, editar o eliminar un procedimiento de programación debemos ejecutar el programa para probar la animación. Si encontramos cualquier error en una prueba de ejecución, debemos editar el procedimiento. Anteriormente mencionamos que este ciclo se conoce como depuración. **Depuración** es el proceso por el que se ejecuta varias veces la animación, y se ajustan las sentencias de control, los procedimientos y los argumentos después de cada ejecución. También debemos recordar guardar a menudo durante la depuración del programa.

3.10 - Comentarios de Programación



Los comentarios son esenciales para comprender el funcionamiento del código. Se recomienda agregar los comentarios conforme crea el procedimiento de programación. Nunca deje los comentarios para el final, es mucho más fácil agregarlos durante la codificación. Incluir comentarios de programación en una animación ayuda a las personas a comprender el flujo de la programación. Los **comentarios**:

- Describen la intención de los procedimientos de programación.
- No afectan a la funcionalidad del programa ya que se ignoran durante su ejecución.
- Normalmente se colocan por encima del bloque de los procedimientos de programación que describen.
- Se suelen escribir en primer lugar, en programas de gran tamaño, como un esquema de las instrucciones de programación.

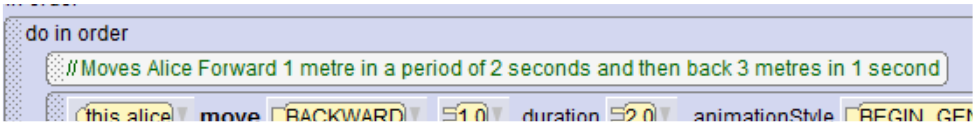
Los comentarios pueden ser una excelente herramienta para organizar el desarrollo de un programa. Para los programas de gran tamaño, podemos crear un comentario que indique el final del programa. El comentario *end of program* ayuda a minimizar

el desplazamiento al agregar procedimientos de programación a un procedimiento *myFirstMethod* largo.

Los comentarios no se compilan como parte del procedimiento, de modo que no tienen ningún efecto en cómo se ejecuta el programa. Podemos agregar tantos comentarios como deseemos. Estos se pueden utilizar para explicar la función del procedimiento, así como su diseño. Los comentarios se muestran en color verde.

Pasos para agregar comentarios:

1. En las estructuras de programación, haga clic en el botón *//comment* y arrastra el botón hasta colocarlo encima del procedimiento de programación que tendrá el comentario.
2. Suelta el botón del mouse.



Actividad de Avalúo 3: Unidad 3

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Circule la respuesta correcta.

39. El método de _____ es una manera para eliminar el procedimiento de programación.
    a. right click
    b. Delete en la barra de menú
    c. Remove

40. Son valores que indican al procedimiento cómo realizar su tarea (privilegios, dirección, velocidad, etc.).
    a. argumentos
    b. funciones
    c. códigos

41. Los comentarios se encuentran en el(la):
    a. panel de métodos
    b. menú de instancia
    c. estructura de programación

42. La perspectiva _____ es la que se muestra por defecto al abrir el Alice.
    a. Editor de Escena
    b. Editor de Código
    c. Galería

43. Es el proceso en el que se ajustan los procedimientos de programación y los argumentos después de cada ejecución.
    a. depuración
    b. ejecutar
    c. reordenar

44. Cuando indicamos que el movimiento de las instancias es egocéntrico, nos referimos que:
    a. se detiene el movimiento durante un determinado número de segundos
    b. se desplaza hacia otra instancia
    c. se mueven en función de la dirección hacia la que están colocadas

45. Un(a) _____ es una parte del código del programa que define la forma en que se debe ejecutar el objeto.
    a. código
    b. sentencia de control
    c. procedimiento

46. Un argumento como marcador de posición lo que indica es una:
    a. posición temporal que se sustituirá más tarde
    b. posición específica que tiene que tener esa instancia
    c. posición alterna entre dos valores

47. La sentencia de control _____ es la que está por defecto en el espacio de edición.
    a. do together
    b. do in order
    c. variable

48. Debemos _____ el programa con frecuencia para verificar si se obtienen los resultados deseados.
    a. salvar
    b. depurar
    c. ejecutar

Ejercicio 2: Menciona.

49. La opción que nos permite cambiar el valor del argumento a un valor en específico.

50. Tres ejemplos de procedimientos de movimiento y su descripción.

51. La ubicación que nos permite con la tecla Ctrl reordenar un procedimiento de programación.

52. La opción que nos permite desactivar procedimientos de programación.

53. Hacia donde debes dirigirte para eliminar el procedimiento de programación.

UNIDAD 4: Rotación y Asignación Aleatoria

Estándar: Programa y desarrollo de aplicaciones

Objetivos: 1. Diseña, desarrolla, prueba e implementar programas y aplicaciones.

4.1 - Desarrollo del Programa Mediante Enfoque en Sentido Descendente

Al igual que un constructor utiliza un juego de planos y sigue una serie de pasos, un programador utiliza un plan y sigue una serie de pasos para crear un programa. Los constructores trabajan con un propósito y definen objetivos específicos durante la construcción: que las habitaciones cuenten con eficiencia energética, que la casa cumpla con los códigos de construcción, etc. Los programadores también trabajan con un propósito y definen objetivos, porque sin ellos, no se puede medir el éxito.

Pasos para utilizar un enfoque en sentido descendente de la programación:

1. Defina el escenario de alto nivel (el propósito o meta del programa).
2. Documenta las acciones paso a paso en un guion gráfico textual. Esto ayuda a comprender totalmente las acciones que se deben programar.
3. Crea una tabla para alinear los pasos del guion gráfico con las instrucciones de programación.
4. Revisa la tabla durante el desarrollo de la animación para asegurarse de que cumple las especificaciones.
5. Continúa revisando la tabla según sea necesario.

Para este propósito se alinea el guion gráfico (*storyboard*) textual, se estudiará en detalle en la próxima sección, con las instrucciones de programación. Veamos un ejemplo del guion gráfico textual con las instrucciones de programación.

Ejemplo de storyboard textual

- Programar las siguientes acciones en orden:
  - El conejito camina.
  - Alice dirige la mirada al conejito.
  - El conejito dice: "Is this the party?".
  - Alice dice: "¡Oh, no!".
  - El conejito gira a la izquierda.
  - Programar las siguientes acciones conjuntamente:
    - El conejito se aleja rápidamente.
    - Alice mueve la cabeza.

Cuanto más exacto sea el guion gráfico textual, más fácil será la codificación. Debe asegurarse de detallar cada paso individual del escenario.

Se crea una tabla donde se muestra cómo las acciones definidas en el guion gráfico se relacionan directamente con los procedimientos de programación.

Orden de las instrucciones	Acción del guión gráfico	Instrucciones de programación
Do in order	El conejito camina.	El conejito avanza 2 metros.
	Alice dirige la mirada al conejito.	Alice se gira hacia el conejito.
	El conejito dice: "Is this the party!".	El conejito dice: "Is this the party!".
	Alice dice: "¡Oh, no!".	Alice dice: "¡Oh, no!".
	El conejito gira a la izquierda.	El conejito gira a la izquierda 0,25.
Do together	El conejito se aleja rápidamente. Alice mueve la cabeza.	El conejito avanza 4 metros en 1 segundo. Alice gira la cabeza de izquierda a derecha y otra vez a la izquierda.

4.2 - Movimiento de Instancias

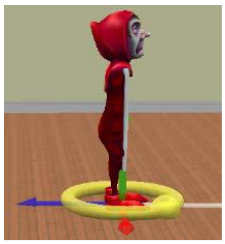
En unidades anteriores, vimos que el movimiento de las instancias es egocéntrico: las instancias se mueven en función de la dirección hacia la que están colocadas. Una instancia se puede mover en seis direcciones: hacia arriba, hacia abajo, hacia adelante, hacia atrás, hacia la derecha y hacia la izquierda. Los procedimientos de rotación *turn* y *roll* se utilizan también para asignar movimientos a las instancias en *Alice*.

Algunos ejemplos de procedimientos de rotación son:

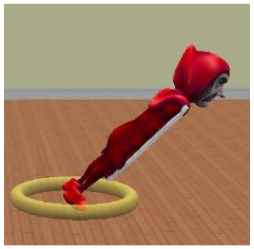
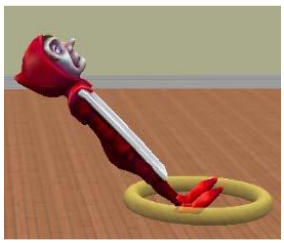
Procedimiento	Descripción
Turn	Gira una instancia hacia la izquierda, derecha, adelante o atrás respecto a su punto central. Giro a la izquierda y a la derecha sobre el eje vertical de la instancia; giro hacia delante y hacia atrás sobre el eje horizontal de la instancia.
Roll	Una instancia rueda a la izquierda o a la derecha sobre el punto central con el eje horizontal de la instancia.

Turn permite:

- girar una instancia hacia la izquierda y a la derecha, respecto a su punto central con un eje vertical.

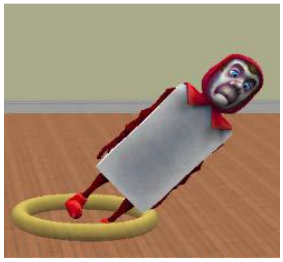
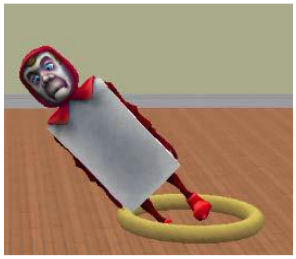



- girar la instancia hacia adelante y hacia atrás sobre su punto central con un eje horizontal.

Roll permite:

- rodar hacia la izquierda y hacia la derecha sobre su punto central con un eje horizontal.

Pasos para agregar el procedimiento de rotación:

1. Haga clic en la instancia deseada.
2. En el Panel de Métodos, haga clic en la instrucción de programación *turn* o *roll* y arrastra la instrucción al espacio de edición.
3. Haga clic en el argumento de la dirección deseada.
4. Haga clic en el valor de cantidad deseada (1.0 = un giro o vuelta completa).
5. Probar y depurar la animación agregada.

4.3 - Movimiento y Rotación de Sub-partes

Algunas instancias tienen sub-partes móviles que pueden girar y rotar. Las sub-partes de una instancia muestran anillos que indican su rango de movimiento. La rotación como el movimiento se pueden aplicar a una instancia completa o a las sub-partes que seleccionemos. El movimiento de las sub-partes de la instancia aporta a la animación un aspecto más realista. Por ejemplo, el reloj tiene manecillas de horas y minutos que pueden rodar sobre el punto central del reloj. La clave de una rotación correcta es conocer el punto central de una instancia.

Programar las sub-partes de una instancia

Al igual que podemos posicionar las sub-partes de la instancia antes de la animación (en el editor de escena), podemos agregar un procedimiento en el editor de códigos para que la instancia realice una animación (ejemplo: una persona mueva las piernas, brazos, cabeza, etc.) al ejecutar el programa.

Pasos para agregar un procedimiento las sub-partes de la instancia:

1. Haga clic en la flecha del menú de instancia.
2. Coloca el puntero sobre la flecha de la instancia deseada.
3. Haga clic en la sub-parte que desea modificar.

Aparece el nombre con la sub-parte seleccionada y la instancia en la escena aparece con unos aros alrededor de la sub-parte. Cada color de aro representa un posicionamiento diferente.

4. Haga clic en la instrucción de programación deseada y arrastra la instrucción al espacio de edición.
5. Haga clic en los argumentos y valores deseados, según la instrucción agregada.

Una vez se ejecute la animación, podrás observar el movimiento de la sub-parte, según el procedimiento utilizado.

6. Probar y depurar la animación agregada.

4.4 - Sentencias de Control

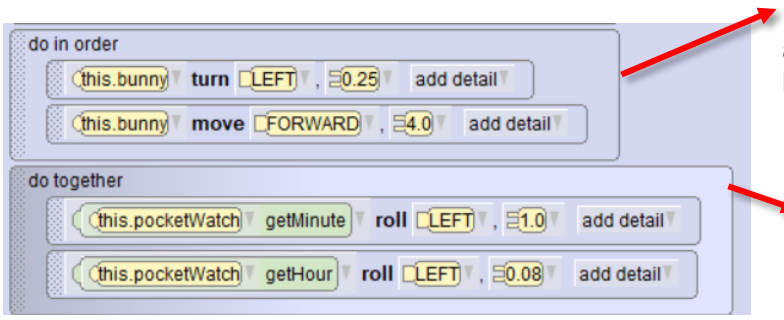
En las unidades anteriores discutimos que las instrucciones de programación se colocan en el espacio de edición dentro de una sentencia de control llamada *do in order*. A continuación, veremos cómo utilizar las **sentencias de control** para indicarle al programa cómo debe ejecutar estas sentencias de programación. Por defecto, en el espacio de edición de la pestaña *myFirstMethod* contiene una sentencia de control *do in order*. En esta sentencia, toda instrucción que demos al programa se ejecutará en orden secuencial. Sin embargo, *Alice* nos permite programar acciones de muchas otras formas. Las sentencias de control que más frecuentemente utilizamos son:

- ***do in order***: ejecutar los procedimientos de instrucción en el orden en que están. Este es el principio FIFO, primero en entrar primero en salir.
- ***do together***: ejecutar los procedimientos de instrucción simultáneamente; hacerlo juntos; a la misma vez.
- ***if/else***: ejecutar un procedimiento mediante una condición si es verdadera o falsa.
- ***count***: ejecutar los procedimientos de instrucción un número determinado de veces.
- ***while***: ejecutar los procedimientos de instrucción mientras se cumpla una condición especificada.

Las sentencias de control se encuentran en la parte inferior del editor de códigos. Al arrastrarlas al espacio de edición representarán bloques de código en los que se agrupan sentencias individuales.

Los procedimientos dentro de la sentencia *do in order* se van a ejecutar en el orden secuencial en que estén en el bloque. O sea, que una vez finalice la primera animación, continúa la siguiente y así sucesivamente hasta finalizar con la última animación en esa sentencia de control. Cuando se agrega varios procedimientos

dentro de una sentencia de control *do together*, el programa los ejecutará simultáneamente, a la misma vez.



Pasos para agregar una sentencia de control:

1. Haga clic en el botón de la sentencia deseada y arrastra el botón hasta colocarlo en el lugar deseado.
2. Suelta el botón del mouse.
3. Agrega la instrucción de programación dentro de la sentencia agregada en el espacio de edición.

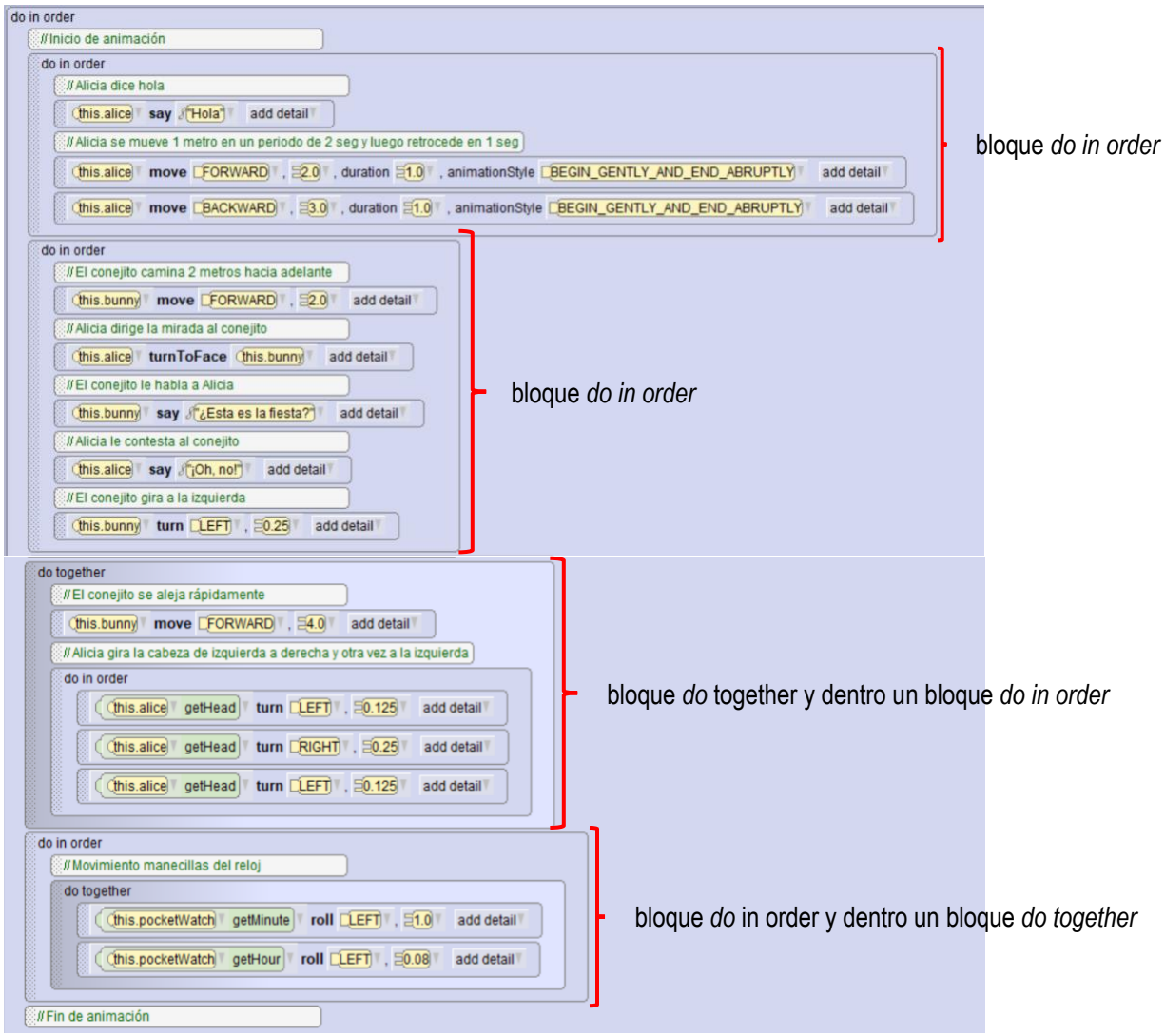
También podemos agregar las sentencias de control luego de agregado el procedimiento. Solo tenemos que arrastrar el procedimiento a la sentencia de control agregada en el espacio de edición.

Bloques de programación

Aunque en el espacio de edición de la pestaña *myFirstMethod* contiene una sentencia de control *do in order* por defecto, siempre podemos agregar otras sentencias de control *do in order* al área de programación. Esto permite mantener los procedimientos organizados en bloques. Los bloques se pueden copiar fácilmente en el portapapeles o moverse de un lugar a otro en un programa de gran tamaño. Las sentencias de control *do in order* anidadas mejoran el rendimiento de la animación y facilitan la edición del programa al programador.

La **anidación** es el proceso por el que se coloca algo dentro de algo, como los huevos colocados unos junto a otros en un nido. Las sentencias anidadas agregan una estructura visual a un programa, de forma muy similar a la que un esquema incorpora una estructura a un informe. A continuación, se anidan varias sentencias *do in order* y *do together*. Observamos como todos los bloques resultantes tienen comentarios describiendo la acción que desempeñarán.

Ejemplo:



4.5 - Números Aleatorios

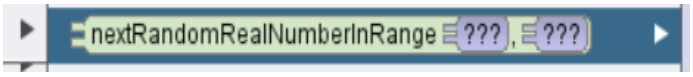
Los números aleatorios son los números generados por el equipo con un patrón predecible para su secuencia. Estos se generan dentro de un determinado rango de números. Los equipos pueden necesitar números aleatorios para:

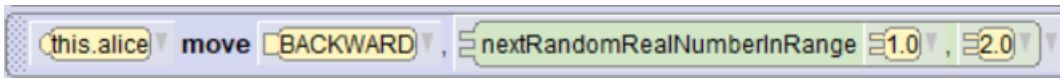
- **Seguridad**: por ejemplo, contraseñas generadas aleatoriamente.
- **Simulación**: por ejemplo, modelación de ciencias de la tierra (es decir, erosión a lo largo del tiempo).

El rango de números aleatorios abarca tanto positivos como negativos, y puede incluir números enteros y partes fraccionarias de un número. Incluso, es normal que se repita el mismo número.

En la vida real, los animales, por ejemplo, no se mueven en línea recta, o en líneas geométricas. Cambian de dirección ligeramente a medida que caminan, nadan y vuelan. Podemos emplear los números aleatorios en el argumento de distancia de un método para que la instancia se mueva de forma menos previsible (y, por lo tanto, más realista). Los números aleatorios también se suelen utilizar para crear juegos que necesitan comportamientos impredecibles.

Pasos para aplicar un número aleatorio:

1. En el área del editor de códigos, haga clic en la flecha del argumento con el valor al que le aplicaremos el número aleatorio.
2. Coloca el puntero en la opción *Random* en la lista desplegable.
3. Coloca el puntero en la opción 

4. Coloca el puntero en el primer valor deseado.
5. Mueva el puntero hacia la derecha y haga clic en el segundo valor.
   Se registrarán los dos valores dentro del procedimiento
   

6. Probar y depurar los cambios realizados en el procedimiento.
   Si deseamos aplicar otro valor que no sea el que se muestra por defecto, haga clic en la opción *Custom Decimal Number* y escriba el primer y segundo valor.

Actividad de Avalúo 4: Unidad 4

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Completa la oración con la información correcta.

54. Un procedimiento que depende de que la condición sea verdadera para ejecutarse se logra con la sentencia de control _____.
55. El procedimiento de rotación _____ permite a la instancia moverse de izquierda y derecha sobre su punto central con un eje horizontal.
56. Al ejecutarse unos procedimientos a la misma vez, se utilizó la sentencia de control _____.
57. Al aplicar el valor de _____ al argumento de turn o roll, la instancia hará una vuelta completa.
58. El proceso para poner una cosa dentro de otra, se llama _____.
59. Las _____ le indican al programa como debe ejecutar los procedimientos de programación.
60. Los _____ se utilizan para crear juegos que necesitan comportamientos impredecibles.
61. El procedimiento de rotación _____ permite una instancia moverse respecto a su punto central con un eje vertical hacia la derecha o hacia la izquierda.
62. La sentencia de control _____ es ejecutada cuando el procedimiento se ejecuta por un número determinado de veces.
63. Las _____ de una instancia muestran anillos que indican su rango de movimiento.
64. El movimiento de las instancias es _____.
65. Para lograr un enfoque en sentido descendente de la programación el _____ se alinea con las instrucciones de programación.
66. Los procedimientos dentro de la sentencia de control _____ se ejecutan con el principio FIFO.
67. El procedimiento de rotación turn permite que la instancia gire hacia adelante y hacia atrás sobre su punto central con un eje _____.
68. La sentencia de control _____ se ejecuta mientras se cumpla una condición.

UNIDAD 5: DECLARACIÓN DE PROCEDIMIENTOS

Estándar: Programa y desarrollo de aplicaciones

Objetivos:
1. Diseña, desarrolla, prueba e implementar programas y aplicaciones.

5.1 - Escenarios y Animaciones

Los animadores profesionales comienzan su proceso con el desarrollo de un escenario, o una historia, que proporciona un objetivo a la animación. El escenario es la idea principal que hay detrás de la animación. Definir el escenario, y la animación para representar el escenario, es el primer paso de la programación de la animación. Por ejemplo, podemos querer contar una historia que representa un conflicto y una resolución, o impartir una lección que enseña un concepto matemático. Quizás queremos desarrollar un proceso para simular o demostrar, o un juego para entretenimiento o formación. Para cada uno de estos tipos de escenario, veamos algunas ideas que podrían concretarse:

Tipo de escenario	Escenario	Animación
Historia	Un gato necesita ayuda para bajar de un árbol.	Un bombero sube al árbol para salvar al gato.
Lección	Memorizar símbolos químicos es difícil.	Un juego cronometrado para hacer coincidir símbolos químicos con sus definiciones.
Proceso	Un auto tiene un neumático desinflado.	Una demostración que muestre cómo cambiar un neumático en un auto virtual.
Juego	Un avión debe evitar los objetos que se encuentran en su ruta conforme vuela.	Un juego interactivo para pilotar un avión alrededor de los objetos del cielo.

5.2 - Guiones Gráficos

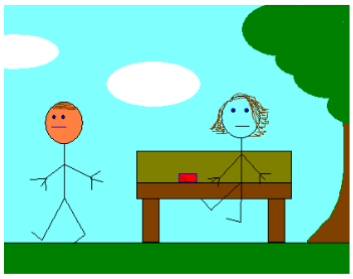
Un **guion gráfico** identifica las especificaciones de diseño del escenario de la animación: aspecto de los objetos, cómo se mueven, hablan, interactúan, etc. Una vez definido el escenario, se puede comenzar a desarrollar el guion gráfico de la animación. Para planificar una animación se utilizan dos tipos de guiones gráficos:

- **Visual**: Muestra una serie de imágenes ilustradas que representan las principales escenas de la animación.
- **Textual**: Lista detallada y ordenada de las acciones que realiza cada uno de los objetos en cada escena de la animación.

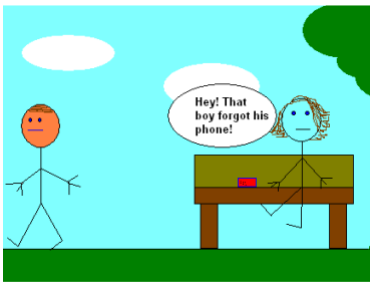
Los guiones gráficos pueden elaborarse con papel y lápiz, o mediante herramientas digitales. Estas últimas pueden ser programas de procesamiento de textos, programas de dibujo o pintura, o presentaciones. También podemos crear guiones gráficos mediante comentarios en el área del editor de códigos de *Alice*. No importa qué método empleemos para crearlos, debemos siempre detallar lo que está sucediendo en cada una de las etapas, de forma que podamos crear el código a partir de ahí.

Guiones gráficos visuales

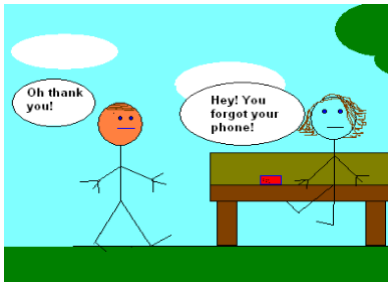
El **guion gráfico visual** ayuda al lector a comprender los componentes de una escena, la configuración de la escena inicial, los objetos en movimiento y estáticos de la escena, las acciones que se llevarán a cabo y las interacciones del usuario que se producirán durante la ejecución de la animación. Ejemplo de un guion gráfico visual:



Chico y chica sentados en un banco del parque. El chico se va, pero deja en el banco su teléfono.



La chica ve el teléfono móvil. Ella piensa: *"¡Hey! ¡Ese chico olvidó su teléfono!!*



La chica dice en voz alta:
"¡Hey! ¡Olvidaste tu teléfono!"
El chico se gira y vuelve al banco. *Él dice:*
"¡Oh! ¡Gracias!".

Guiones gráficos textuales

Un **guion gráfico textual** ayuda al lector a comprender las acciones que se llevarán a cabo durante la animación. Los objetos en movimiento y estáticos se pueden identificar fácilmente dentro de las sentencias de acción, pero puede ser necesaria una descripción más detallada si también participan otros programadores en la implantación de cualquier escena. Para crear un guion gráfico textual de la manera más realista posible, es importante asegurarnos de que hemos simplificado todas las tareas hasta sus componentes individuales. Para la codificación en *Alice*, un guion gráfico que describa la acción de ponerse de pie no es lo suficientemente detallado. Debemos describir las acciones de todas las articulaciones que intervienen en la maniobra de ponerse de pie. En informática, un guion gráfico textual es un algoritmo. Un **algoritmo** es una lista de acciones para llevar a cabo una tarea o resolver un problema.

Los componentes de un guion gráfico textual son:

Componente	Definición	Ejemplos
Escena	Lugar (o mundo en *Alice* 3) donde se desarrolla la historia.	Parque, biblioteca, escuela, hogar
Objetos	Personajes en movimiento o estáticos que se programan para moverse y actuar.	Animales, coches, personas, árboles
Acciones	Instrucciones sobre cómo debe actuar cada objeto en la escena.	Caminar 2 metros, girar a la izquierda, decir: "*Hello*!"
Interacciones del usuario	Formas en que el usuario que esté visualizando la animación puede manipular los objetos de la animación.	Comandos del teclado o clics del *mouse* para hacer que los objetos se muevan
Especificaciones de diseño	Cómo se deben mostrar los objetos y el escenario en la animación.	Tamaño, posición, ubicación, color

Ejemplo 1 de guion gráfico textual:

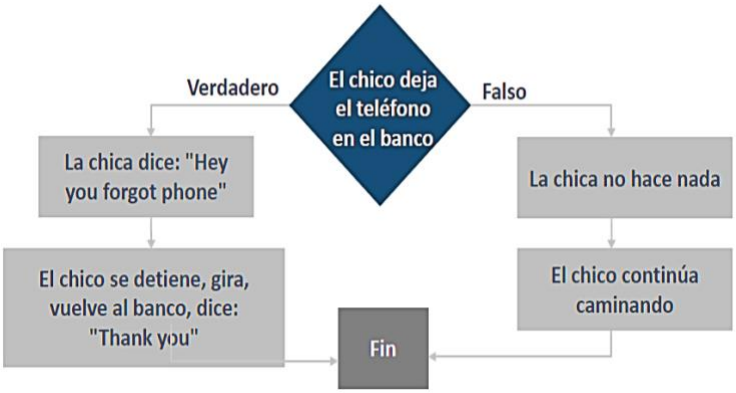
- Programar las siguientes acciones en orden:
  - Chico y chica sentados en un banco del parque.
  - El chico se pone de pie y se aleja, dejando su teléfono celular en el banco del parque.
  - La chica dirige la mirada hacia el teléfono.
  - La chica piensa: "Hey! That boy forgot his phone!"
  - La chica dice en voz alta: "Hey! You forgot your phone!"
  - El chico se detiene y gira.
  - El chico regresa caminando hacia el banco del parque y dice: "Oh! Thank you!".

Ejemplo 2 de guion gráfico textual:

En este ejemplo se muestra cómo puede desarrollar el guion gráfico escribiendo en primer lugar comentarios en el editor de códigos del programa. Posteriormente, puede desarrollar la animación directamente a partir del guion gráfico.

Uso de guiones gráficos para organizar el programa

Los guiones gráficos textuales se pueden utilizar para generar sentencias de comentarios del programa y organizar el desarrollo del programa. La creación de un diagrama de flujo de un guion gráfico nos ayudará a organizar los pasos y orden para la secuencia de la animación. En informática, un guion gráfico textual es un algoritmo.



Un **diagrama de flujo** permite planificar el flujo del código mostrando los procesos y el resultado de cada decisión. La imagen muestra un diagrama sencillo. La complejidad del diagrama de flujo depende del

tamaño del programa. El guion gráfico también puede ayudar a los programadores a identificar acciones repetitivas de un objeto y acciones idénticas que pueden realizar varios objetos.

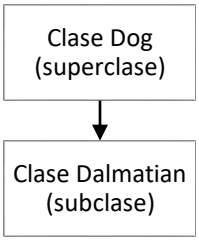
5.3 - Herencia de Clases

Al igual que los animales en el mundo real, los objetos en el mundo de la programación heredan las características de su clase, incluidos todos los métodos (procedimientos y funciones) de la clase. Por ejemplo, todos los objetos de la **clase** de cuadrúpedos de *Alice* heredan las características de los cuadrúpedos de cuatro patas, una cabeza, un cuerpo, etc. Dentro de esta superclase de cuadrúpedos, existen **subclases** para perros, gatos, lobos, leones, vacas, etc. Cada subclase agrega características que identifican más específicamente a los objetos dentro de ella.

Comprender que las subclases pertenecen a superclases nos ayudará, como programadores, a identificar comportamientos repetitivos en el guion gráfico. Por ejemplo, si deseamos programar un perro, un gato y un león caminando, debemos programar el comportamiento repetitivo de caminar en el nivel de superclase, o cuadrúpedo. El resultado es que todas las subclases (perro, gato, león) pueden utilizar la característica hereditaria de caminar y no tenemos que programar el comportamiento repetitivo de caminar para cada uno de los objetos de forma individual.

Características hereditarias

Examinemos cómo un dálmata hereda sus características:



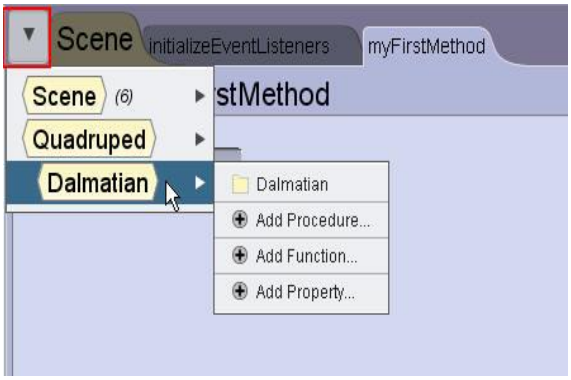
- Las características de la clase *Dog* (clase principal o superclase) incluyen cuatro patas, dos ojos, pelo y la capacidad de ladrar.
- Las características de la clase de raza *Dalmatian* (clase secundaria o subclase, que es un subjuego de la clase *Dog*) incluyen piel blanca, puntos negros y otras características.

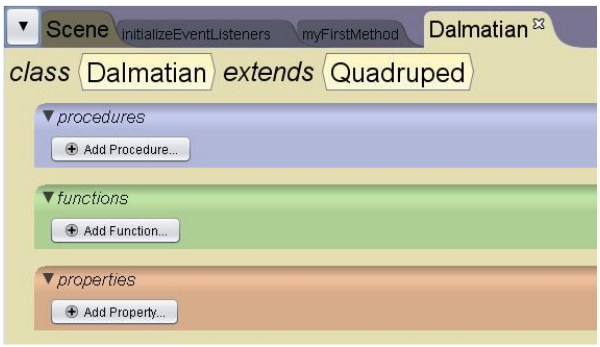
Cuando se crea un objeto *Dalmatian*, hereda los procedimientos, las funciones y las propiedades de la clase de cuadrúpedo y la subclase *Dalmatian* que podemos ver en el editor de códigos. En informática, **herencia** significa que cada objeto de la subclase hereda las propiedades y métodos de su superclase.

Es importante destacar que las subclases pueden heredar las características de su superclase, pero no al contrario. Es un proceso unidireccional. Los diferentes tipos de subclase tendrán acceso al código que existe en la superclase. Cada tipo específico de perro tiene acceso al código que se escribe en la clase *Dog*. Esto significa que todo lo genérico sobre los perros se codifican en el nivel *Dog* mientras que las características que hacen único a un determinado perro se codifican en el nivel de subclase.

Creación de métodos hereditarios

Además de los métodos predefinidos, podemos crear nuestros propios métodos y hacer que se muestren, o que estén disponibles para cualquier objeto de la subclase. Los métodos hereditarios se mostrarán siempre al principio de la lista de métodos predefinidos una vez que se crean. Para crear nuestros propios métodos, utilizaremos el menú desplegable de jerarquías de clase. Este se encuentra a la izquierda de la pestaña *myFirstMethod* (indicado con una flecha que apunta hacia abajo). Allí veremos la lista de clases y subclases de la animación.





En esta lista, podremos seleccionar una superclase o subclase para ver sus procedimientos, funciones y propiedades definidas.

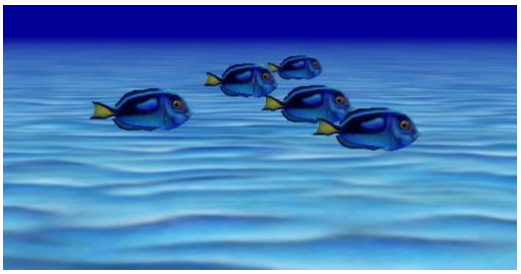
5.4 - Abstracción de Procedimientos

Cuando revisamos el código existente, o el guion gráfico textual, quizás nos encontremos con varias instancias que realizan las mismas acciones. Podemos crear un procedimiento independiente que agrupe y contenga las acciones identificadas. De esta manera reduciremos considerablemente el tamaño del código, lo simplificaremos, y facilitaremos su lectura. También garantizaremos que todos los objetos de una clase, así como sus subclases, se comporten de la misma forma. Este proceso recibe el nombre de abstracción procesal, o abstracción de procesos.

La **abstracción de procesos** se define como:

- búsqueda de procedimientos de programación
- identificar los procedimientos de programación que se repiten
- extraer para crear un nuevo procedimiento.

En la imagen, para la animación de un pez nadando, es necesaria la escritura de varios procedimientos que se deben repetir para cada uno de los peces, que ocupa mucho espacio.



Estas instrucciones podrían agruparse y crear un nuevo procedimiento llamado *swim*.

Otro ejemplo puede ser el de un pájaro que necesita volar, pero no hay ninguna instrucción de programación *fly* disponible para los objetos de la clase pájaro.

Las instrucciones contenidas pueden agregarse como un nuevo procedimiento independiente llamado *fly*.

La abstracción de procesos se puede producir antes o después de crear los procedimientos de programación. Sin embargo, al desarrollar el guion gráfico en primer lugar, el programador puede identificar más fácilmente los procedimientos que serán necesarios antes de que comience la programación. Podemos declarar un procedimiento cuando:

- Los movimientos no tienen un procedimiento por defecto, como un pájaro volando.
- Varios objetos o varias clases necesitan utilizar movimientos como, por ejemplo, los cuadrúpedos para dar saltos hacia arriba y hacia abajo.
- Tenemos movimientos simples, pero que necesitan muchas sentencias de programación, como por ejemplo una persona que mueva las partes del cuerpo al caminar.

Pasos para declarar un procedimiento:

1. Haga un clic en la flecha del botón de jerarquía de clases.
2. Haga clic en la clase o subclase que heredará el procedimiento. Todas las subclases heredarán el procedimiento también. Éstas se insertan con sangrado bajo su superclase.
3. Haga clic en el botón *Add Procedure*.
4. Escriba el nombre para el procedimiento.
5. Haga clic al botón Ok.

Se creará una nueva pestaña con el nombre del procedimiento.

6. Realiza los pasos para declarar un procedimiento de programación según la animación que deseas crear.

Pasos para asignar el nuevo procedimiento a una instancia del objeto:

1. Haga un clic en la pestaña *myFirstMethod*.
2. Haga clic en la instancia a la que se le aplicará el nuevo procedimiento.

En la pestaña de Procedimientos, donde están los procedimientos editables (antes de los procedimientos por defecto) aparece el nuevo procedimiento con el nombre que escribimos.

3. Haga clic en el nuevo procedimiento y arrastra al área del editor de código.

Volvemos al nuevo procedimiento haciendo clic en la pestaña de la parte superior del editor de códigos con el nombre del procedimiento.

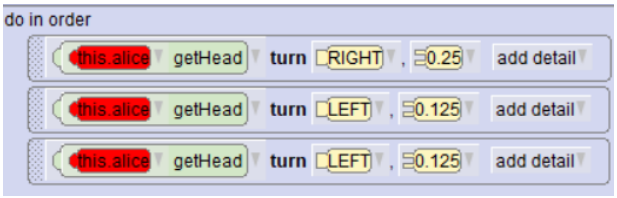
Abstracción de procedimientos y uso del portapapeles

Durante la programación podemos identificar continuamente oportunidades para declarar procedimientos mediante técnicas de abstracción de procedimientos. Después de haber agregado varios procedimientos de instrucción en el espacio de edición en la pestaña *myFirstMethod*, podemos determinar que el procedimiento de programación servirá mejor al programa si estuviera en un procedimiento declarado. En otras palabras, si escribimos el mismo procedimiento de instrucción dos veces o más, probablemente podamos abstraerlo. Para ahorrar tiempo, podemos arrastrar los procedimientos de instrucciones de programación al icono del portapapeles y luego crear el nuevo procedimiento.

Pasos para declarar un nuevo procedimiento mediante abstracción del procedimiento:

1. Una vez haya utilizado dos a más veces un procedimiento de instrucción, haga clic con el botón derecho del mouse en el manejador de instrucción del procedimiento a utilizar.
2. Haga clic en la opción *Copy to Clipboard*. Haga este paso para todos los procedimientos a utilizar en el nuevo procedimiento a declarar.
3. Haga un clic en la flecha del botón de jerarquía de clases.
4. Haga clic en la clase o subclase que heredará el procedimiento.
5. Haga clic en el botón *Add Procedure*.
6. Escriba el nombre para el procedimiento.
7. Haga clic al botón *Ok*.
8. Haga clic en el portapapeles y arrastra con el mouse al espacio de edición. Haga este paso hasta ubicar todos los procedimientos en el espacio de edición.

   Los nombres de las instancias se mostrarán en rojo, ya que al declarar un nuevo procedimiento no puede estar asignado a una instancia en particular.



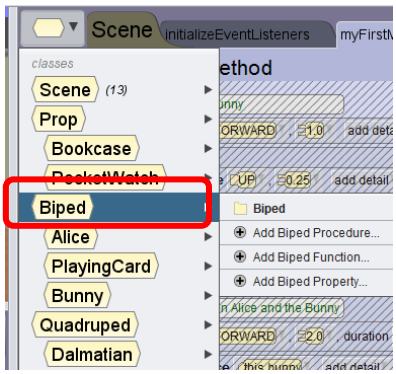
9. Haga clic en la flecha del nombre de la instancia.
10. Haga clic en la opción *this*.

Para asignar el nuevo procedimiento a la instancia, primero tenemos que eliminar los procedimientos de instrucción y luego asignar el nuevo procedimiento a la instancia.

Uso de procedimientos hereditarios

Los procedimientos declarados en el nivel de superclase están disponibles para los demás objetos de esa clase. Por ejemplo, un procedimiento *bipedWave* creado para

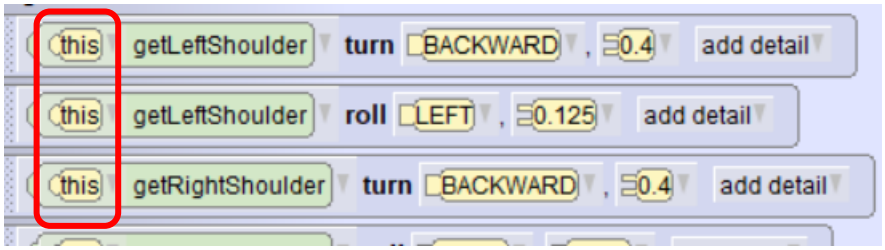
que la instancia *playingCard* salude con la mano (levante las manos en señal de saludo) se podría utilizar para hacer que todos los de una superclase saluden con la mano también.



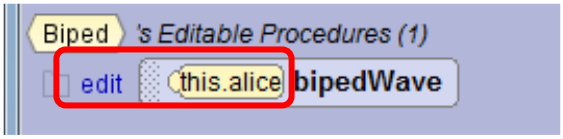
Para declarar un procedimiento en el nivel de superclase, nos aseguramos de seleccionarla correctamente en el botón de jerarquías de clases. De esta forma todas las subclases que se encuentren bajo esta superclase tendrán acceso a ese procedimiento.

Identificador de objeto *this*

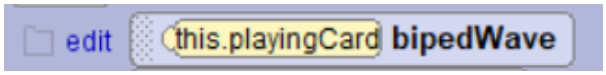
Cuando se crea un procedimiento declarado, el identificador de objeto, ***this***, se utiliza para indicar que la instancia que llama al procedimiento es *this*, o sea para cualquier instancia que le asigne el procedimiento declarado.



Al seleccionar una instancia y agregar el procedimiento declarado (ejemplo; *bipedWave*), a la instancia (ejemplo: instancia *Alice*); *this* hace referencia al nombre de la instancia en el procedimiento declarado.



Si seleccionamos otra instancia (ejemplo; *playingCard*) y le asignamos el mismo procedimiento declarado (procedimiento *bipedWave*), *this* hace referencia ahora al nombre de la otra instancia en el procedimiento declarado.



Básicamente, *this* siempre hace referencia a la instancia de la clase que llama al procedimiento.

Actividad de Avalúo 5: Unidad 5

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Circula la respuesta correcta.

69. En la jerarquía de clases selecciono el dragon, significa que voy a:
    a. Declarar un procedimiento a la clase Biped
    b. Declarar un procedimiento a la subclase dragon
    c. Declarar un procedimiento a la clase cuadrúpedo

70. En informática, un guion gráfico textual es un:
    a. superclase
    b. guion gráfico
    c. algoritmo

71. Significa que cada objeto de la subclase tendrá las propiedades y métodos de su superclase.
    a. herencia
    b. diagrama de flujo
    c. procedimientos

72. Es una lista detallada y ordenada de las acciones que realizan los objetos en cada escena de la animación.
    a. guion gráfico visual
    b. guion gráfico textual
    c. escenario

73. Es el proceso de búsqueda de código de programación, con la identificación de sentencias de programación repetitivas y su extracción para sus propios métodos.
    a. herencia
    b. algoritmo
    c. abstracción de procedimientos

74. Muestra los procesos y el resultado de cada decisión.
    a. sentencias de control
    a. diagrama de flujo
    b. guion gráfico

75. Identifica las especificaciones de diseño del escenario de la animación.
    a. algoritmo
    b. guion gráfico
    c. superclase

76. Identificador de objeto al crear un procedimiento declarado.
    a. this
    b. herencia
    c. procedimiento

77. Ayuda al lector a comprender los componentes de una escena.
    a. guion gráfico visual
    b. guion gráfico textual
    c. escenario

78. Es la idea principal que hay detrás de la animación.
    a. objeto *this*
    b. guion gráfico visual
    c. escenario

Ejercicio 2: Menciona y defina los componentes de un guion gráfico textual.

	Ejemplo	Componentes	Definición
79.	nadar hacia adelante y hacia atrás		
80.	oprimir la tecla Z para que el pez nade		
81.	pez		
82.	cuando el pez se detenga cambie de color		
83.	mar		

UNIDAD 6: ESTRUCTURAS DE PROGRAMACIÓN

Estándar: Programa y desarrollo de aplicaciones

Objetivos: 1. Diseña, desarrolla, prueba e implementar programas y aplicaciones.

6.1 - Argumentos

En las pasadas unidades, vimos que un programa informático necesita **argumentos** para indicar cómo se debe implantar el procedimiento. Los argumentos de un procedimiento se pueden editar o definir con mayor detalle para controlar el movimiento y la temporización del objeto. Los ejemplos de argumentos de Alice 3 incluyen:

- Privilegios
- Dirección
- Cantidad de dirección
- Tiempo de duración

Al agregar un procedimiento de programación, se nos pedirá que seleccionemos un valor para cada argumento del procedimiento. Normalmente, se puede seleccionar un valor predefinido como un **marcador de posición** (un valor temporal) que se puede cambiar posteriormente durante un ciclo de edición. Utilizar un valor de marcador de posición suele ser un enfoque común para crear y mejorar el rendimiento de la animación.

6.2 - Movimientos Simultáneos

Cuando vamos al Editor de Código en la pestaña *myFirstMethod*, encontraremos por defecto la sentencia de control *do in order* ingresada. Ésta permite ejecutar sentencias de programación orden secuencial. Sin embargo, hay ocasiones en que deseamos crear movimientos simultáneos para una instancia. Para esto, emplearemos la sentencia de control *do together*. Un movimiento ejecutado conjuntamente podría ser tan simple como levantar a la vez ambos brazos de una instancia bípedo desde una posición suspendida hasta colocar el brazo recto sobre la cabeza.

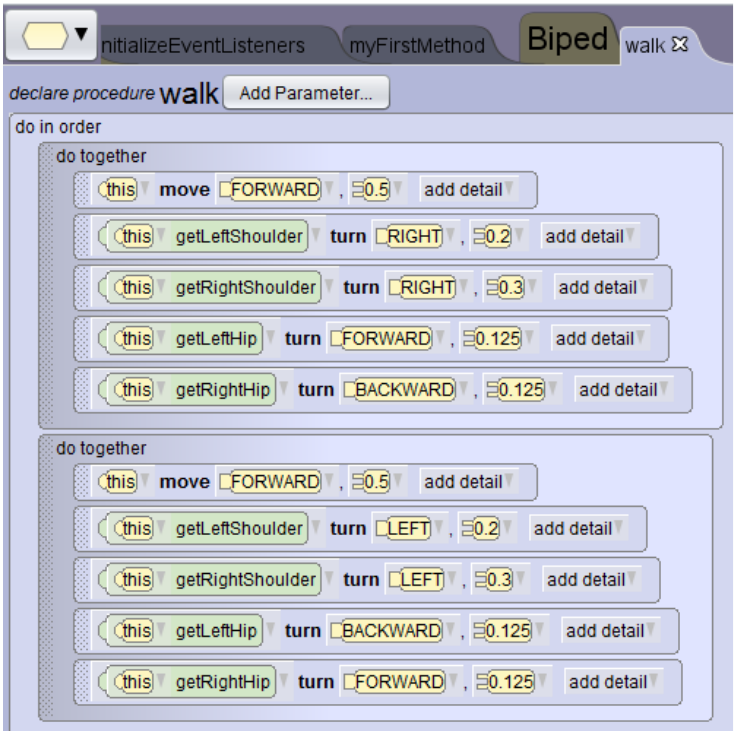


Otro ejemplo es el movimiento de caminar, para el que se necesita el movimiento simultáneo de las caderas y los hombros. Para crear el movimiento de caminar de un bípedo, utilizamos una serie de procedimientos *move*, *roll* y *turn*, y sentencias de control *do together*. La programación puede ser diferente para los diferentes objetos, porque no hay dos objetos que caminen del mismo modo. Veamos estas instrucciones incluidas en un guion gráfico textual:

Ejemplo de guión gráfico textual del movimiento de caminar

Orden de las instrucciones	Instrucciones de programación
Do Together	Todo el cuerpo se desplaza hacia delante
	El hombro izquierdo gira hacia la derecha
	El hombro derecho gira hacia la derecha
	La cadera izquierda gira hacia delante
	La cadera derecha gira hacia atrás
Do Together	Todo el cuerpo se desplaza hacia delante
	El hombro izquierdo gira hacia la izquierda
	El hombro derecho gira hacia la izquierda
	La cadera izquierda gira hacia atrás
	La cadera derecha gira hacia delante

Veamos ahora cómo se trasladarían estas instrucciones a un código que ejecute el movimiento de caminar en *Alice*.



Anulación de procedimientos entre sí

Un error común es incluir dos procedimientos que se cancelan entre sí en una sentencia *do together.* Por ejemplo, si incluye un movimiento para moverte hacia arriba a 1 metro y moverte hacia abajo a 1 metro, en una sentencia *do together*, no ocurrirá nada. Los procedimientos se cancelan entre sí, ya que no podemos movernos hacia arriba y hacia abajo a la misma vez.

6.3 - Procedimiento *SetVehicle*

El procedimiento *setVehicle* emplea el concepto de una instancia de pasajero y una instancia de vehículo. La instancia de pasajero se selecciona cuando se utiliza el procedimiento *setVehicle* para especificar el vehículo del pasajero. Cuando la instancia de vehículo esté programada para moverse, la instancia de pasajero se moverá automáticamente al mismo tiempo. Por ejemplo, este procedimiento puede emplearse cuando programamos a una persona que monta en camello o a caballo, o a una cámara que sigue a un helicóptero para grabar la escena desde el punto de vista del helicóptero.

Veamos un ejemplo del procedimiento *setVehicle*.

La persona se coloca sobre el camello (en el editor de escena). Se realiza los pasos para definir (identificar) el camello como el vehículo de la persona, la persona será el pasajero. Cuando el camello se mueva, la persona permanece en la parte superior y se mueve con el camello.

Veamos un segundo ejemplo del procedimiento *setVehicle*.

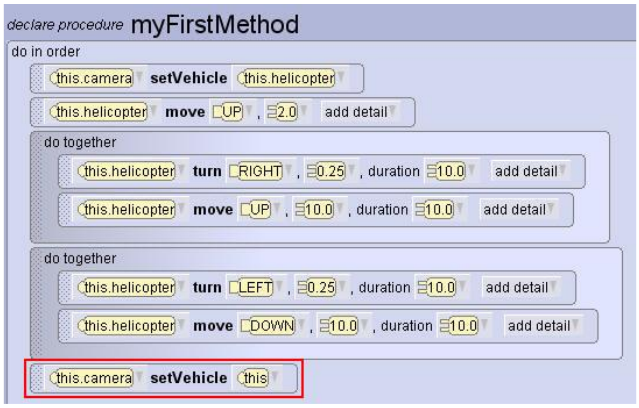
El helicóptero irá filmando por el terreno. Se realiza los pasos para definir el helicóptero como el vehículo de la cámara; la cámara será el pasajero. Cuando el helicóptero se mueva, la cámara se mueve con el helicóptero y hace la animación de estar filmando desde la perspectiva del helicóptero.

Pasos para asignar el procedimiento setVehicle:

1. Determina qué instancia será el vehículo y qué instancia será el pasajero.
2. Haga clic en la instancia que será el pasajero.
3. En la pestaña *Procedures*, haga clic en la instrucción de programación *setVehicle* y arrastra con el mouse al espacio de edición.
4. Suelta el botón del mouse.
5. Haga clic en la instancia que será el vehículo.
6. Siga los pasos para crear un procedimiento de instrucción de programación.

Detener el procedimiento setVehicle

Si desea que la instancia definida pasajero no esté ya vinculada con la instancia definida vehículo, o sea que se puedan mover individualmente, arrastra otro procedimiento *setVehicle* al área del editor de códigos y defina el objeto de vehículo en *this*,



UNIDAD 7: FUNCIONES

Estándar: Programa y desarrollo de aplicaciones

Objetivos: 1. Diseña, desarrolla, prueba e implementar programas y aplicaciones.

Las **funciones** responden preguntas sobre la instancia, como su altura, ancho, profundidad e incluso su distancia de otra instancia. En otras palabras, podemos decir que las funciones se utilizan para realizar preguntas acerca de las propiedades de un objeto. Son similares a los procedimientos salvo que devuelven un valor de un tipo determinado. Las funciones proporcionan respuestas precisas a preguntas, tales como:

- ¿Cuál es la distancia entre las instancias de *Dalmatian* y *bunny*?
- ¿Cuál es la altura de *playingCard*?
- ¿Cuál es el ancho de *pocketWatch*?

Las funciones pueden ser utilizadas como argumentos los cuales se van añadiendo a cada uno de los procedimientos de programación y que hacen referencia al movimiento, la dirección o velocidad del objeto. Una **función booleana** devuelve un valor *true* o *false*. Por ejemplo, si se llama la función *isFacing* para determinar si el objeto *Alice* está orientado hacia el objeto *bunny*, se devolverá un valor *true* o *false*.

Las funciones se encuentran en el Panel de Métodos, en la pestaña *Functions*. La pestaña ***Functions*** contiene todas las funciones generales que hereda un objeto, así como el acceso a las articulaciones específicas disponibles únicamente para esa clase. El uso de funciones para crear animaciones dinámicas que se ejecutan de forma diferente en cada ejecución es una de las principales ventajas de *Alice*.

7.1 - Pestaña *Functions*

Las funciones nos van a permitir medir algo, controlar el movimiento. Siempre se van a utilizar para hacer preguntas acerca de las propiedades de una instancia. Son similares a los procedimientos salvo que devuelven un valor de un tipo determinado. Las funciones responden preguntas sobre una instancia como su altura, ancho, profundidad e incluso su distancia de otra instancia. Las funciones son importantes, ya que pueden proporcionarle información sobre la instancia en cualquier punto de la animación. La información se puede utilizar para realizar animaciones precisas.

Pasos asignar función:

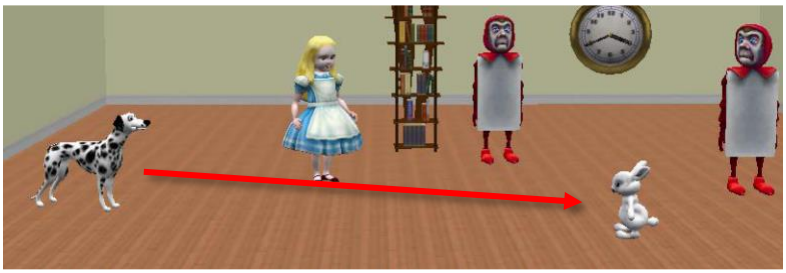
1. Realiza los pasos para declarar un procedimiento de instrucción.
2. Haga clic en la pestaña *Functions*.
3. Haga clic en mosaico deseado y arrastra hasta el valor del argumento.
   Alice ayuda a visualizar las ubicaciones en que se puede colocar las funciones al oscurecer la pantalla (color violeta) y resaltar los valores que se pueden sustituir por la función utilizada

4. Prueba y depura.

7.2 - Uso de la Función *GetDistanceTo*

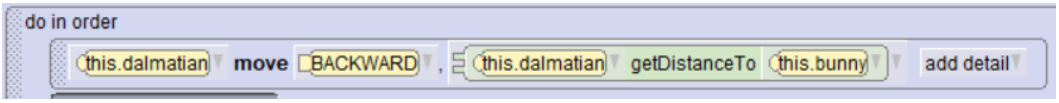
Supongamos que deseamos mover la instancia *Dalmatian* directamente hacia la instancia *Bunny* sin tener que determinar manualmente, mediante un proceso de prueba y error, la distancia entre ambas instancias. Podemos averiguar la distancia si especificamos un valor de marcador de posición y probamos el movimiento hasta que nos acerquemos al resultado final deseado, pero una forma más eficiente es utilizar una función para determinar la distancia exacta que debe recorrer el

movimiento. La función *getDistanceTo* como parte de un procedimiento *move* puede ayudarnos a solucionar este problema de distancia.



Pasos asignar la función getDistanceTo:

5. Determina la instancia en movimiento y la instancia de destino.
6. Haga clic en la instancia en movimiento.
7. En la pestaña *Procedures*, haga clic a la instrucción de programación *move* y arrastra al espacio de edición.
8. Haga clic en los valores de los argumentos deseados.
9. Haga clic en la pestaña *Functions*.
10. Haga clic en mosaico *getDistanceTo* y arrastra hasta el valor de la distancia.
11. Haga clic en el objeto de destino.



Prueba de la función

En el ejemplo anterior, con la función *getDistanceTo* se puede leer como “determinar la distancia desde el centro de la instancia *Dalmatian* hasta el centro de la instancia *Bunny* y, a continuación, mover la instancia *Dalmatian* hacia delante la distancia especificada".

Podemos hacer clic en el botón *Run* para probar el procedimiento de programación. Veremos que el objeto *Dalmatian* se desplaza hasta el centro del objeto *Bunny*. Esto se debe a que la función *getDistanceTo* calcula la distancia entre los centros de ambas instancias. La función calcula la distancia desde el centro de la instancia en movimiento hasta el centro de la instancia destino.

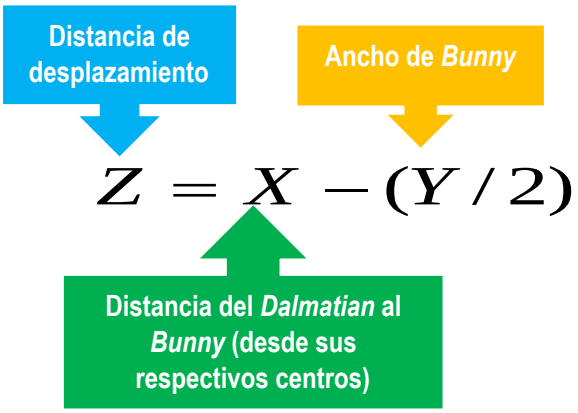
Evitar colisiones

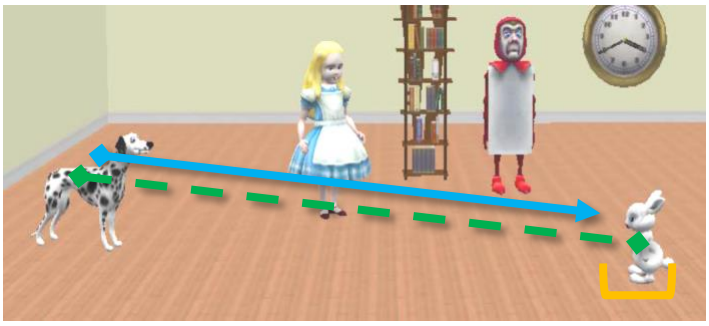
A veces, el uso de funciones tiene resultados imprevistos. La distancia entre las instancias es calculada desde su centro por esa razón pueden chocar, pues no se

midió la profundidad o longitud. Veamos cómo podemos utilizar las funciones para solucionar este problema.

7.3 - Operadores Matemáticos

Podemos mejorar las llamadas de funciones mediante los operadores matemáticos (+) suma, (-) resta, (*) multiplicación y (/) división. Por ejemplo, podemos reducir la distancia de movimiento de una instancia para evitar una colisión. La función *getDistance* determina la distancia entre las instancias *Dalmatian* y *Bunny*. Para reducir el valor devuelto por la función, utilizamos el operador de resta para sustraerle un valor específico. Este valor específico se obtiene al llamar a la función g*etWidth* y dividiendo el valor obtenido por la mitad.



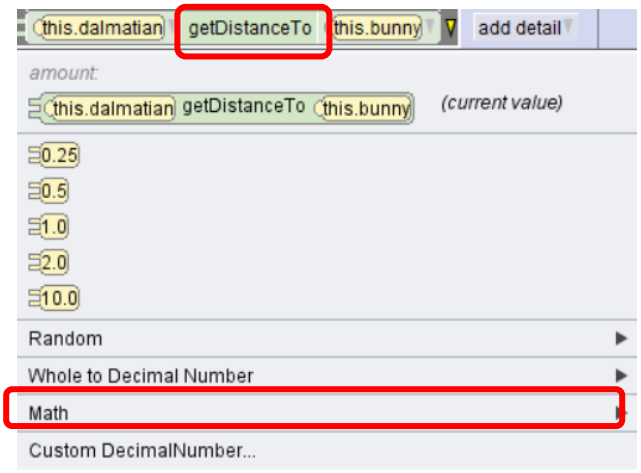


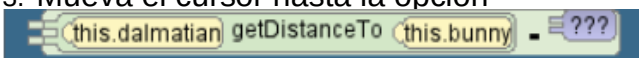
Examinemos el cálculo matemático Z = X -(Y/2):

- **Z** representa la distancia total que se desplazará la instancia *Dalmatian*.
- **X** representa la distancia entre las instancias de *Dalmatian* y *Bunny*.
- **Y** representa el ancho de *Bunny*.
- **Y / 2** representa el ancho de *Bunny* dividido por 2.
- **( )** representan el orden de prioridad.

¿Por qué dividir el ancho del objeto *Bunny* en nuestro cálculo? Porque deseamos que en la animación de la instancia *Dalmatian* se mueva hasta el mismo borde de la instancia *Bunny*. Si se utiliza todo el ancho de *Bunny*, la instancia *Dalmatian* se detendrá más lejos de la instancia *Bunny* de lo deseado.

Pasos para utilizar operadores matemáticos para evitar la colisión:



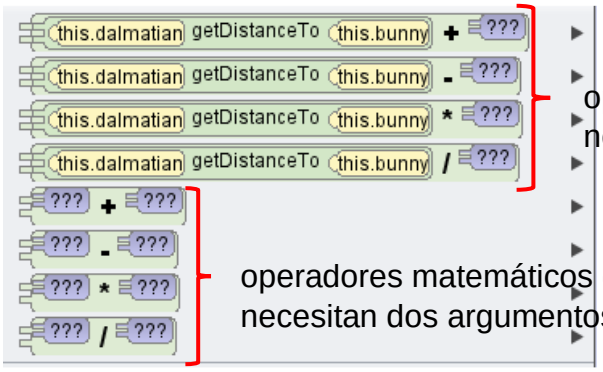
1. Haga clic en el nombre de la función entre las dos instancias.
2. Coloca el cursor en la opción *Math* de la lista desplegable.
3. Mueva el cursor hasta la opción 

4. Haga clic en el valor fijo deseado.
5. Probar y depurar.

Podemos ajustar el valor fijo para mejores resultados.

Descripción de los menús matemáticos

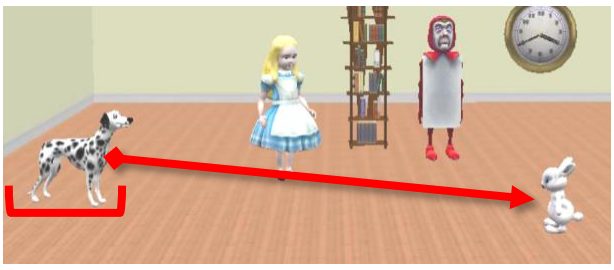
La siguiente imagen muestra los operadores matemáticos (+ -* /) que necesitan uno o dos argumentos. Cada opción proporcionará uno o dos menús en cascada para especificar los valores de los argumentos.



Recordamos que podemos seleccionar valores de marcador de posición para los argumentos. Los valores de marcador de posición se pueden editar siempre.

Eliminación de la profundidad del objeto de la función

Otra forma precisa de evitar colisiones es eliminar la profundidad (longitud) de la instancia en movimiento de la función. En la imagen a la derecha, la instancia *Dalmatian* recorrerá la distancia hasta la instancia *Bunny*, menos la profundidad de *Dalmatian*.



Cuando se calcula un valor de distancia, se mide desde un centro de la instancia hasta el centro de la otra instancia. Lo mismo ocurre con los cálculos matemáticos. Al restar la profundidad de la instancia *Dalmatian* al objeto *Bunny*, en realidad se resta del centro de la instancia *Bunny*.

Pasos para eliminar la profundidad de la función:

1. Haga clic en el mosaico *getDepth* y arrastra hasta el valor de distancia resaltado.

2. Probar y depurar.

Realizamos un ajuste con cálculos matemáticos adicionales si es necesario.

Actividad de Avalúo 6: Unidad 6 y 7

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Cierto o Falso. Si la premisa es cierta, escriba la **C**. Si la premisa es falsa, escriba la **F** y sustituya la(s) palabra(s) subrayada(s) para convertirla en cierta.

Ejemplo: En la pestaña jerarquía de clases se encuentra el Editor de Código.
Respuesta: F, myFirstMethod

79. La pestaña Functions contiene todas las funciones generales que hereda un objeto.
80. La función getWith nos ayuda a evitar una colisión al restar la profundidad de las dos instancias.
81. El procedimiento move es parte de la función getDistanceTo.
82. La instrucción de programación setVehicle se encuentra en la pestaña de procedimientos.
83. En el panel myFirstMethod encontramos por defecto la sentencia do together.
84. Una función count devuelve dos valores.
85. Las funciones pueden ser utilizadas como instrucciones de programación que se van añadiendo a cada uno de los procedimientos de programación.
86. El procedimiento setVehicle emplea el concepto de una instancia de pasajero y una instancia de vehículo
87. Un marcador de procedimiento es un valor temporal que se puede cambiar posteriormente durante un ciclo de edición.
88. Las funciones responden preguntas relacionadas a la altura, ancho y profundidad de la instancia.
89. Para desvincular la instancia de pasajero con la instancia de vehículo en un procedimiento setVehicle, se utiliza el mismo procedimiento y se define el objeto de vehículo a el pasajero
90. La sentencia do in order permite ejecutar procedimientos secuenciales.
91. Para evitar que los objetos colisionen podemos reducir la distancia de movimiento de un objeto.
92. Los manejadores de instrucción, en un programa informático, se necesitan para indicar cómo se debe implantar el procedimiento.
93. Un movimiento simultáneo se logra con la sentencia de control if/else.

CLAVES DE ACTIVIDADES DE AVALÚO

Clave Actividad de Avalúo 1: Unidad 1

Ejercicio 1: Selección Múltiple

1. **b**
2. **a**
3. **a**
4. **a**, **b**
5. **c**
6. **a**, **b**, **c**
7. **a**, **c**
8. **c**
9. **b**
10. **c**
11. **c**
12. **a**

Ejercicio 2: Identifica Múltiple

13. **menú de instancia, aparecen todas las instancias del objeto que están en la escena, muestra la instancia del objeto seleccionada en la escena y podemos seleccionar la instancia del objeto**
14. **panel de propiedades, atributos para aplicar a la instancia del objeto**
15. **escenario**
16. **sentencias de control**
17. **botón Run, ejecutar la animación**
18. **espacio de edición**
19. **botón Edit Code, cambiar al editor de código**
20. **panel de métodos**
21. **galería, objetos tridimensionales para agregar al escenario**
22. **botón de Setup Scene, cambiar al editor de escena**

Clave Actividad de Avalúo 2: Unidad 2

Ejercicio 1: Selección Múltiple

23. **c**
24. **e**
25. **f**
26. **d**
27. **g**
28. **b**
29. **f**
30. **h**
31. **a**
32. **d**

Ejercicio 2: Ordena

33. **b**, **a**, **c**
34. **d**, **c**, **a**, **b**
35. **a**, **d**, **c**, **b**
36. **b**, **d**, **a**, **c**
37. **c**, **b**, **a**
38. **e**, **a**, **b**, **d**, **c**, **f**

Clave Actividad de Avalúo 3: Unidad 3

Ejercicio 1: Selección Múltiple

39. **a**
40. **a**
41. **c**
42. **b**
43. **a**
44. **c**
45. **c**
46. **a**
47. **b**
48. **c**

Ejercicio 2: Menciona

49. **Custom Decimal Number**
50. El estudiante puede escribir tres de cualquiera de estos procedimientos:

    Move - la instancia se desplaza en cualquiera de sus seis direcciones.

    Move Toward - la instancia se desplaza hacia otra instancia.

    Move Away From - la instancia se aleja de otra instancia.

    Move To - la instancia se desplaza de su posición actual al punto central de la instancia de destino.

    Move and Orient To - la instancia se desplaza de su posición actual al punto central de la instancia de destino y ajusta la orientación de la instancia para que coincida con la orientación de la instancia de destino.

    Delay - Se detiene el movimiento de una instancia durante un determinado número de segundos.

    Say - crea una burbuja de llamada con texto que hace que parezca que el objeto habla.
51. **manejador de instrucción**
52. **Is Enabled**
53. **Panel de Métodos**

Clave Actividad de Avalúo 4: Unidad 4

Ejercicio 1: Llena blanco

54. **if/else**
55. **roll**
56. **do together**
57. **1.0**
58. **anidación**
59. **sentencias de control**
60. **números aleatorios**
61. **turn**
62. **count**
63. **sub-partes**
64. **egocéntrico**
65. **guion gráfico textual (storyboard)**
66. **do in order**
67. **horizontal**
68. **while**

Clave Actividad de Avalúo 5: Unidad 5

Ejercicio 1: Selección Múltiple

69. **b**
70. **c**
71. **a**
72. **b**
73. **c**
74. **b**
75. **b**
76. **a**
77. **a**
78. **c**

Ejercicio 2: Menciona y Defina

79. **Acciones - Instrucciones sobre cómo debe actuar cada objeto en la escena**.
80. **Interacciones del usuario - Formas en que el usuario que este visualizando la animación puede manipular los objetos de la animación.**
81. **Especificaciones del diseño - Como se deben mostrar los objetos y el escenario en la animación.**
82. **Escena - Lugar (o mundo en Alice 3) donde se desarrolla la historia.**
83. **Objetos - Personajes en movimiento o estáticos que se programan para moverse y actuar.**

Clave Actividad de Avalúo 6: Unidad 6 y 7

Ejercicio 1: Cierto o Falso

79. **C**
80. **F**, **getDepth**
81. **C**
82. **C**
83. **F**, **do in order**
84. **F**, **booleana**
85. **F**, **argumentos**
86. **C**
87. **F**, **marcador de posición**
88. **C**
89. **F**, **this**
90. **C**
91. **C**
92. **F**, **argumemtos**
93. **F**, **do together**

REFERENCIA

About Alice. (2017). Recuperado el 2 de junio de 2020 de https://www.*Alice*.org/

Conozca más sobre la tecnología Java. (s.f.). Recuperado el 2 de junio de 2020 de https://www.java.com/es/about/ .

Codecademy. (2020). *Learn Java.* Recuperado el 2 de junio de 2020 de https://www.codecademy.com/learn/learn-java

National Business Education Association. (2013). *Information Technology*. Recuperado de https://neapolitanlabs.education/assets/north_linn/files/business_education.pdf

Oracle Academy.(2019). *Conceptos fundamentales de Java.* Recuperado el 2 de junio de 2020 de https://www.ilearning.oracle.com

Estimada familia:

El Departamento de Educación de Puerto Rico (DEPR) tiene como prioridad el garantizar que a sus hijos se les provea una educación pública, gratuita y apropiada. Para lograr este cometido, es imperativo tener presente que los seres humanos son diversos. Por eso, al educar es necesario reconocer las habilidades de cada individuo y buscar estrategias para minimizar todas aquellas barreras que pudieran limitar el acceso a su educación.

La otorgación de acomodos razonables es una de las estrategias que se utilizan para minimizar las necesidades que pudiera presentar un estudiante. Estos permiten adaptar la forma en que se presenta el material, la forma en que el estudiante responde, la adaptación del ambiente y lugar de estudio y el tiempo e itinerario que se utiliza. Su función principal es proveerle al estudiante acceso equitativo durante la enseñanza y la evaluación. Estos tienen la intención de reducir los efectos de la discapacidad, excepcionalidad o limitación del idioma y no, de reducir las expectativas para el aprendizaje. Durante el proceso de enseñanza y aprendizaje, se debe tener altas expectativas con nuestros niños y jóvenes.

Esta guía tiene el objetivo de apoyar a las familias en la selección y administración de los acomodos razonables durante el proceso de enseñanza y evaluación para los estudiantes que utilizarán este módulo didáctico. Los acomodos razonables le permiten a su hijo realizar la tarea y la evaluación, no de una forma más fácil, sino de una forma que sea posible de realizar, según las capacidades que muestre. El ofrecimiento de acomodos razonables está atado a la forma en que su hijo aprende. Los estudios en neurociencia establecen que los seres humanos aprenden de forma visual, de forma auditiva o de forma kinestésica o multisensorial, y aunque puede inclinarse por algún estilo, la mayoría utilizan los tres.

Por ello, a continuación, se presentan algunos ejemplos de acomodos razonables que podrían utilizar con su hijo mientras trabaja este módulo didáctico en el hogar. Es importante que como madre, padre o persona encargada en dirigir al estudiante en esta tarea los tenga presente y pueda documentar cuales se utilizaron. Si necesita más información, puede hacer referencia a la ***Guía para la provisión de acomodos razonables*** (2018) disponible por medio de la página www.de.pr.gov, en educación especial, bajo Manuales y Reglamentos.

GUÍA DE ACOMODOS RAZONABLES PARA LOS ESTUDIANTES QUE TRABAJARÁN BAJO MÓDULOS DIDÁCTICOS

Acomodos de presentación	Acomodos en la forma de responder	Acomodos de ambiente y lugar	Acomodos de tiempo e itinerario
Cambian la manera en que se presenta la información al estudiante. Esto le permite tener acceso a la información de diferentes maneras. El material puede ser presentado de forma auditiva, táctil, visual o multisensorial.	Cambian la manera en que el estudiante responde o demuestra su conocimiento. Permite a los estudiantes presentar las contestaciones de las tareas de diferentes maneras. Por ejemplo, de forma verbal, por medio de manipulativos, entre otros.	Cambia el lugar, el entorno o el ambiente donde el estudiante completará el módulo didáctico. Los acomodos de ambiente y lugar requieren de organizar el espacio donde el estudiante trabajará.	Cambian la cantidad de tiempo permitido para completar una evaluación o asignación; cambia la manera, orden u hora en que se organiza el tiempo, las materias o las tareas.
Aprendiz visual: ▪ Usar letra agrandada o equipos para agrandar como lupas, televisores y computadoras ▪ Uso de láminas, videos pictogramas. ▪ Utilizar claves visuales tales como uso de colores en las instrucciones, resaltadores (highlighters), subrayar palabras importantes. ▪ Demostrar lo que se espera que realice el estudiante y utilizar modelos o demostraciones. ▪ Hablar con claridad, pausado ▪ Identificar compañeros que	**Aprendiz visual:** ▪ Utilizar la computadora para que pueda escribir. ▪ Utilizar organizadores gráficos. ▪ Hacer dibujos que expliquen su contestación. ▪ Permitir el uso de láminas o dibujos para explicar sus contestaciones ▪ Permitir que el estudiante escriba lo que aprendió por medio de tarjetas, franjas, láminas, la computadora o un comunicador visual. ▪ Contestar en el folleto. **Aprendiz auditivo**:	**Aprendiz visual:** ▪ Ambiente silencioso, estructurado, sin muchos distractores. ▪ Lugar ventilado, con buena iluminación. ▪ Utilizar escritorio o mesa cerca del adulto para que lo dirija. **Aprendiz auditivo**: ▪ Ambiente donde pueda leer en voz alta o donde pueda escuchar el material sin interrumpir a otras personas. ▪ Lugar ventilado, con buena iluminación y donde se les permita el movimiento mientras repite	**Aprendiz visual y auditivo:** ▪ Preparar una agenda detalladas y con códigos de colores con lo que tienen que realizar. ▪ Reforzar el que termine las tareas asignadas en la agenda. ▪ Utilizar agendas de papel donde pueda marcar, escribir, colorear. ▪ Utilizar “post-it” para organizar su día. ▪ Comenzar con las clases más complejas y luego moverse a las sencillas. ▪ Brindar tiempo extendido para completar sus tareas.

Acomodos de presentación	Acomodos en la forma de responder	Acomodos de ambiente y lugar	Acomodos de tiempo e itinerario
puedan servir de apoyo para el estudiante ▪ Añadir al material información complementaria **Aprendiz auditivo**: ▪ Leerle el material o utilizar aplicaciones que convierten el texto en formato audible. ▪ Leer en voz alta las instrucciones. ▪ Permitir que el estudiante se grabe mientras lee el material. ▪ Audiolibros ▪ Repetición de instrucciones ▪ Pedirle al estudiante que explique en sus propias palabras lo que tiene que hacer ▪ Utilizar el material grabado ▪ Identificar compañeros que puedan servir de apoyo para el estudiante **Aprendiz multisensorial:** ▪ Presentar el material segmentado (en pedazos) ▪ Dividir la tarea en partes cortas ▪ Utilizar manipulativos ▪ Utilizar canciones ▪ Utilizar videos	▪ Grabar sus contestaciones ▪ Ofrecer sus contestaciones a un adulto que documentará por escrito lo mencionado. ▪ Hacer presentaciones orales. ▪ Hacer videos explicativos. ▪ Hacer exposiciones **Aprendiz multisensorial:** ▪ Señalar la contestación a una computadora o a una persona. ▪ Utilizar manipulativos para representar su contestación. ▪ Hacer presentaciones orales y escritas. ▪ Hacer dramas donde represente lo aprendido. ▪ Crear videos, canciones, carteles, infografías para explicar el material. ▪ Utilizar un comunicador electrónico o manual.	en voz alta el material. **Aprendiz multisensorial:** ▪ Ambiente se le permita moverse, hablar, escuchar música mientras trabaja, cantar. ▪ Permitir que realice las actividades en diferentes escenarios controlados por el adulto. Ejemplo el piso, la mesa del comedor y luego, un escritorio.	**Aprendiz multisensorial:** ▪ Asistir al estudiante a organizar su trabajo con agendas escritas o electrónicas. ▪ Establecer mecanismos para recordatorios que le sean efectivos. ▪ Utilizar las recompensas al terminar sus tareas asignadas en el tiempo establecido. ▪ Establecer horarios flexibles para completar las tareas. ▪ Proveer recesos entre tareas. ▪ Tener flexibilidad en cuando al mejor horario para completar las tareas. ▪ Comenzar con las tareas más fáciles y luego, pasar a las más complejas. ▪ Brindar tiempo extendido para completar sus tareas.

Acomodos de presentación	Acomodos en la forma de responder	Acomodos de ambiente y lugar	Acomodos de tiempo e itinerario
▪ Presentar el material de forma activa, con materiales comunes. ▪ Permitirle al estudiante investigar sobre el tema que se trabajará ▪ Identificar compañeros que puedan servir de apoyo para el estudiante			

HOJA DE DOCUMENTAR LOS ACOMODOS RAZONABLES UTILIZADOS AL TRABAJAR EL MÓDULO DIDÁCTICO

Nombre del estudiante: _____ **Número de SIE:**_____
Materia del módulo: _____ **Grado:**_____

Estimada familia:

1. Utiliza la siguiente hoja para documentar los acomodos razonables que utiliza con tu hijo en el proceso de apoyo y seguimiento al estudio de este módulo. Favor de colocar una marca de cotejo [✓] en aquellos acomodos razonables que utilizó con su hijo para completar el módulo didáctico. Puede marcar todos los que aplique y añadir adicionales en la parte asignada para ello.

Acomodos de presentación	Acomodos de tiempo e itinerario
Aprendiz visual: ☐ Usar letra agrandada o equipos para agrandar como lupas, televisores y computadoras ☐ Uso de láminas, videos pictogramas. ☐ Utilizar claves visuales tales como uso de colores en las instrucciones, resaltadores (*highlighters*), subrayar palabras importantes. ☐ Demostrar lo que se espera que realice el estudiante y utilizar modelos o demostraciones. ☐ Hablar con claridad, pausado ☐ Identificar compañeros que puedan servir de apoyo para el estudiante ☐ Añadir al material información complementaria **Aprendiz auditivo**: ☐ Leerle el material o utilizar aplicaciones que convierten el texto en formato audible. ☐ Leer en voz alta las instrucciones. ☐ Permitir que el estudiante se grabe mientras lee el material. ☐ Audiolibros ☐ Repetición de instrucciones ☐ Pedirle al estudiante que explique en sus propias palabras lo que tiene que hacer ☐ Utilizar el material grabado ☐ Identificar compañeros que puedan servir de apoyo para el estudiante **Aprendiz multisensorial:** ☐ Presentar el material segmentado (en pedazos) ☐ Dividir la tarea en partes cortas	**Aprendiz visual:** ☐ Utilizar la computadora para que pueda escribir. ☐ Utilizar organizadores gráficos. ☐ Hacer dibujos que expliquen su contestación. ☐ Permitir el uso de láminas o dibujos para explicar sus contestaciones ☐ Permitir que el estudiante escriba lo que aprendió por medio de tarjetas, franjas, láminas, la computadora o un comunicador visual. ☐ Contestar en el folleto. **Aprendiz auditivo**: ☐ Grabar sus contestaciones ☐ Ofrecer sus contestaciones a un adulto que documentará por escrito lo mencionado. ☐ Hacer presentaciones orales. ☐ Hacer videos explicativos. ☐ Hacer exposiciones **Aprendiz multisensorial:** ☐ Señalar la contestación a una computadora o a una persona. ☐ Utilizar manipulativos para representar su contestación. ☐ Hacer presentaciones orales y escritas. ☐ Hacer dramas donde represente lo aprendido. ☐ Crear videos, canciones, carteles, infografías para explicar el material. ☐ Utilizar un comunicador electrónico o manual.

Acomodos de presentación	Acomodos de tiempo e itinerario
☐ Utilizar manipulativos ☐ Utilizar canciones ☐ Utilizar videos ☐ Presentar el material de forma activa, con materiales comunes. ☐ Permitirle al estudiante investigar sobre el tema que se trabajará ☐ Identificar compañeros que puedan servir de apoyo para el estudiante	
Acomodos de respuesta	**Acomodos de ambiente y lugar**
Aprendiz visual: ☐ Ambiente silencioso, estructurado, sin muchos distractores. ☐ Lugar ventilado, con buena iluminación. ☐ Utilizar escritorio o mesa cerca del adulto para que lo dirija. **Aprendiz auditivo**: ☐ Ambiente donde pueda leer en voz alta o donde pueda escuchar el material sin interrumpir a otras personas. ☐ Lugar ventilado, con buena iluminación y donde se les permita el movimiento mientras repite en voz alta el material. **Aprendiz multisensorial:** ☐ Ambiente se le permita moverse, hablar, escuchar música mientras trabaja, cantar. ☐ Permitir que realice las actividades en diferentes escenarios controlados por el adulto. Ejemplo el piso, la mesa del comedor y luego, un escritorio.	**Aprendiz visual y auditivo:** ☐ Preparar una agenda detalladas y con códigos de colores con lo que tienen que realizar. ☐ Reforzar el que termine las tareas asignadas en la agenda. ☐ Utilizar agendas de papel donde pueda marcar, escribir, colorear. ☐ Utilizar “post-it” para organizar su día. ☐ Comenzar con las clases más complejas y luego moverse a las sencillas. ☐ Brindar tiempo extendido para completar sus tareas. **Aprendiz multisensorial:** ☐ Asistir al estudiante a organizar su trabajo con agendas escritas o electrónicas. ☐ Establecer mecanismos para recordatorios que le sean efectivos. ☐ Utilizar las recompensas al terminar sus tareas asignadas en el tiempo establecido. ☐ Establecer horarios flexibles para completar las tareas. ☐ Proveer recesos entre tareas. ☐ Tener flexibilidad en cuando al mejor horario para completar las tareas. ☐ Comenzar con las tareas más fáciles y luego, pasar a las más complejas. ☐ Brindar tiempo extendido para completar sus tareas.
Otros: _____ _____ _____ _____ _____ _____ _____ _____ _____	

2. Si tu hijo es un candidato o un participante de los servicios para estudiantes aprendices del español como segundo idioma e inmigrantes considera las siguientes sugerencias de enseñanza:

- Proporcionar un modelo o demostraciones de respuestas escritas u orales requeridas o esperadas.
- Comprobar si hay comprensión: use preguntas que requieran respuestas de una sola palabra, apoyos y gestos.
- Hablar con claridad, de manera pausada.
- Evitar el uso de las expresiones coloquiales, complejas.
- Asegurar que los estudiantes tengan todos los materiales necesarios.
- Leer las instrucciones oralmente.
- Corroborar que los estudiantes entiendan las instrucciones.
- Incorporar visuales: gestos, accesorios, gráficos organizadores y tablas.
- Sentarse cerca o junto al estudiante durante el tiempo de estudio.
- Seguir rutinas predecibles para crear un ambiente de seguridad y estabilidad para el aprendizaje.
- Permitir el aprendizaje por descubrimiento, pero estar disponible para ofrecer instrucciones directas sobre cómo completar una tarea.
- Utilizar los organizadores gráficos para la relación de ideas, conceptos y textos.
- Permitir el uso del diccionario regular o ilustrado.
- Crear un glosario pictórico.
- Simplificar las instrucciones.
- Ofrecer apoyo en la realización de trabajos de investigación.
- Ofrecer los pasos a seguir en el desarrollo de párrafos y ensayos.
- Proveer libros o lecturas con conceptos similares, pero en un nivel más sencillo.
- Proveer un lector.
- Proveer ejemplos.
- Agrupar problemas similares (todas las sumas juntas), utilizar dibujos, láminas, o gráficas para apoyar la explicación de los conceptos, reducir la complejidad lingüística del problema, leer y explicar el problema o teoría verbalmente o descomponerlo en pasos cortos.
- Proveer objetos para el aprendizaje (concretizar el vocabulario o conceptos).
- Reducir la longitud y permitir más tiempo para las tareas escritas.
- Leer al estudiante los textos que tiene dificultad para entender.
- Aceptar todos los intentos de producción de voz sin corrección de errores.
- Permitir que los estudiantes sustituyan dibujos, imágenes o diagramas, gráficos, gráficos para una asignación escrita.
- Esbozar el material de lectura para el estudiante en su nivel de lectura, enfatizando las ideas principales.
- Reducir el número de problemas en una página.
- Proporcionar objetos manipulativos para que el estudiante utilice cuando resuelva problemas de matemáticas.

3. Si tu hijo es un estudiante dotado, es decir, que obtuvo 130 o más de cociente intelectual (CI) en una prueba psicométrica, su educación debe ser dirigida y desafiante. Deberán considerar las siguientes recomendaciones:

- Conocer las capacidades especiales del estudiante, sus intereses y estilos de aprendizaje.
- Realizar actividades motivadoras que les exijan pensar a niveles más sofisticados y explorar nuevos temas.
- Adaptar el currículo y profundizar.
- Evitar las repeticiones y las rutinas.
- Realizar tareas de escritura para desarrollar empatía y sensibilidad.
- Utilizar la investigación como estrategia de enseñanza.
- Promover la producción de ideas creativas.
- Permitirle que aprenda a su ritmo.
- Proveer mayor tiempo para completar las tareas, cuando lo requiera.
- Cuidar la alineación entre su educación y sus necesidades académicas y socioemocionales.