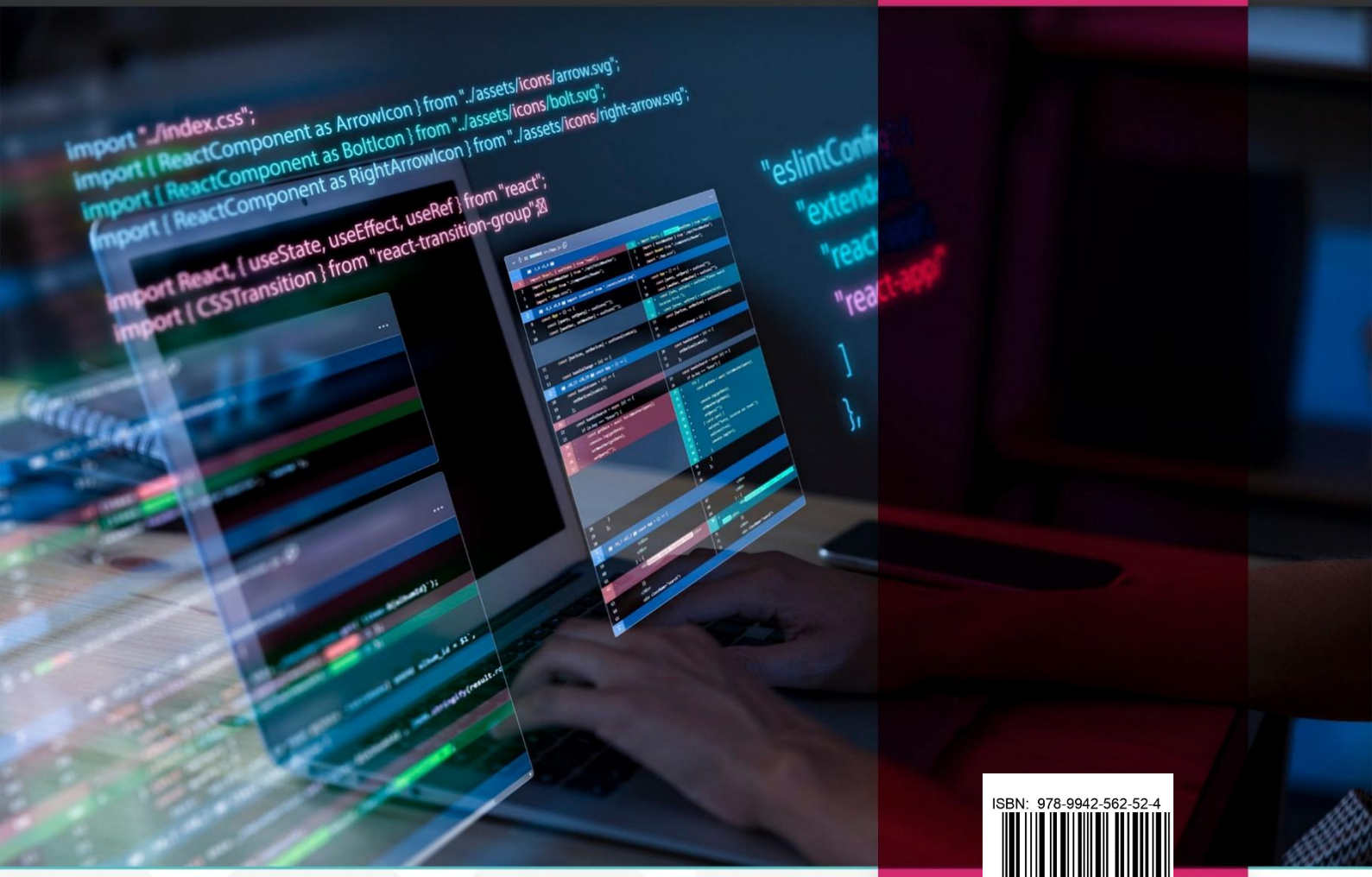
INSTITUTO SUPERIOR TECNOLÓGICO QUITO

Formamos tu PROPÓSITO DE VIDA

Guía práctico experimental de

PROGRAMACIÓN

Autor: Alexis Xavier Urgíles Pillalaza



ISBN: 978-9942-562-52-4

2025

GUÍA PRÁCTICO EXPERIMENTAL DE PROGRAMACIÓN

CARRERA: ANIMACIÓN MULTIMEDIA

PERIODO ACADÉMICO: SEPTIEMBRE 2025 – FEBRERO 2026

DOCENTE TUTOR: ALEXIS URGILÉS PILLALAZA

EDICIÓN: PRIMERA

AÑO: 2025

TRABAJO EN EDICIÓN:

DOXA EDITION

EQUIPO EDITORIAL:

MSc. KEYERMAN TOAPANTA CISNEROS

MSc. DAVID ALEJANDRO PEREIRA SALAZAR

ISBN:978-9942-562-52-4

ISBN: 978-9942-562-52-4

9 789942 562524

QUITO – ECUADOR

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

1. GUÍA PRÁCTICA EXPERIMENTAL DE PROGRAMACIÓN

1.1 Descripción de la asignatura

La asignatura de Programación proporciona a los estudiantes una comprensión sólida de los principios básicos de la programación, centrándose en el desarrollo de habilidades algorítmicas y la lógica computacional. A lo largo del curso, los estudiantes aprenderán a descomponer problemas complejos en soluciones simples mediante el uso de lenguajes de programación, particularmente PsInt para los fundamentos algorítmicos, y Processing para la visualización gráfica interactiva.

Se explorarán conceptos clave como la estructura de control, manipulación de datos, funciones y arreglos, que son esenciales para cualquier tipo de programación. Además, se hará énfasis en la creación de aplicaciones interactivas visuales que son altamente relevantes en el ámbito de la multimedia y el diseño gráfico.

Esta asignatura no solo proporcionará las bases teóricas, sino que también ayudará a los estudiantes a desarrollar la lógica que necesitarán para resolver problemas en proyectos reales relacionados con la creación de gráficos interactivos, animaciones y medios visuales. La programación en Processing permitirá a los estudiantes aplicar lo aprendido en proyectos multimedia interactivos, como animaciones y aplicaciones gráficas.

Esta asignatura se integra en la carrera de Multimedia para dotar a los estudiantes de las herramientas tecnológicas que necesitarán en áreas como el diseño interactivo, el arte digital, la animación y el desarrollo de interfaces, lo que enriquece su perfil profesional en un mercado laboral que demanda cada vez más habilidades técnicas y creativas combinadas.

1.2 Objetivos de la asignatura

Brindar a los estudiantes la capacidad de analizar y comprender los principios y fundamentos de la programación orientada a la resolución de problemas. A través de la exploración crítica de estructuras algorítmicas y conceptos fundamentales de programación, así como la aplicación de lenguajes de programación como PsInt y Processing, los estudiantes desarrollarán habilidades para diseñar y crear soluciones a problemas reales, con un enfoque especial en el campo de la multimedia, este conocimiento les permitirá aplicar herramientas como Processing para la creación de aplicaciones gráficas interactivas y visualizaciones multimedia.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

1.3 Contenidos mínimos

1. Lenguaje de programación.
2. Algoritmos.
3. Solución de problemas.
4. Estructura de control y evaluación.
5. Programación estructural.
6. Serialización

1.4 Resultados de aprendizaje

Desarrolla e implementa programas y aplicaciones con el uso de las tecnologías para la elaboración de algoritmos utilizando lenguaje de programación.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

2 UNIDAD 1: Introducción a la programación

2.1 Practica experimental 1

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Construcción de algoritmos cotidianos mediante pensamiento lógico estructurado

Objetivos

Elaborar algoritmos cotidianos utilizando lógica secuencial clara, identificando entradas, procesos y salidas, como base para el pensamiento computacional y la posterior formalización en pseudocódigo.

Antecedentes/Escenario

La programación se fundamenta en la capacidad de representar procesos de manera estructurada. Actividades diarias como preparar una bebida, utilizar un dispositivo o clasificar información siguen secuencias lógicas que pueden convertirse en algoritmos. Representar estas tareas de forma desambiguada fortalece la comprensión del flujo lógico previo a la codificación.

Recursos necesarios

Formato de taller.

Cuaderno o documento digital para escribir los algoritmos.

Conexión a internet (opcional).

Aula virtual para la entrega.

Video de apoyo sobre algoritmos (del repositorio institucional).

Ejemplos mostrados en clase.

Planteamiento del problema

Las tareas cotidianas suelen describirse de manera vaga o subjetiva, lo que dificulta su traducción a instrucciones precisas. La actividad consiste en transformar procesos comunes en

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

algoritmos claros, ordenados y sin ambigüedad, identificando qué elementos funcionan como entradas, proceso y salida.

Pasos por realizar

Paso 1: Selección de tres actividades de la vida diaria.

Paso 2: Identificación de entradas, secuencia de pasos y resultados.

Paso 3: Redacción de los algoritmos con pasos numerados y precisos.

Paso 4: Verificación de coherencia lógica y ausencia de ambigüedad.

Paso 5: Organización del contenido en formato de entrega.

Desarrollo

Para iniciar la actividad se seleccionaron tres procesos cotidianos que permiten evidenciar con claridad la presencia de una secuencia lógica y de elementos identificables como entradas, proceso y salida. Las actividades escogidas fueron: preparar una bebida caliente, encender y preparar una computadora para trabajar y organizar archivos digitales en carpetas. La elección de estas tareas responde a su frecuencia en la vida diaria y a la facilidad con la que pueden transformarse en algoritmos estructurados.

En la primera actividad, preparar una bebida caliente, se identificaron como entradas el agua, una taza, un hervidor, un sobre de té o café y una cuchara. El proceso consistió en calentar el agua, colocar el sobre en la taza, verter el agua caliente y mezclar el contenido. La salida final se representa como la bebida lista para ser consumida. Este ejemplo permite comprender la importancia de describir cada acción de forma precisa para evitar ambigüedades en la interpretación.

La segunda actividad, encender una computadora personal, se analizó considerando como entradas el equipo, la conexión eléctrica y la intervención del usuario. El proceso abarcó la acción de encender el dispositivo, iniciar sesión y abrir los programas necesarios para comenzar a trabajar. La salida obtenida es el equipo completamente operativo. Este caso evidencia la estructura lineal de tareas que deben ejecutarse en orden para obtener un resultado funcional.

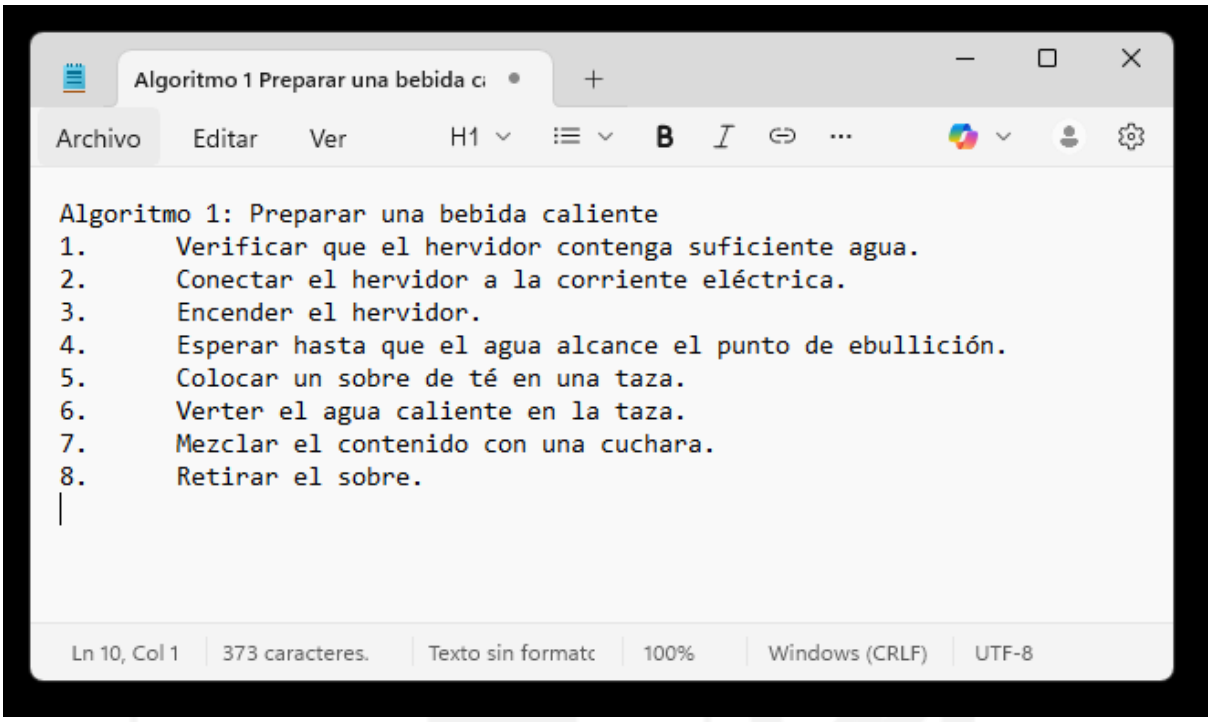
Finalmente, la actividad de organizar archivos digitales estableció como entradas los archivos, las carpetas y la computadora. El proceso se centró en revisar el contenido, clasificar los elementos y reubicarlos según una lógica determinada. La salida se refleja en un directorio ordenado. Esta actividad resulta útil para practicar la descomposición de un proceso que, aunque simple, requiere decisiones sobre clasificación y estructura.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Estos tres ejemplos evidencian cómo los procesos cotidianos pueden transformarse en algoritmos claros y secuenciales, permitiendo comprender la lógica previa al diseño de pseudocódigo y fortaleciendo el pensamiento computacional desde una perspectiva aplicada.

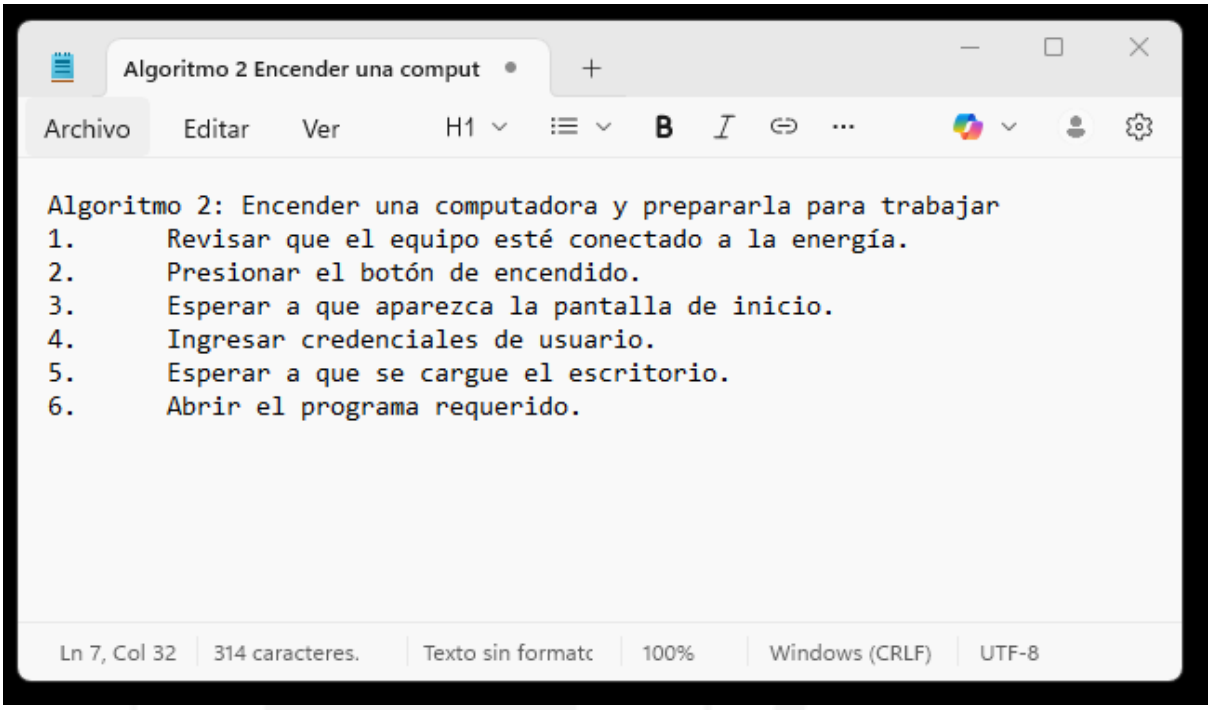
Algoritmos desarrollados

Algoritmo 1: Preparar una bebida caliente

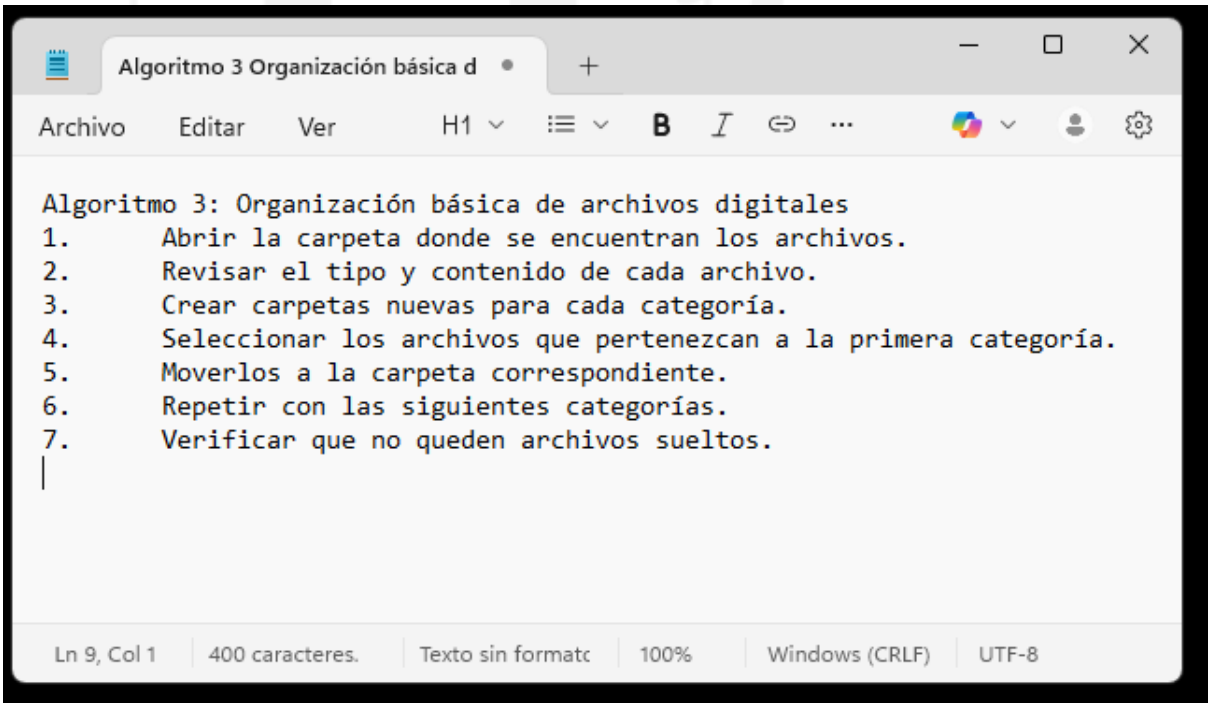


Algoritmo 2: Encender una computadora y prepararla para trabajar

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



Algoritmo 3: Organización básica de archivos digitales



Resultados esperados

Se espera que el estudiante desarrolle la capacidad de descomponer actividades cotidianas en algoritmos estructurados, identificando correctamente entradas, procesos y salidas. Al finalizar la práctica, el estudiante deberá redactar secuencias lógicas claras, ordenadas y sin ambigüedad, aplicando pensamiento computacional básico. Además, se prevé que logre analizar la coherencia de

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

cada paso, detectando posibles vacíos lógicos y mejorando su precisión. Finalmente, el estudiante demostrará comprensión inicial de cómo estos algoritmos podrán traducirse posteriormente a pseudocódigo y programación formal.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

2.2 Practica experimental 2

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Representación de algoritmos mediante diagramas de flujo

Objetivos

Representar la lógica de un proceso mediante un diagrama de flujo empleando simbología estándar para inicio, procesos, decisiones y fin, fortaleciendo la comprensión visual del recorrido lógico antes de la codificación.

Antecedentes/Escenario

Los diagramas de flujo permiten visualizar el comportamiento secuencial de un algoritmo antes de programarlo. En programación, videojuegos y multimedia, esta representación ayuda a anticipar errores, identificar bifurcaciones y analizar comportamientos.

Procesos simples como verificar una contraseña, seleccionar una opción o determinar un resultado basado en condiciones pueden ser modelados mediante diagramas que revelan la estructura lógica del problema

Recursos necesarios

Formato de taller.
Papel o documento digital para diagramar.
Plantilla de símbolos estándar (inicio/fin, proceso, decisión, entrada/salida).
Herramientas sugeridas: Draw.io, Figma, PowerPoint, LibreOffice Draw.
Ejemplo de diagrama mostrado en clase.

Planteamiento del problema

Los procesos que incluyen decisiones suelen generar confusión si no se visualizan correctamente. El desafío consiste en convertir un proceso cotidiano con al menos una condición en un diagrama de flujo claro, correctamente conectado y con rutas alternativas bien definidas

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Pasos por realizar

Paso 1: Selección de un proceso cotidiano que contenga decisiones.

Paso 2: Identificación de elementos: entradas, pasos y posibles caminos.

Paso 3: Elaboración de un borrador del algoritmo en forma secuencial.

Paso 4: Construcción del diagrama de flujo utilizando la simbología correcta.

Desarrollo

Selección del proceso. - El proceso elegido para el ejemplo consiste en la validación de una contraseña antes de acceder a un sistema.

Identificación de Elementos. - Entradas: usuario, contraseña ingresada.
Proceso: solicitar contraseña, compararla con la contraseña correcta, permitir o denegar el acceso.
Salida: acceso concedido o mensaje de error.

Elaboración de un borrador del algortimo

Solicitar al usuario que ingrese una contraseña.

Comparar la contraseña ingresada con la contraseña correcta.

Si la contraseña es correcta, permitir acceso.

Si la contraseña es incorrecta, mostrar mensaje de error.

Finalizar el proceso.

Construcción del diagrama de flujo (Descripción estructurada)

Inicio

Representado con el símbolo de terminador.

Solicitar contraseña

Bloque de proceso donde se registra el dato.

Comparar contraseña ingresada con contraseña correcta

Flecha hacia un símbolo de **decisión**.

Decisión: ¿La contraseña es correcta?

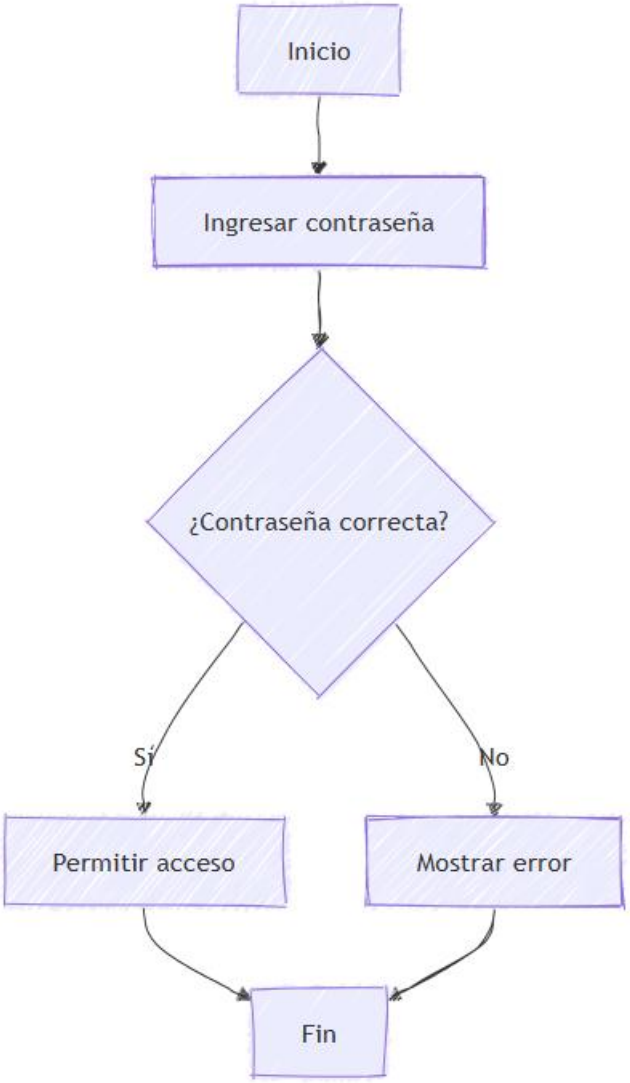
Sí → Dirige hacia un proceso que indica “Acceso permitido”.

No → Dirige hacia un proceso que indica “Mostrar mensaje de error”.

Fin

Ambas rutas convergen hacia un terminador.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



Nota: Diagrama de flujo creado con autoría propia creado en https://mermaid.live

Resultados esperados

Se espera que el estudiante adquiera la capacidad de representar algoritmos mediante diagramas de flujo aplicando simbología estándar de forma adecuada. Al finalizar la actividad, deberá identificar correctamente entradas, procesos y decisiones, estructurando el recorrido lógico sin ambigüedades. Asimismo, el estudiante demostrará comprensión visual del flujo del algoritmo, diferenciando rutas alternativas y asegurando conexiones coherentes entre cada componente. Finalmente, será capaz de modelar procesos cotidianos con decisiones, anticipando posibles errores antes de su implementación en pseudocódigo o programación.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

2.3 Practica experimental 3

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Esquemas de algoritmos que utilizan decisiones o ciclos

Objetivos

Elaborar esquemas algorítmicos que incorporen estructuras de decisión y repetición, identificando correctamente sus condiciones, rutas y comportamientos lógicos, fortaleciendo el análisis previo a la formalización en pseudocódigo.

Antecedentes/Escenario

Las estructuras de decisión (IF–ELSE) y las estructuras repetitivas (WHILE, REPEAT, FOR) permiten modelar comportamientos dinámicos en algoritmos. Estas estructuras son fundamentales en programación, ya que permiten: Tomar decisiones basadas en condiciones, repetir acciones hasta cumplir un objetivo.

Controlar flujos variables en videojuegos, interfaces o sistemas interactivos. Ejemplos cotidianos como verificar si un documento está completo, repetir un intento, contar elementos o clasificar información incorporan implícitamente decisiones o ciclos.

Recursos necesarios

Formato de taller
Cuaderno o documento digital.
Ejemplos proporcionados en clase.
Material audiovisual sobre estructuras algorítmicas condicionales y repetitivas.

Planteamiento del problema

Muchos procesos requieren rutas alternativas o la repetición de acciones hasta alcanzar una condición específica. El desafío consiste en transformar tareas habituales en esquemas algorítmicos

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

que utilicen al menos una decisión y un ciclo, logrando claridad en las condiciones y en los momentos en que la ejecución avanza, se detiene o se repite.

Pasos por realizar

Paso 1: Selección de dos procesos cotidianos que incluyan decisiones o repeticiones.

Paso 2: Identificación de entradas, condiciones, rutas y resultados.

Paso 3: Elaboración del esquema algorítmico utilizando listas estructuradas.

Paso 4: Verificación de coherencia en las condiciones y flujo del proceso.

Desarrollo

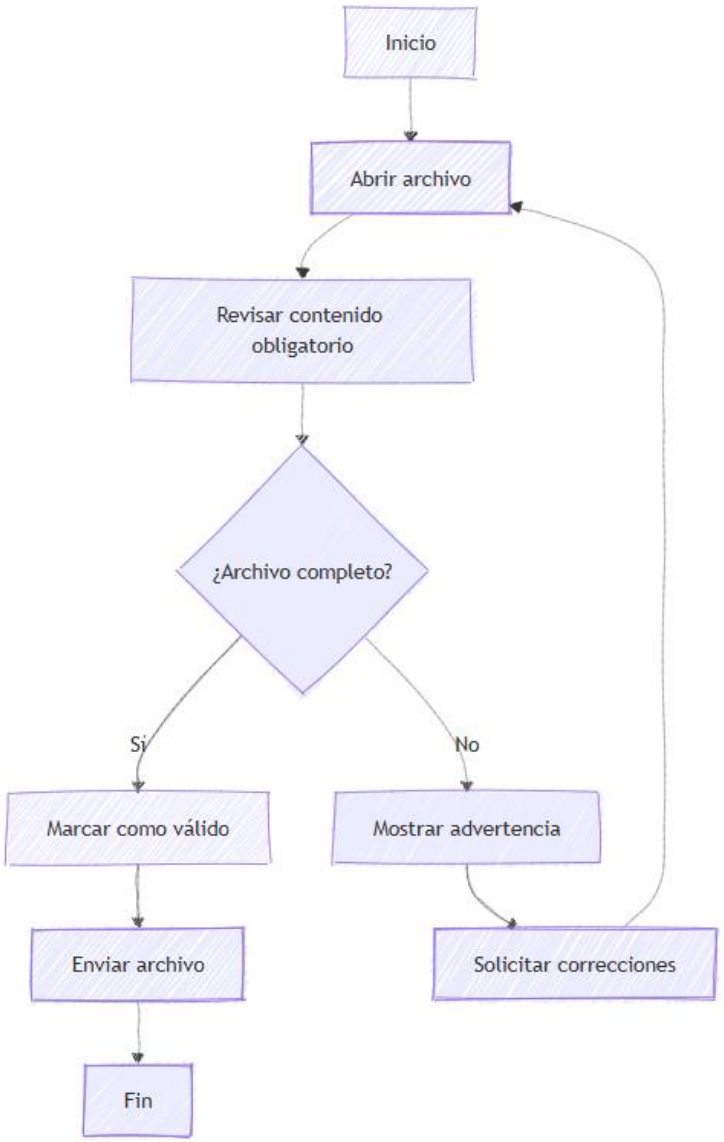
Para la elaboración de los esquemas algorítmicos se seleccionaron dos procesos cotidianos que permiten evidenciar el uso de decisiones y ciclos dentro de un flujo lógico estructurado. El primer proceso elegido corresponde a la verificación de un archivo digital antes de enviarlo, dado que este tipo de tarea requiere comparar requisitos, evaluar condiciones y tomar rutas alternativas según el estado del archivo. El segundo proceso consiste en contar los elementos de una lista, actividad que se presta de manera natural al uso de un ciclo repetitivo debido a que implica recorrer secuencialmente cada elemento hasta alcanzar el total.

En el análisis del primer proceso, verificar si un archivo está completo, se identificaron como entradas el archivo digital y la lista de requisitos exigidos. La condición principal consiste en determinar si el archivo cumple o no con todos los elementos solicitados, lo que conduce a dos posibles rutas: enviar el archivo cuando se encuentra completo o solicitar correcciones cuando no lo está. El proceso implica revisar el contenido, validar cada requisito y decidir si se debe continuar o regresar al inicio para corregirlo. La salida final se refleja en un archivo correctamente validado y enviado. Esta estructura permite visualizar cómo una sola decisión puede modificar el flujo del algoritmo y generar pequeñas repeticiones hasta lograr el resultado esperado.

En cuanto al segundo proceso, contar elementos de una lista, se estableció como entrada la lista que se desea recorrer. El comportamiento algorítmico inicia con la inicialización de un contador en cero y la colocación del índice en la primera posición. A partir de allí, se ejecuta un ciclo que se repite mientras existan elementos sin revisar. En cada iteración, el algoritmo accede al elemento actual, incrementa el contador y avanza al siguiente elemento hasta finalizar el recorrido. La salida del proceso es el valor final del contador, que representa la cantidad total de elementos encontrados. Este ejercicio permite comprender de manera clara cómo se comporta un ciclo secuencial y cómo las condiciones controlan su inicio y finalización.

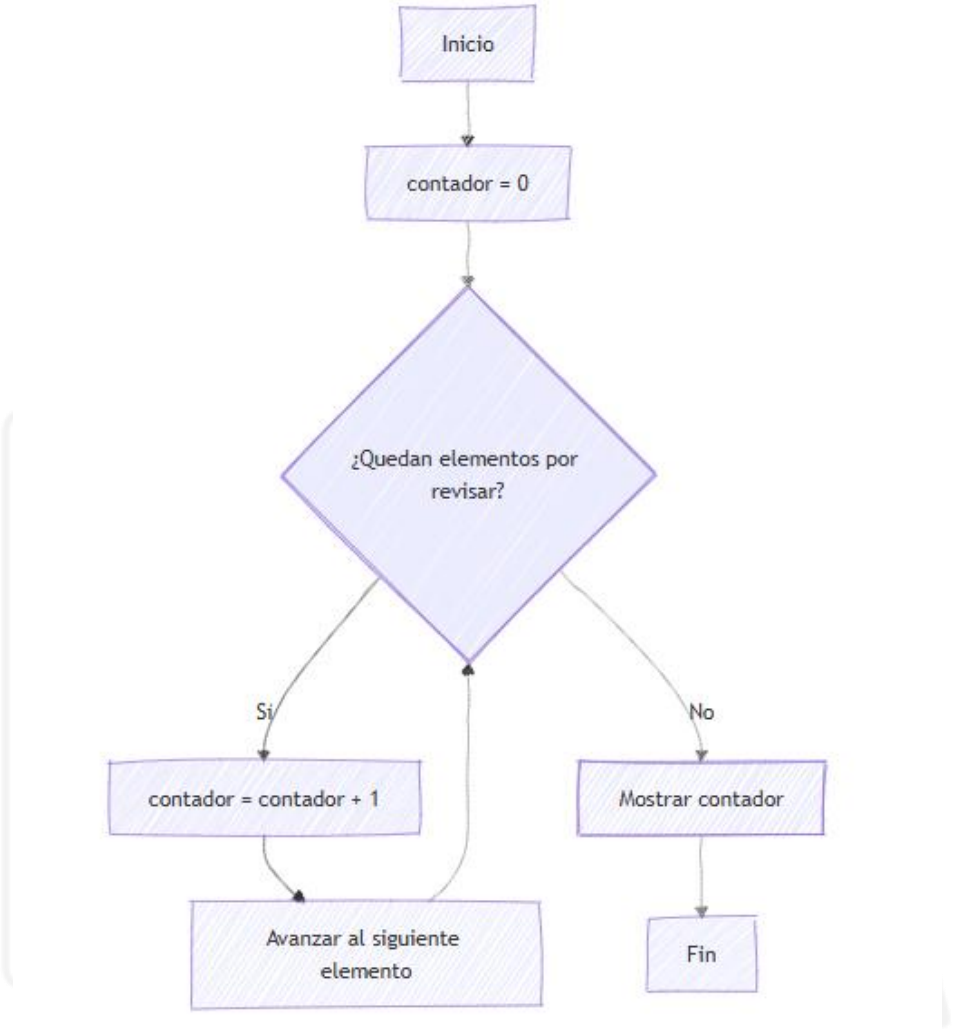
(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Ambos procesos evidencian la importancia de las estructuras de decisión y repetición dentro del diseño algorítmico. La verificación del archivo demuestra cómo una condición puede redirigir el flujo del proceso, mientras que el conteo de elementos ilustra un ciclo repetitivo controlado por una condición de continuidad. Estos esquemas permiten fortalecer el pensamiento lógico y preparan al estudiante para formalizar posteriormente estos procesos en pseudocódigo o diagramas de flujo más complejos.



Nota: El diagrama de flujo del algoritmo 1.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



Nota: El diagrama de flujo del algoritmo 2.

Resultados esperados

Se espera que el estudiante sea capaz de construir esquemas algorítmicos que integren correctamente estructuras de decisión y repetición, identificando sus condiciones, flujos alternativos y puntos de control. Al finalizar la actividad, deberá representar procesos cotidianos mediante esquemas organizados, evidenciando lógica coherente y secuencias bien estructuradas. Asimismo, demostrará comprensión del comportamiento dinámico de ciclos y decisiones, reconociendo cuándo una acción debe evaluarse, repetirse o finalizar, fortaleciendo así la base conceptual necesaria para la formalización en pseudocódigo.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

2.4 Practica experimental 4

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Proponer soluciones a problemas encontrados en algoritmos del proyecto (videojuego del perro saltador)

Objetivos

Analizar errores o ineficiencias en los algoritmos iniciales del proyecto del videojuego y proponer soluciones estructuradas que optimicen la lógica del juego.

Antecedentes/Escenario

En el proyecto se desarrolla un videojuego donde un perro corre automáticamente, salta plataformas y esquiva obstáculos. El sistema controla saltos, colisiones y conteo de obstáculos superados. Los primeros algoritmos suelen contener problemas comunes como condiciones mal planteadas, ciclos infinitos o colisiones mal procesadas.

Identificar y corregir estos problemas permite mejorar la experiencia de juego y preparar el código para su implementación en Processing o un motor posterior.

Recursos necesarios

Documento o boceto de los algoritmos originales del proyecto.
Notas del flujo del juego.
Hojas de revisión o editor digital.

Planteamiento del problema

Los algoritmos iniciales presentan errores en la detección de colisiones, conteo de saltos o en la transición entre niveles. Se requiere un análisis detallado para detectar problemas y plantear mejoras claras y aplicables.

Pasos por realizar

Paso 1: Identificar las secciones problemáticas del algoritmo.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Paso 2: Describir el error y su causa probable.

Paso 3: Proponer una solución detallada.

Paso 4: Reescribir la versión corregida del algoritmo.

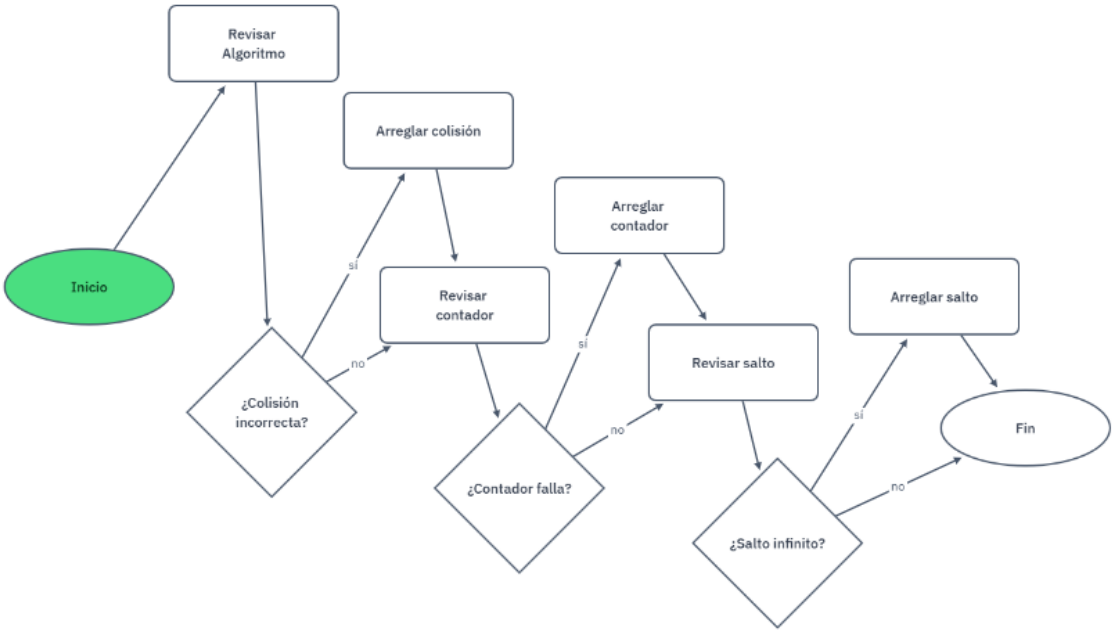
Desarrollo

Durante el análisis del proyecto del videojuego del perro saltador se identificaron varios problemas dentro de los algoritmos iniciales que afectaban directamente el funcionamiento del juego. El primer inconveniente detectado correspondió a una detección de colisión incorrecta, ya que el sistema marcaba una colisión incluso cuando el perro no llegaba a tocar el obstáculo. Este comportamiento se debía a que la condición evaluaba únicamente la posición horizontal, ignorando la comparación con la posición vertical del personaje. Para corregirlo, se planteó una solución que incorpora la verificación simultánea de las coordenadas X e Y, garantizando que la colisión solo se active cuando realmente exista superposición entre ambas dimensiones. La versión ajustada del algoritmo obtiene primero las posiciones del perro y del obstáculo para luego validar la superposición en X y la proximidad del perro respecto a la altura del obstáculo, logrando así una detección más precisa.

El segundo problema relevante se relacionó con el conteo incorrecto de los obstáculos superados. El contador incrementaba repetidamente mientras el perro se mantenía en estado de salto, lo que generaba registros inflados y sin relación con los obstáculos reales. Tras revisar la lógica, se determinó que la causa estaba en que el incremento estaba ligado al estado del personaje y no al momento exacto en que el obstáculo pasaba la posición del perro. La solución consistió en condicionar el conteo al cruce efectivo del obstáculo frente al personaje y en marcar dicho obstáculo como “contado” para evitar duplicaciones. Con el ajuste, el contador aumenta solo cuando la posición del obstáculo se sitúa por detrás del perro por primera vez, generando un comportamiento más consistente y coherente con la dinámica del juego.

Finalmente, se encontró un tercer problema en la mecánica del salto, donde el personaje quedaba atrapado en un ciclo infinito que impedía su descenso. La causa principal radicaba en que el algoritmo únicamente incrementaba la altura del salto sin contemplar una fase de caída. La solución propuso dividir el salto en dos fases claramente definidas: un ascenso controlado por un impulso positivo hasta alcanzar una altura máxima y un descenso progresivo regulado por la gravedad hasta volver al nivel del suelo. Este ajuste permite que el personaje alterna de manera natural entre los estados de salto, descenso y carrera, eliminando comportamientos anómalos y favoreciendo una jugabilidad más fluida.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



(+593) 96 356 1961
admisiones@itq.edu.ec
Antonio de Ulloa N28-30 y Diego de Atienza (Esq).
WWW.ITQ.EDU.EC

2.5 Practica experimental 5

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Construir cinco esquemas algorítmicos independientes que incluyan estructuras condicionales y repetitivas.

Objetivos

Construir cinco esquemas algorítmicos independientes que incluyan estructuras condicionales y repetitivas.

Antecedentes/Escenario

Las decisiones y ciclos permiten modelar comportamientos dinámicos como validaciones, repeticiones o recorridos de datos. Su comprensión es esencial para videojuegos, aplicaciones y sistemas interactivos.

Recursos necesarios

Ejemplo de diagrama mostrado en clase.

Formato de taller.

Ejemplos de estructuras IF y WHILE del curso.

Planteamiento del problema

El reto consiste en crear cinco algoritmos que representen distintos contextos donde se apliquen decisiones o ciclos de manera clara y funcional.

Pasos por realizar

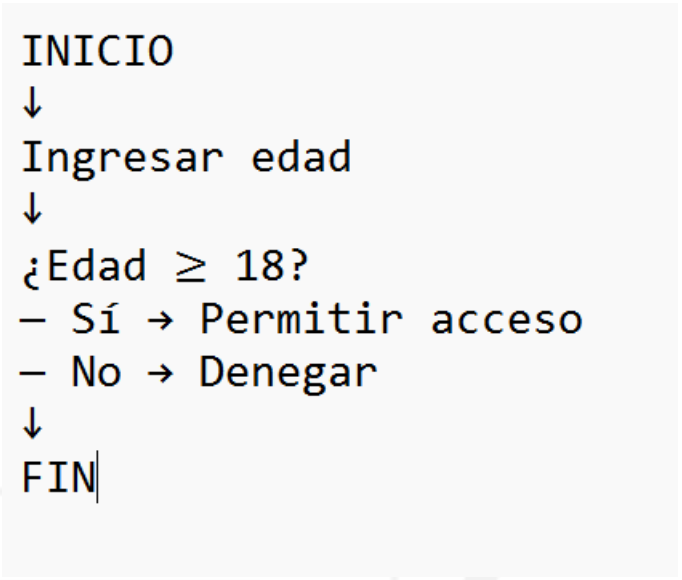
Selección de contextos.

Identificación de condiciones.

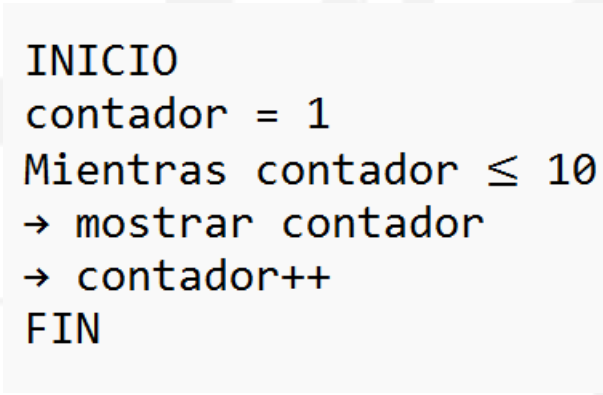
(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Construcción de los esquemas.

Desarrollo



NOTA: Algoritmo informal realizado en bloc de notas



NOTA: Algoritmo informal realizado en bloc de notas

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

```
INICIO
↓
Mostrar menú
↓
Ingresar opción
↓
¿Opción válida?
– Sí → Ejecutar acción
– No → Solicitar nuevamente
FIN
```

NOTA: Algoritmo informal realizado en bloc de notas

```
INICIO
carga = 0
Mientras carga < 100
→ carga += 5
→ Mostrar barra
FIN
```

NOTA: Algoritmo informal realizado en bloc de notas

```
INICIO
índice = 0
Mientras nombre[índice] != buscado
→ índice++
Si final de lista → Mostrar “No encontrado”
FIN
```

NOTA: Algoritmo informal realizado en bloc de notas

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Resultados esperados

Al finalizar la actividad, el estudiante será capaz de construir cinco esquemas algorítmicos coherentes, cada uno con estructuras condicionales y repetitivas correctamente definidas. Se espera que logre identificar condiciones, rutas alternativas y ciclos funcionales dentro de diversos contextos, demostrando dominio del análisis lógico previo a la programación. Los esquemas deberán reflejar claridad secuencial, ausencia de ambigüedad y capacidad para transformar situaciones reales en modelos algorítmicos bien estructurados, fortaleciendo así su pensamiento computacional.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

2.6 Practica experimental 6

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Creación de pseudocódigo con funciones simples

Objetivos

Diseñar pseudocódigo empleando funciones básicas que encapsulen acciones específicas.

Antecedentes/Escenario

El uso de funciones permite dividir tareas complejas en unidades pequeñas reutilizables, mejorando la claridad y organización del código previo a la programación real.

Recursos necesarios

Formato de pseudocódigo.

Referencias de funciones simples.

Planteamiento del problema

Crear funciones sencillas que resuelvan tareas concretas y luego integrarlas en un pseudocódigo principal.

Pasos por realizar

Paso 1: Crear un pseudocodigo en PSEInt

Paso 2: Definir 2–3 funciones.

Paso 3: Integrarlas en un algoritmo principal.

Paso 4: Ejecutar el algoritmo

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Desarrollo

```
Proceso Principal
    Definir n1, n2, n3 Como Real
    Definir suma, promedio Como Real
    Definir par Como Logico

    Escribir "Ingrese el primer número:"
    Leer n1

    Escribir "Ingrese el segundo número:"
    Leer n2

    Escribir "Ingrese el tercer número:"
    Leer n3

    suma ← Sumar(n1, n2)
    promedio ← CalcularPromedio(n1, n2, n3)
    par ← EsPar(n1)

    Escribir "La suma de los dos primeros números es: ", suma
    Escribir "El promedio de los tres números es: ", promedio

    Si par Entonces
        Escribir "El primer número es par."
    SiNo
        Escribir "El primer número no es par."
    FinSi

FinProceso
```

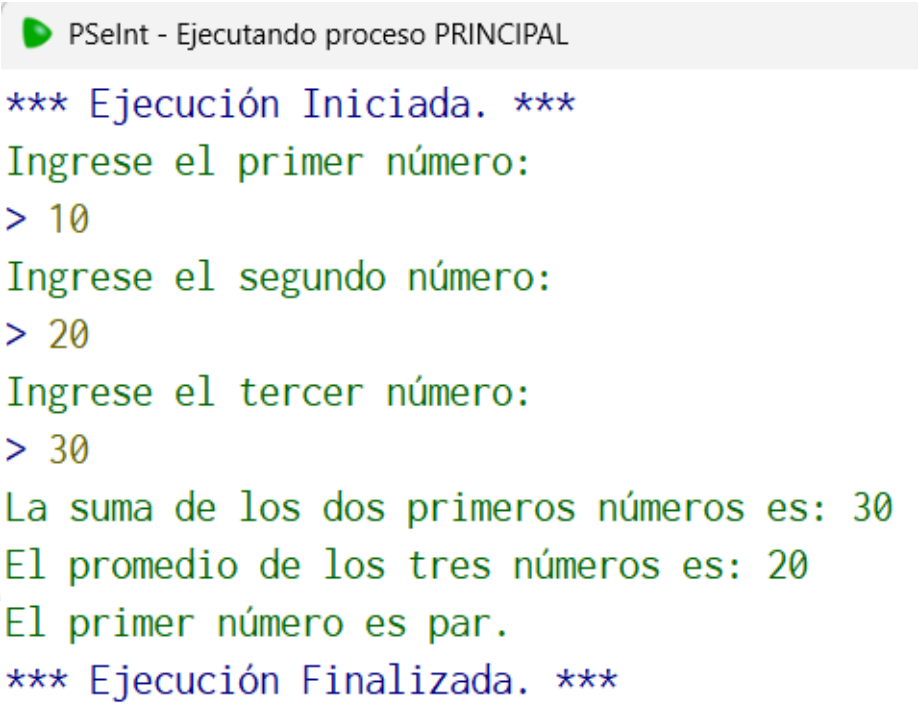
NOTA: Primer proceso creado en PSEint.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

```
Funcion resultado ← Sumar(a, b)
    resultado ← a + b
FinFuncion


Funcion resultado ← CalcularPromedio(a, b, c)
    resultado ← (a + b + c) / 3
FinFuncion


Funcion resultado ← EsPar(num)
    Si num MOD 2 = 0 Entonces
        resultado ← Verdadero
    SiNo
        resultado ← Falso
    FinSi
FinFuncion
```

NOTA: Continuación del algoritmo de la figura 9.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Continuación del algoritmo de la figura 9.

Resultados esperados

Al completar la actividad, el estudiante será capaz de estructurar pseudocódigo empleando funciones simples que organicen acciones de manera modular. Se espera que identifique adecuadamente tareas que pueden encapsularse, defina funciones reutilizables con entradas y salidas claras, y las integre dentro de un algoritmo principal que coordine el flujo general. El resultado deberá mostrar un pseudocódigo ordenado, comprensible y funcional, evidenciando habilidades de abstracción y una correcta separación lógica por responsabilidades.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

3 UNIDAD 2: Programación Gráfica e Interactiva

3.1 Practica experimental 7

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Creación de código que muestre vibe coding

Objetivos

Generar un sketch en Processing que muestre movimiento y color al ritmo de variaciones visuales tipo "vibe coding".

Antecedentes/Escenario

Vibe coding utiliza transiciones suaves de color, movimiento orgánico y cambios rítmicos para crear animaciones estéticas.

Recursos necesarios

Processing instalado.

Planteamiento del problema

Construir una animación que presente pulsos de color y movimiento sinusoidal.

Pasos por realizar

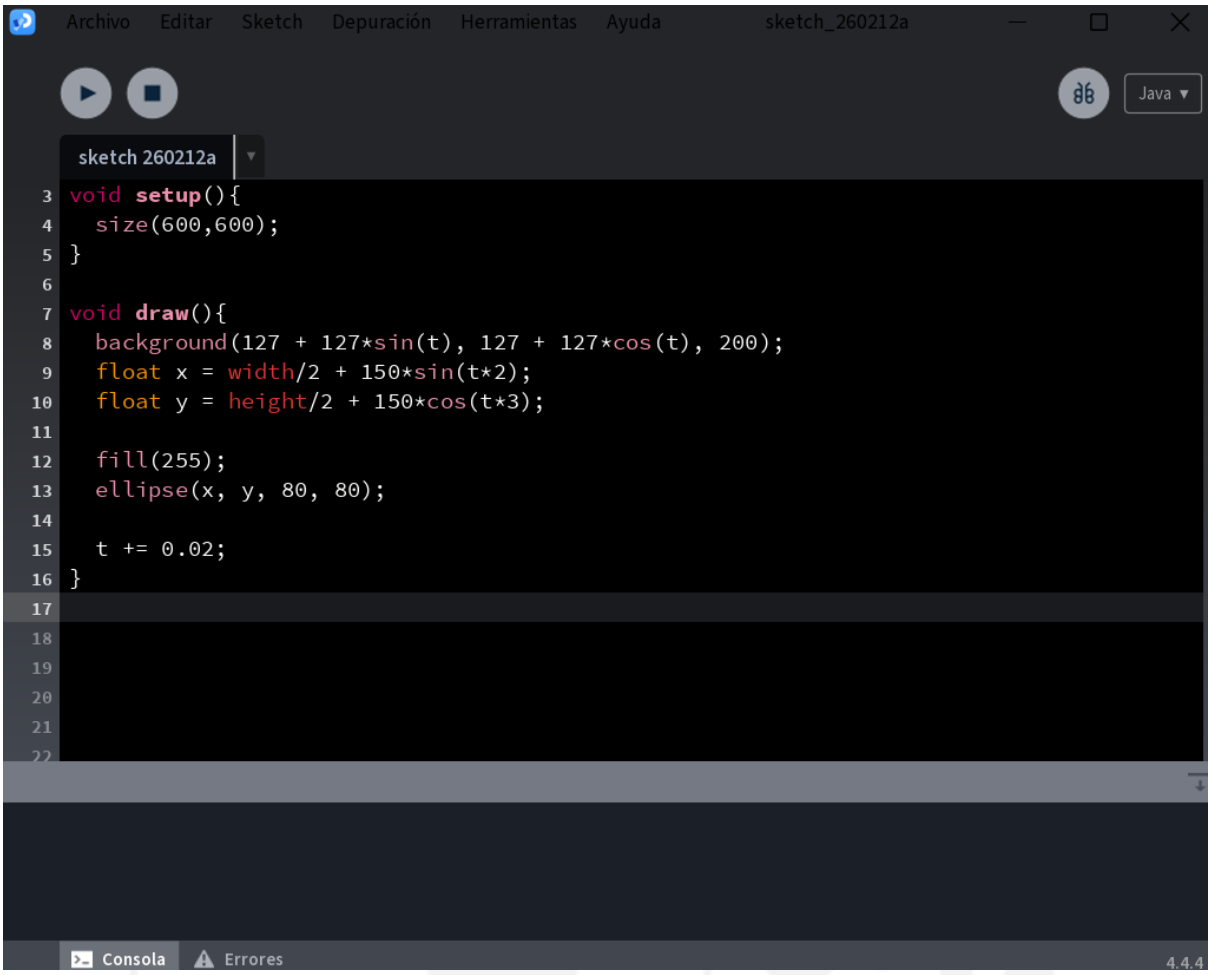
Paso 1: Crear lienzo.

Paso 2: Añadir variaciones de color.

Paso 3: Animar movimientos.

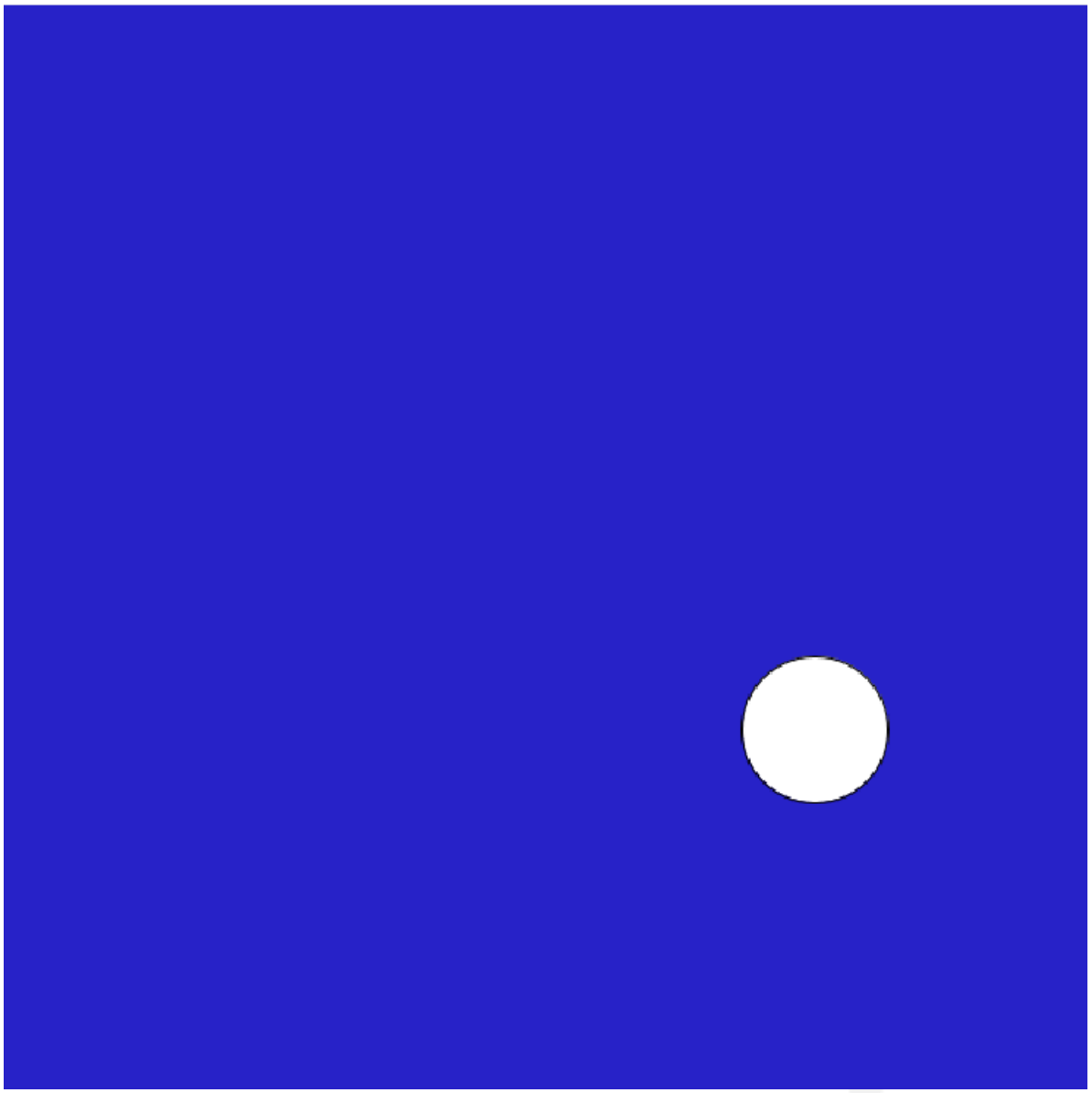
(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Desarrollo



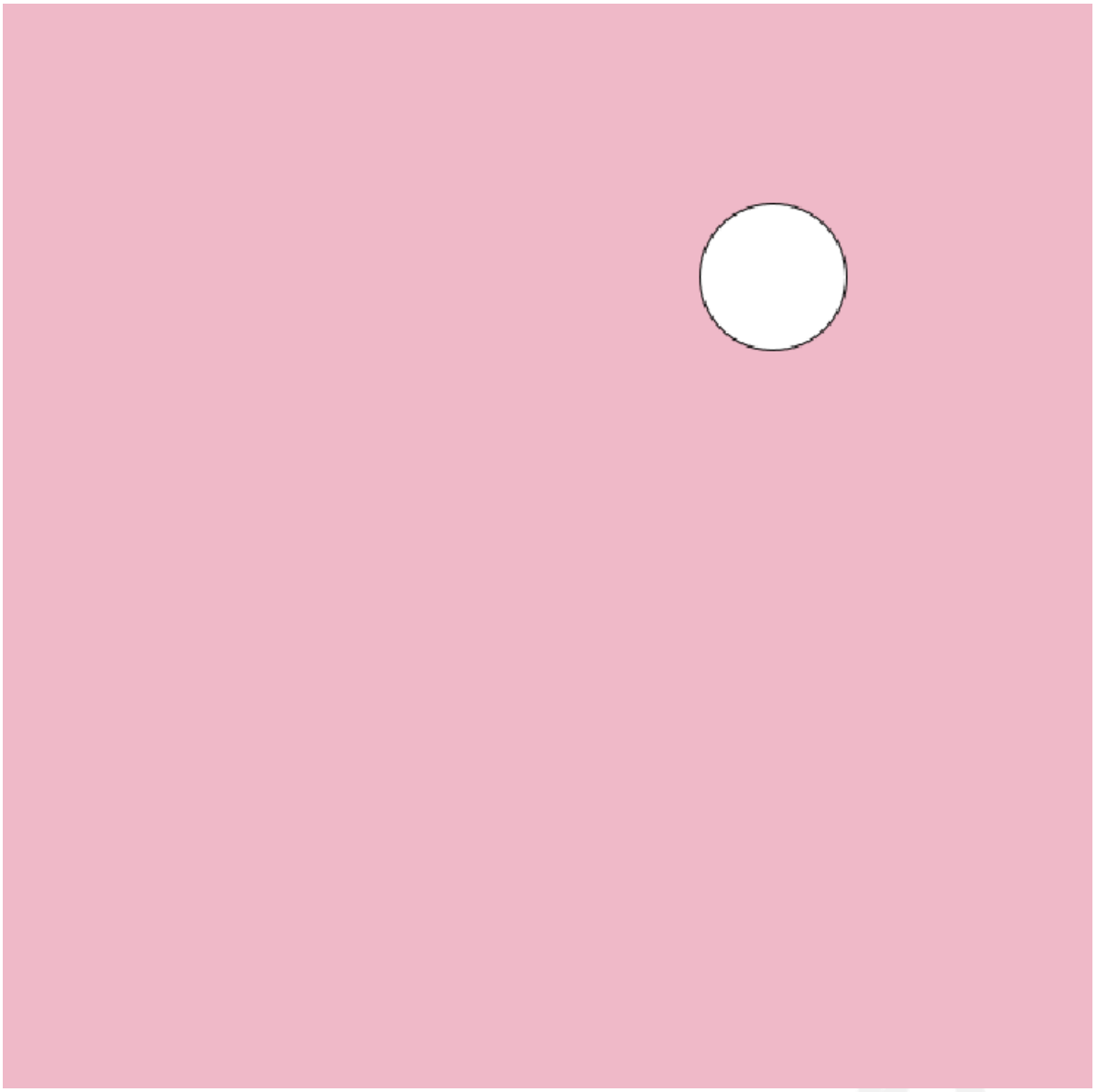
NOTA: Algoritmo integrado en Processing.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



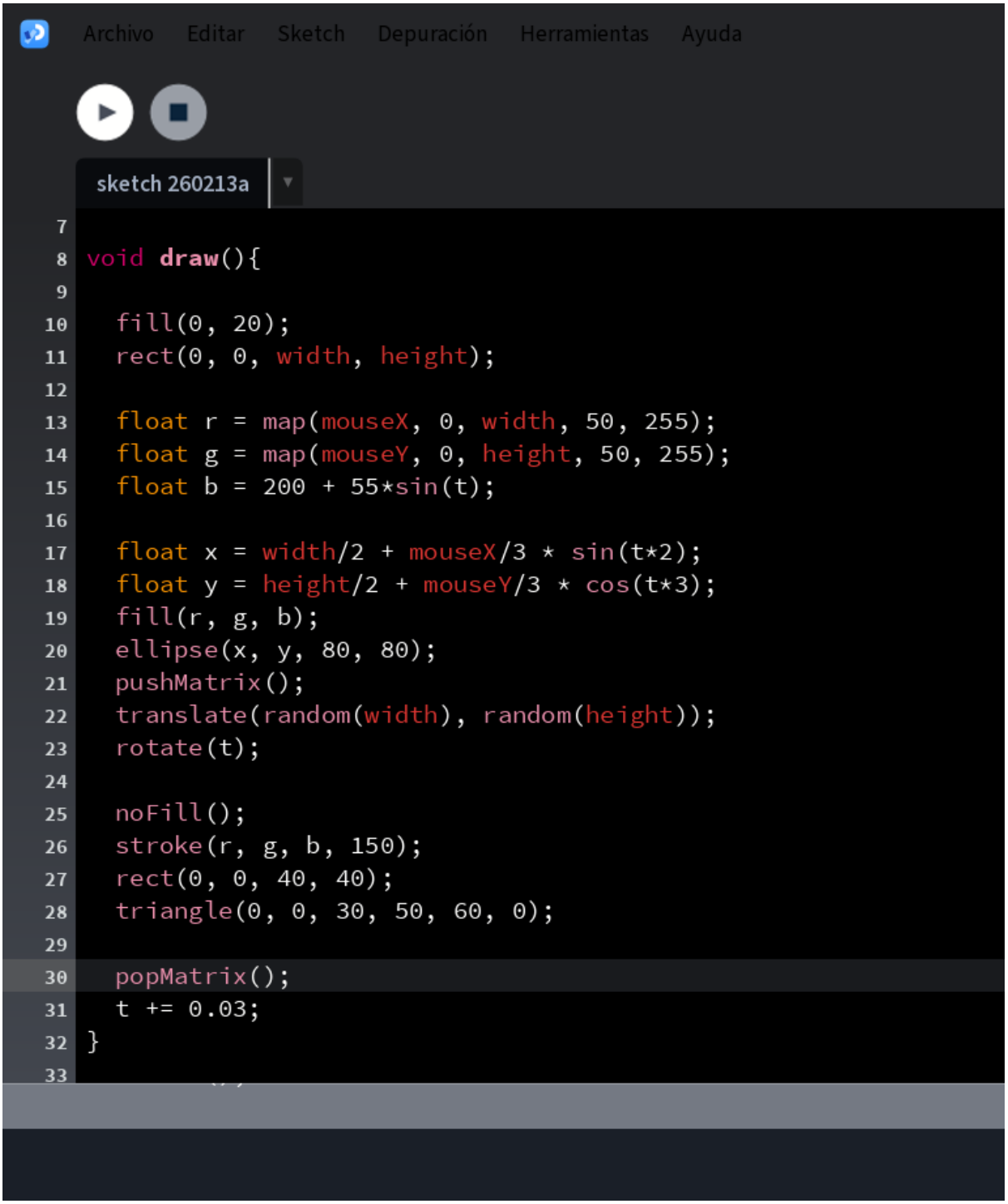
NOTA: Opción de una de las posiciones y colores variantes.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



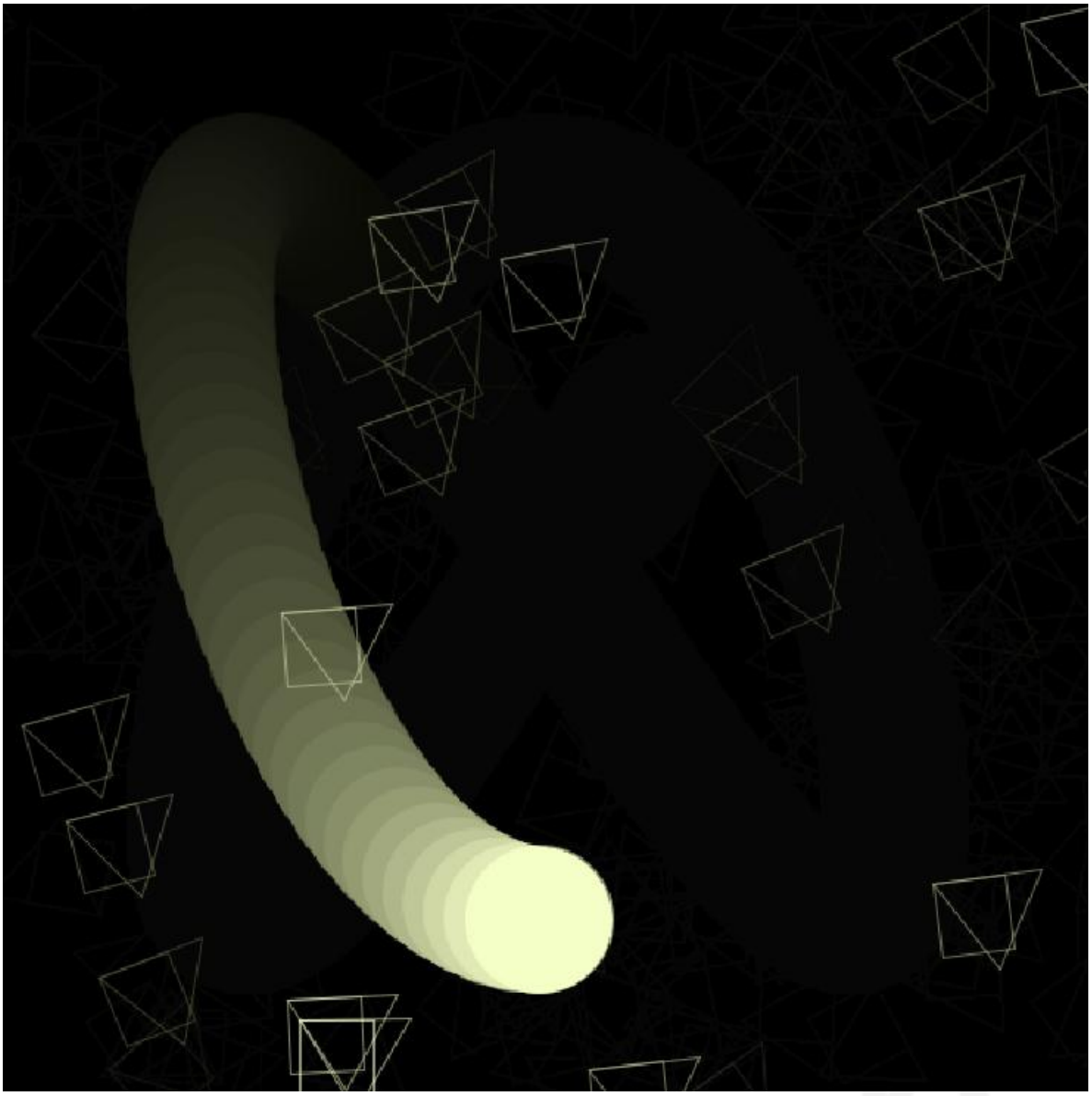
NOTA: Segunda opción de una de las posiciones y colores variantes.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Integración de movimiento con mouse y aplicación de figuras geométricas.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



Nota: Ejecución de algoritmo de la figura 15.

Resultados esperados

Al finalizar la actividad, el estudiante logrará generar un sketch en Processing capaz de expresar visualmente el concepto de *vibe coding* mediante animaciones fluidas y transiciones cromáticas. Se espera que aplique funciones de dibujo dinámico, manipulación de color y movimiento basado en ondas sinusoidales para crear una composición estética coherente. El resultado debe evidenciar control sobre el tiempo, la variación rítmica y la relación entre forma y color, mostrando una comprensión práctica del comportamiento animado dentro de Processing.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

3.2 Practica experimental 8

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Investigación de sintaxis básica para escribir "Hola Mundo" en Java

Objetivos

Identificar la estructura fundamental de un programa Java simple a través del análisis de la sintaxis necesaria para imprimir un mensaje en consola.

Antecedentes/Escenario

Java es un lenguaje orientado a objetos cuya estructura se basa en clases y métodos. Incluso un programa básico como "Hola Mundo" requiere comprender elementos esenciales como clases, métodos estáticos y sentencias de impresión.

Recursos necesarios

Editor de texto o IDE (NetBeans, IntelliJ, VSCode).

Acceso a fuente de documentación oficial de Java.

Planteamiento del problema

Se requiere analizar la sintaxis mínima necesaria para que un programa escrito en Java pueda ejecutarse correctamente y mostrar un mensaje básico en consola.

Pasos por realizar

Paso 1: Revisar estructura de un archivo .java.

Paso 2: Identificar elementos obligatorios.

Paso 3: Elaborar ejemplo funcional.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

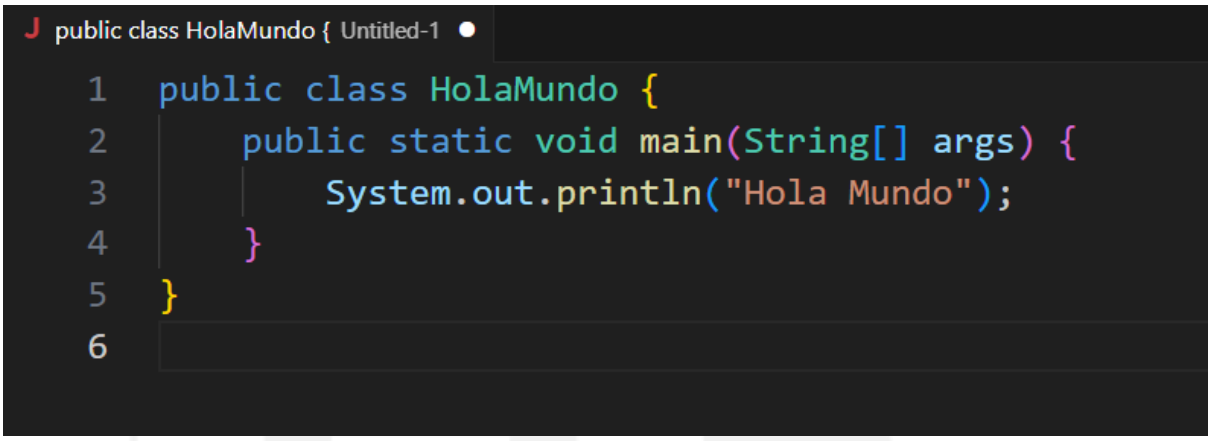
Desarrollo

Estructura analizada del programa Java mínimo:

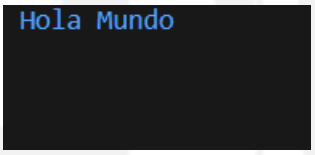
Declaración de clase pública.

Método principal main para iniciar la ejecución.

Uso de System.out.println() para imprimir texto.



NOTA: Algoritmo para la escritura de Hola mundo realizado en Visual Studio Code.



NOTA: Ejecución de algoritmo.

Elementos identificados:

Clase: estructura base del archivo.

Método main: punto de entrada.

Sentencia println: envía texto a consola.

Resultados esperados

Al finalizar la actividad, el estudiante comprenderá la estructura mínima que requiere un programa en Java y reconocerá los elementos esenciales para ejecutar instrucciones básicas. Se espera que identifique correctamente el rol de la clase, el método main y la instrucción de salida en consola, desarrollando así una base sólida para continuar con programas más complejos. Además, adquirirá una familiaridad inicial con el entorno de desarrollo, reforzando su capacidad para interpretar y escribir código simple en Java.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

3.3 Practica experimental 9

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Creación de un algoritmo en Java que solicite una contraseña y entre en un ciclo hasta obtener la correcta

Objetivos

Desarrollar un código funcional que implemente una estructura repetitiva para validar una contraseña ingresada por el usuario.

Antecedentes/Escenario

La verificación de contraseñas es un comportamiento común en programación. Su implementación requiere el uso de ciclos y comparaciones para validar repetidamente hasta cumplir la condición correcta.

Recursos necesarios

IDE o compilador Java.
Consola de entrada/salida.

Planteamiento del problema

Construir un programa que pida la contraseña repetidamente hasta que el usuario introduzca el valor correcto, empleando un ciclo adecuado.

Pasos por realizar

Declarar contraseña válida.

Crear un ciclo.

Comparar entrada con valor correcto.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Desarrollo

```java
import java.util.Scanner;

public class PasswordLoop {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        String correcta = "perrito123";
        String ingreso = "";

        while(!ingreso.equals(correcta)) {
            System.out.print("Ingrese contraseña: ");
            ingreso = sc.nextLine();
        }

        System.out.println("Acceso concedido.");
    }
}
```

NOTA: Implementación de algoritmo para ciclo

```
Output

Ingrese contraseña: 20
Ingrese contraseña: 20
Ingrese contraseña: 30
Ingrese contraseña: perrito123
Acceso concedido.

=== Code Execution Successful ===
```

NOTA: Ejecución de algoritmo de figura 19.

Resultados esperados

Al finalizar la actividad, el estudiante será capaz de implementar un algoritmo en Java que utilice correctamente un ciclo de repetición para validar una contraseña. Se espera que comprenda cómo comparar valores ingresados por el usuario, controlar la ejecución del ciclo hasta cumplir una

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

condición específica y mostrar retroalimentación adecuada en consola. Asimismo, fortalecerá su dominio de estructuras repetitivas y su aplicación en procesos de autenticación básicos, desarrollando una base sólida para futuras prácticas de validación y seguridad elemental.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

4 UNIDAD 3: Integración de Algoritmos Gráficos en Aplicaciones Multimedia

4.1 Practica experimental 10

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Creación de un canvas en Processing con una elipse capaz de colisionar con otro elemento

Objetivos

Implementar un canvas en Processing que dibuje una elipse e incluya detección de colisión contra el mouse o contra un objeto adicional.

Antecedentes

Processing permite programar colisiones básicas comparando distancias o intersecciones geométricas. Este ejercicio introduce conceptos esenciales para videojuegos y simulaciones.

Recursos necesarios

Processing instalado.

Planteamiento del problema

Crear una elipse y detectar cuándo esta colisiona ya sea con el puntero o con un rectángulo dentro del canvas.

Pasos por realizar

Dibujar elementos principales.

Implementar detección de colisión.

Cambiar estado visual al colisionar.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

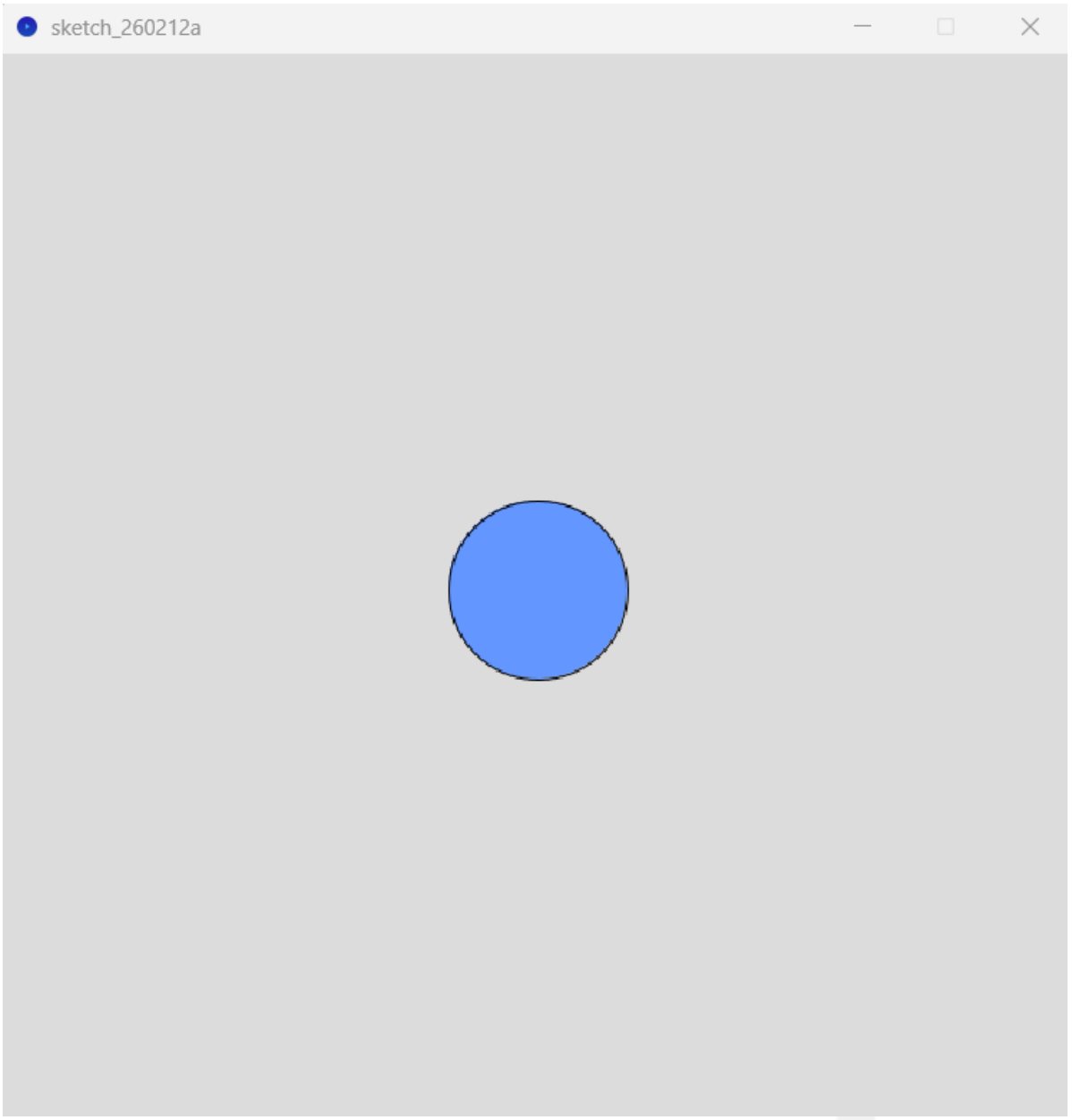
Desarrollo

sketch 260212a

```
void setup(){
  size(600,600);
}

void draw(){
  background(220);

  // elipse
  fill(100,150,255);
  ellipse(x, y, r*2, r*2);

  // colisión con mouse (por distancia)
  float d = dist(mouseX, mouseY, x, y);
  if(d < r){
    fill(255,0,0);
    ellipse(x, y, r*2, r*2);
  }
}
```

NOTA: Algoritmo inicial en Processing

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Algoritmo Ejecutado

```
float boxX = 0;
float boxY = 0;
float boxZ = 0;
float velocidad = 5;

float sphereX = 0;
float sphereY = 0;
float sphereZ = 0;

float sphereR = 50;
float boxSize = 80;

```


(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

NOTA: Declaración de variables para integrar segundo objeto

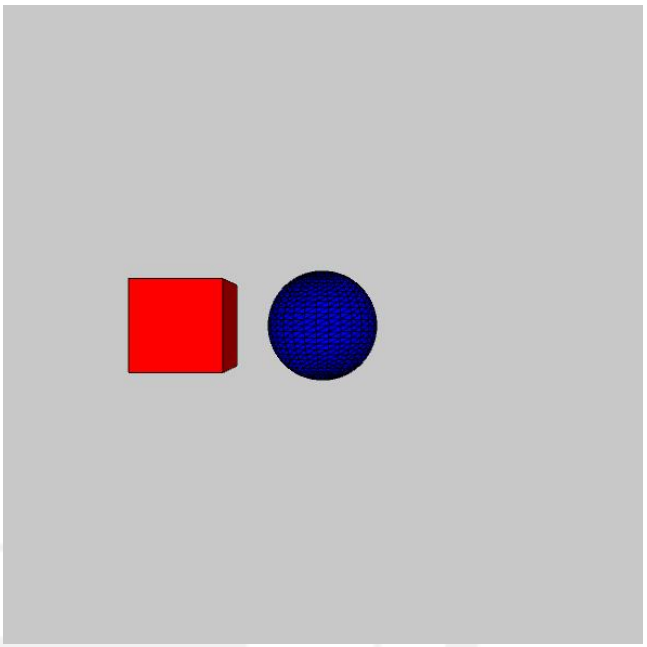
```
void setup(){
  size(600, 600, P3D);
}

void draw(){
  background(200);
  lights();

  translate(width/2, height/2, 0);

  if (keyPressed) {
    if (key == 'w' || key == 'W') boxY -= velocidad;
    if (key == 's' || key == 'S') boxY += velocidad;
    if (key == 'a' || key == 'A') boxX -= velocidad;
    if (key == 'd' || key == 'D') boxX += velocidad;
  }
```

NOTA: Implementación de movimiento con WASD para el segundo objeto.

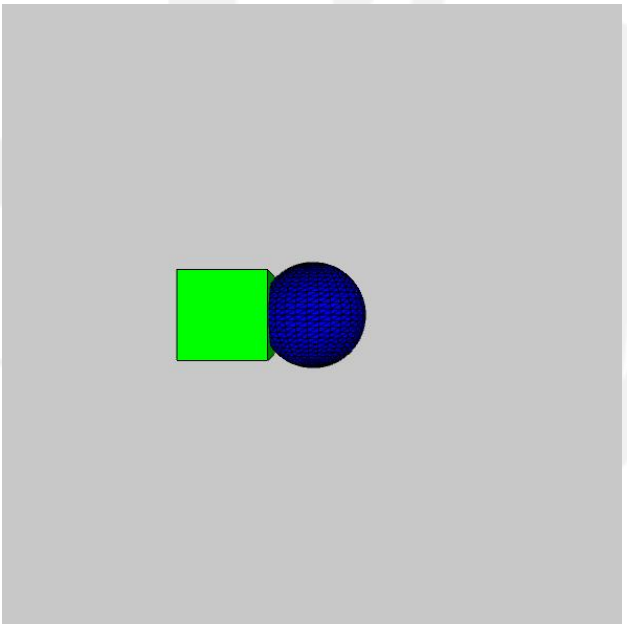
```
  float d = dist(boxX, boxY, boxZ, sphereX, sphereY, sphereZ);
  boolean colision = d < (sphereR + boxSize/2);

  pushMatrix();
  translate(sphereX, sphereY, sphereZ);
  fill(0, 0, 255);
  sphere(sphereR);
  popMatrix();

  pushMatrix();
  translate(boxX, boxY, boxZ);

  if(colision){
    fill(0, 255, 0);
  } else {
    fill(255, 0, 0);
  }

  box(boxSize);
  popMatrix();
}
```

NOTA: Lógica para interacción y acción de colisión entre los dos objetos.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Ejecución de Algoritmo sin colisión.



NOTA: Ejecución de Algoritmo con colisión.

Resultados esperados

Al concluir la actividad, el estudiante será capaz de construir un canvas funcional en Processing que incluya una elipse con detección de colisiones, ya sea frente al puntero del mouse o frente a otro objeto presente en pantalla. Se espera que el estudiante comprenda los principios básicos de intersección geométrica, gestione cambios visuales cuando ocurre una colisión y refuerce habilidades fundamentales para el desarrollo de videojuegos y simulaciones interactivas. Además, logrará

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

relacionar la representación gráfica con la lógica programada, fortaleciendo su capacidad para integrar visualización y comportamiento algorítmico.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

4.2 Practica experimental 11

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Canvas interactivo con cambio de color según posición del mouse y colisión con un cuadrado y un círculo

Objetivo

Construir un entorno interactivo en Processing donde el color del fondo o elementos varíen según la posición del mouse y donde existan colisiones con dos figuras.

Antecedentes/Escenario

El uso de entrada del mouse es esencial en interfaces interactivas, herramientas creativas y videojuegos. Las colisiones múltiples fortalecen la comprensión de detección espacial.

Recursos necesarios

Processing

Planteamiento del problema

Crear un canvas dinámico que responda a la posición del mouse y que evalúe colisiones con un cuadrado y un círculo ubicados en posiciones fijas.

Pasos por realizar

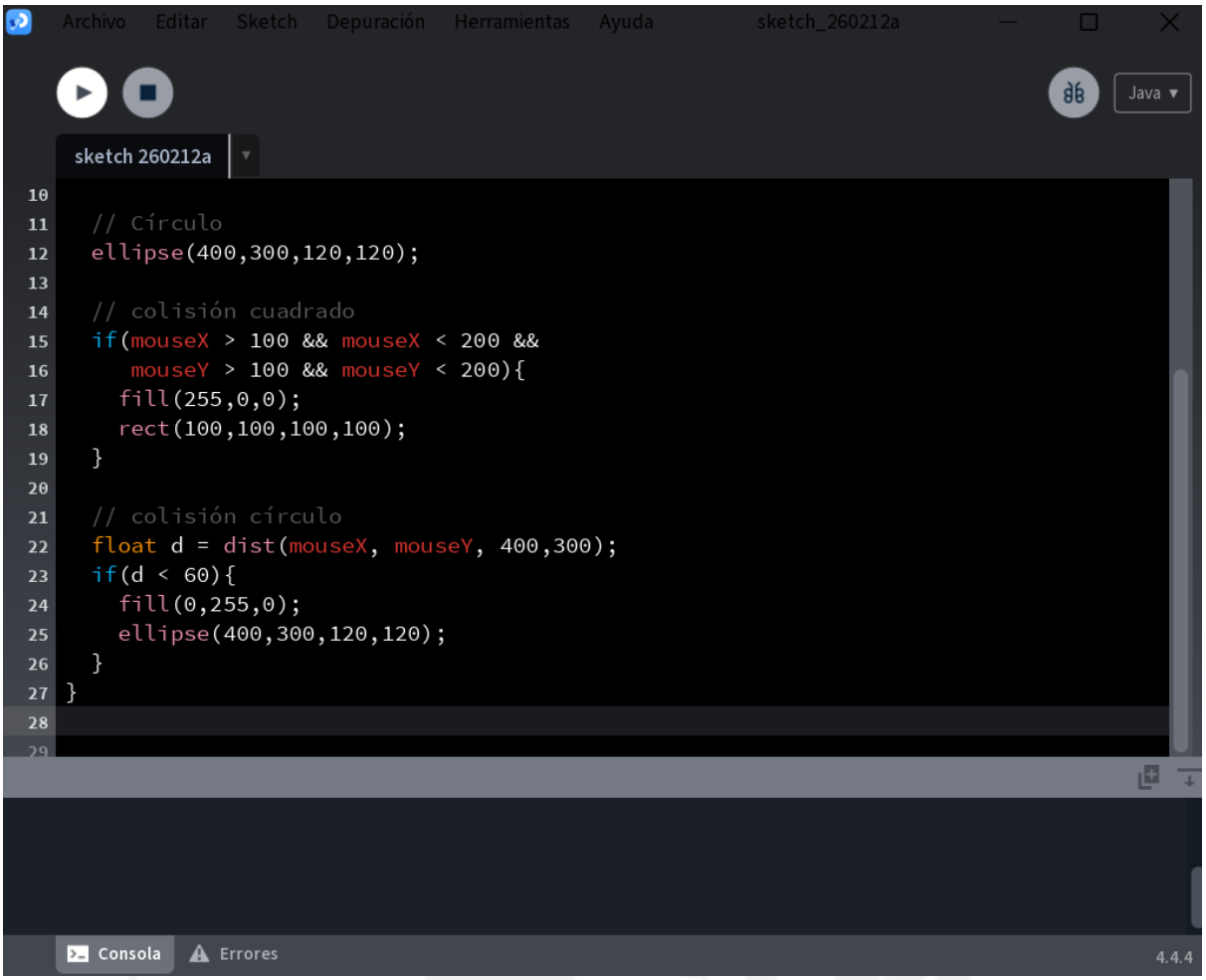
Dibujar figuras.

Evaluar interacciones.

Cambiar color al detectar colisiones.

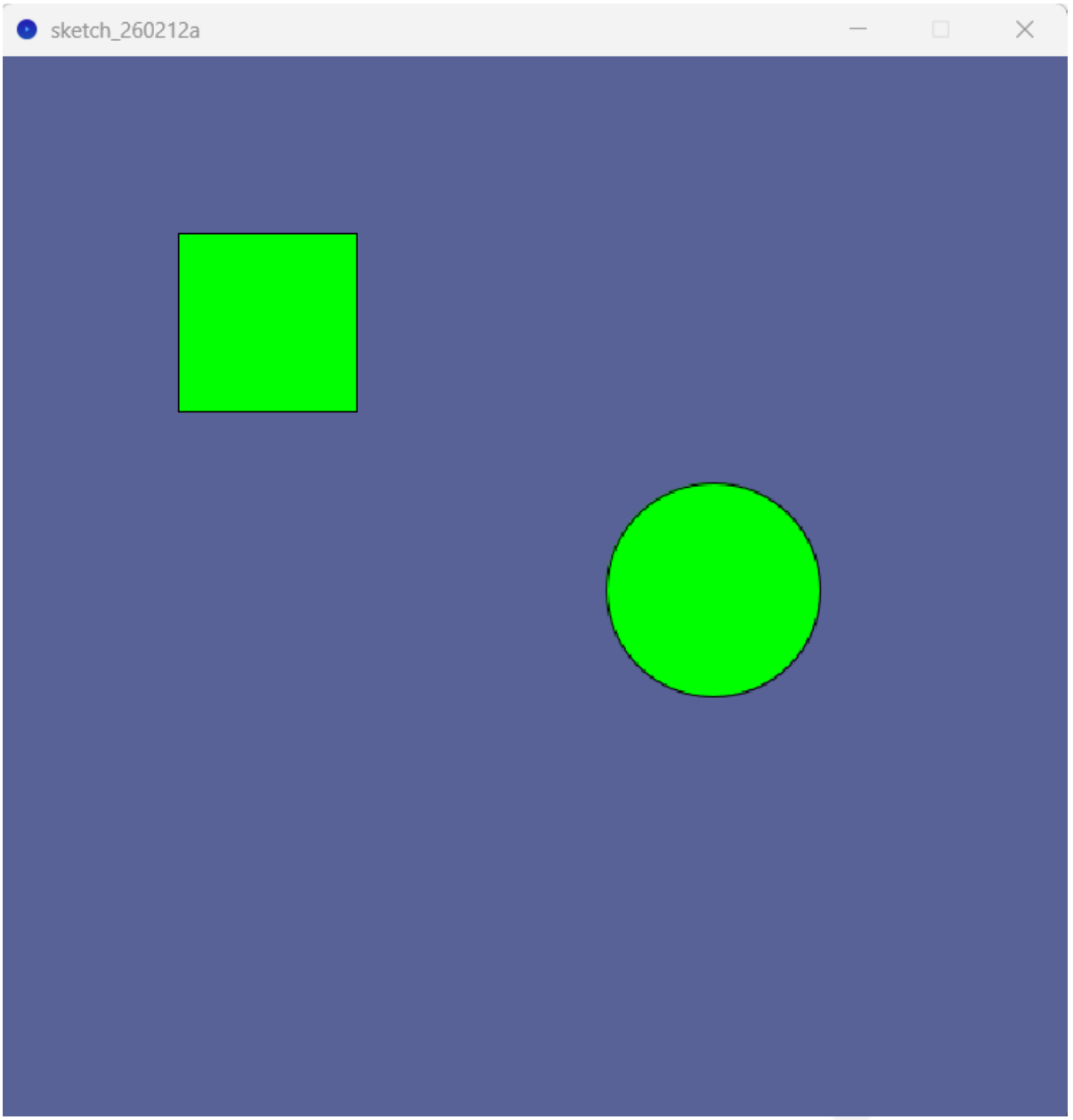
(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Desarrollo



NOTA: Creación de Algoritmo para determinar colisiones

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Verificación de algoritmo.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

```
float rotacion = 0;

void setup() {
  size(600, 600, P3D);
}

void draw() {
  background(mouseX % 255, mouseY % 255, 150);
  lights();

  boolean enEsquina = (mouseX < 150 && mouseY < 150) || (mouseX > 450 && mouseY < 150) ||
                      (mouseX < 150 && mouseY > 450) || (mouseX > 450 && mouseY > 450);
```

NOTA: Creación lógica para determinar posición de mouse y sus respectivas colisiones.

```
dibujarCubo(100, 100);
dibujarCubo(500, 100);
dibujarCubo(100, 500);
dibujarCubo(500, 500);

float d = dist(mouseX, mouseY, width/2, height/2);

pushMatrix();
translate(width/2, height/2, 0);
rotateY(rotacion);

if (d < 60) {
  fill(random(255), random(255), random(255));
} else {
  fill(100, 150, 255);
}
```

***NOTA*:** Establece la posición fija de los cuatro cubos en las esquinas del lienzo y define la esfera central. Incluye una condicional basada en la distancia Euclidiana para activar el cambio de color aleatorio cuando el cursor se posiciona sobre el elemento central.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

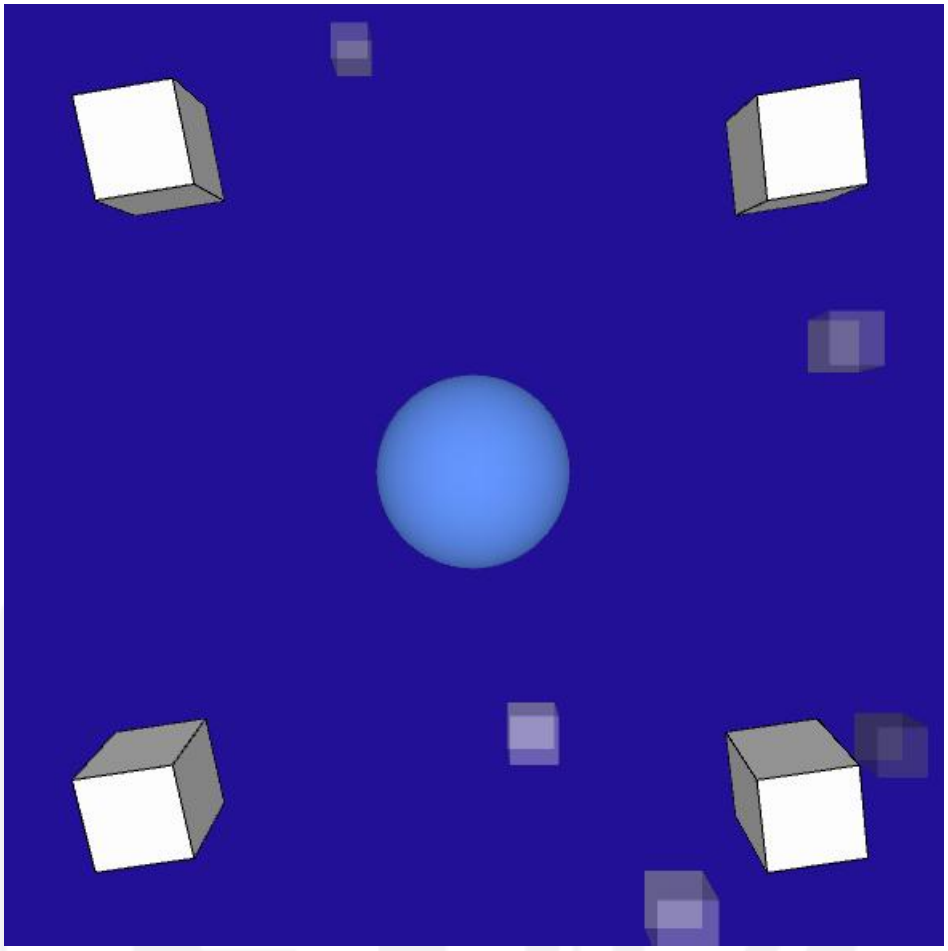
```
  noStroke();
  sphere(60);
  popMatrix();

  rotacion += 0.02;
}

void dibujarCubo(float x, float y) {
  pushMatrix();
  translate(x, y, 0);
  rotateX(rotacion);
  rotateY(rotacion);

  if (dist(mouseX, mouseY, x, y) < 50) {
    fill(255, 0, 0);
  } else {
    fill(255);
  }

  stroke(0);
  box(60);
  popMatrix();
}
```

NOTA: Define la renderización de la esfera central sin bordes y la función personalizada dibujarCubo, la cual gestiona la posición, rotación continua y el cambio de color por proximidad del puntero en el espacio 3D.

```
void crearElementosFondo() {
  for (int i = 0; i < 5; i++) {
    pushMatrix();
    translate(random(width), random(height), random(-200, 0));
    fill(random(255), 100);
    noStroke();
    box(random(20, 50));
    popMatrix();
  }
}
```

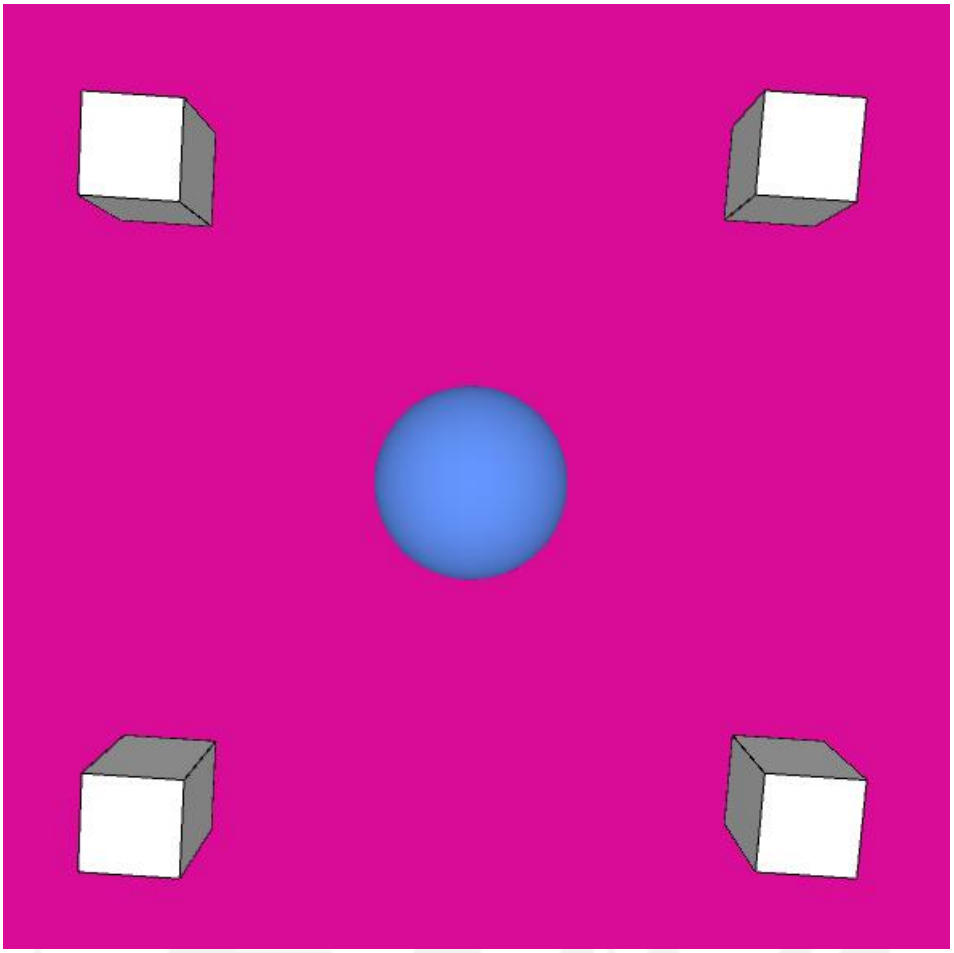
NOTA: Función crearElementosFondo que utiliza un bucle for para generar 5 cubos de tamaño aleatorio en posiciones aleatorias del eje Z. Implementa transparencia en el relleno (fill) para crear un efecto de profundidad decorativo cuando el mouse interactúa con las esquinas.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Ejecución de algoritmo sin acción.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Ejecución de algoritmo con acción.

Resultados esperados

Se espera que el estudiante logre construir un canvas interactivo en Processing capaz de reaccionar dinámicamente al movimiento del mouse y de evaluar colisiones independientes con un cuadrado y un círculo. Al finalizar la actividad, el estudiante comprenderá cómo modificar colores y estados visuales en función de la posición del cursor, aplicará correctamente conceptos de detección espacial mediante distancias y límites geométricos, y consolidará habilidades esenciales para el diseño de sistemas interactivos. El ejercicio permitirá reforzar la relación entre entrada del usuario, lógica de colisión y respuesta visual en tiempo real.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

4.3 Practica experimental 12

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Implementación de una imagen con declaración de variables y métodos para uso en el proyecto

Objetivos

Incorporar una imagen en Processing y estructurar correctamente variables y métodos que gestionen su comportamiento.

Antecedentes/Escenario

La inclusión de imágenes prepara el terreno para sprites, fondos y elementos visuales de videojuegos. Su correcto manejo requiere declarar variables globales y métodos de dibujo.

Recursos necesarios

Processing
Archivo png, jpg

Planteamiento del problema

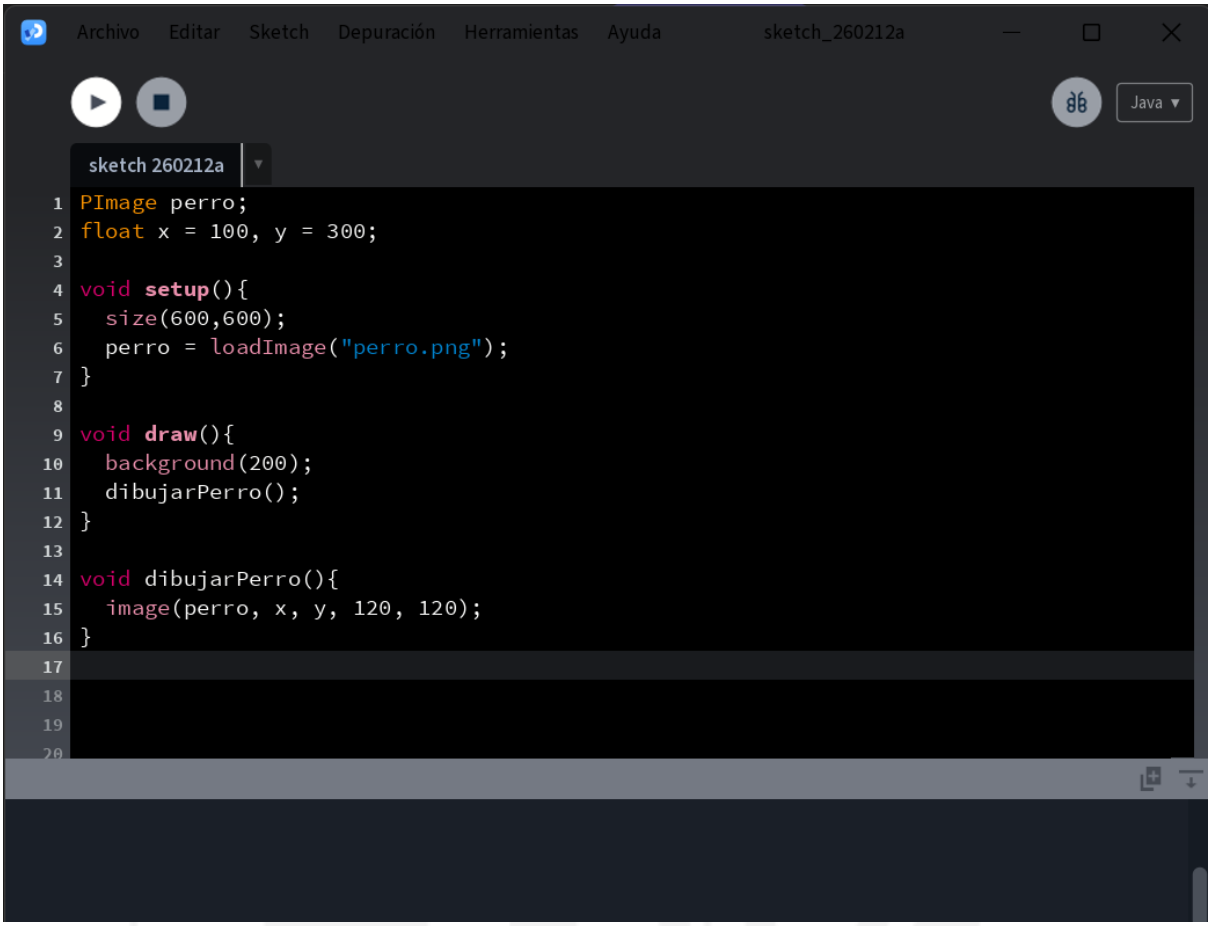
Integrar una imagen en un sketch y definir métodos que manipulen su posición, escala.

Pasos por realizar

Declarar variable de imagen.
Cargar archivo.
Crear método para dibujar.

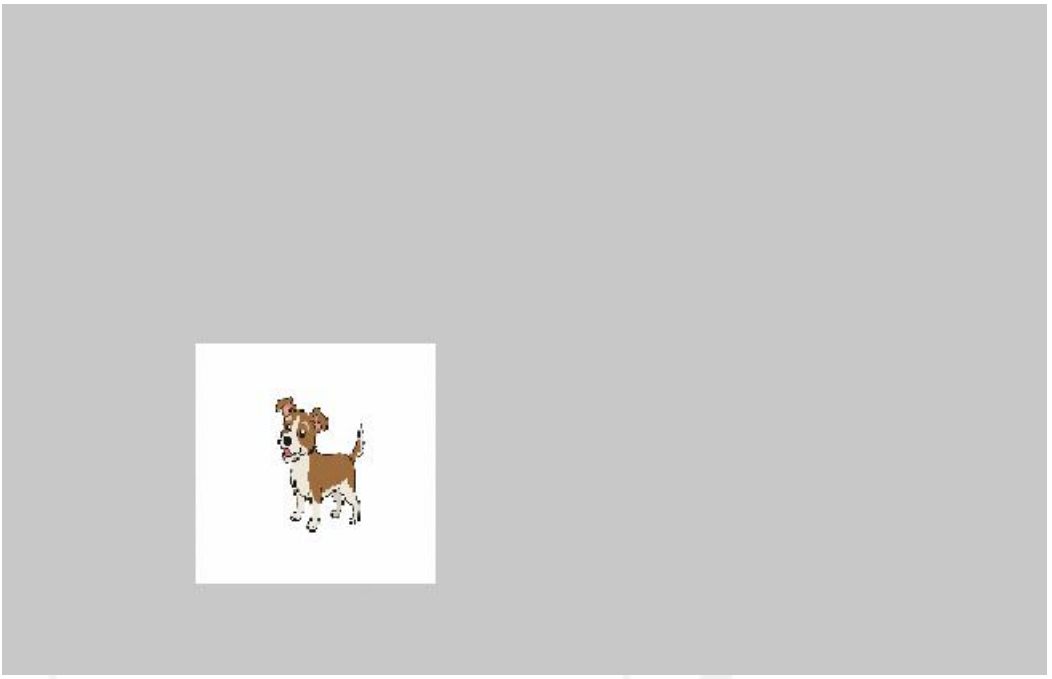
Desarrollo

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Implementación de una imagen en el programa.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Ejecución de algoritmo.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

```
PImage perro;
float x = 100, y = 300;
float velocidadY = 0;
float gravedad = 0.6;
float salto = -12;
float velocidadX = 5;
```

NOTA: Declaración de variables.

```
void draw(){
  background(200);

  velocidadY += gravedad;
  y += velocidadY;

  if (y > height - 120) {
    y = height - 120;
    velocidadY = 0;
  }

  if (keyPressed) {
    if (key == 'a' || key == 'A') x -= velocidadX;
    if (key == 'd' || key == 'D') x += velocidadX;
  }
```

NOTA: Ciclo principal que aplica física de gravedad y colisión con el suelo sobre el eje Y, permitiendo además el desplazamiento lateral del personaje mediante la detección de teclas activas

```
  dibujarPerro();
}

void keyPressed() {
  if ((key == ' ' || key == 'w' || key == 'W') && y >= height - 121) {
    velocidadY = salto;
  }
}

void dibujarPerro(){
  image(perro, x, y, 120, 120);
}
```


(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

NOTA: Implementación del evento de teclado para ejecutar el salto solo cuando el personaje detecta contacto con el suelo, junto con la función encargada de renderizar la imagen en las coordenadas actualizadas.

Resultados esperados

Al finalizar la actividad, el estudiante será capaz de integrar correctamente una imagen en un sketch de Processing mediante la declaración de variables globales y el uso de métodos dedicados para su administración. Se espera que el estudiante comprenda cómo cargar archivos gráficos externos, manipular su posición y controlar su representación dentro del canvas. Además, dominará la estructura modular del código mediante funciones específicas, fortaleciendo bases esenciales para la implementación de sprites, animaciones y elementos visuales interactivos en futuros proyectos de programación y videojuegos.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

4.4 Practica experimental 13

Cargo:	Docente Especialista		
Nombre:	Alexis Urgilés		
Asignatura:	Programación		
Carrera:	Alexis Urgilés	**Nivel:**	Segundo Nivel
Estudiante:			

ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO

Implementación de sonido en Processing con métodos de reproducción y pausa

Objetivos

Integrar un archivo de sonido y controlarlo mediante métodos que permitan iniciar, detener o pausar su reproducción.

Antecedentes

Processing permite reproducir sonido utilizando bibliotecas externas como Minim. Su manejo es clave para videojuegos, animaciones interactivas y simulaciones audiovisuales.

Recursos necesarios

Processing.

Biblioteca Minim.

Archivo MP3 o WAV.

Planteamiento del problema

Incorporar sonido y controlarlo con métodos bien definidos.

Pasos por realizar

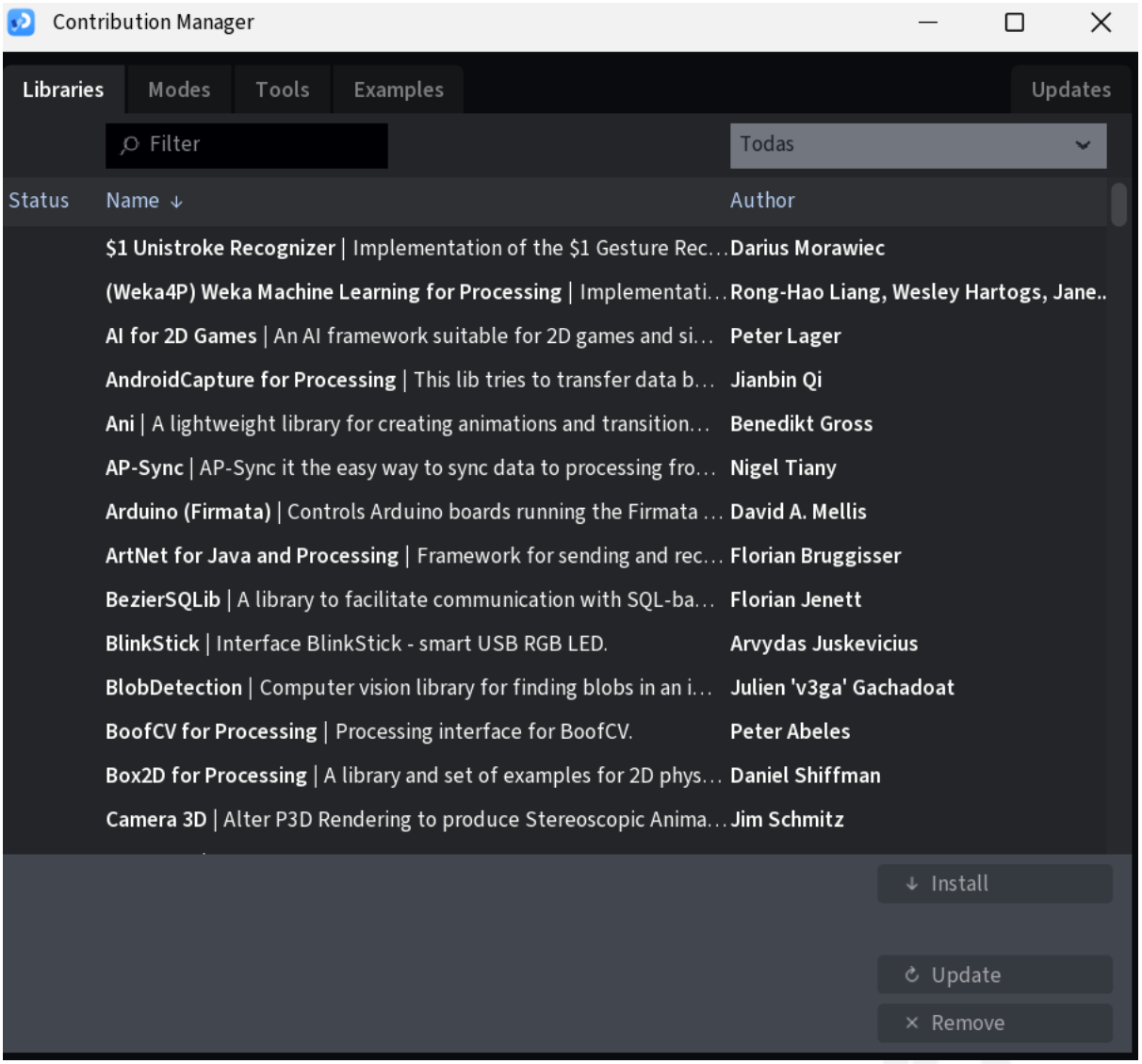
Importar biblioteca.

Cargar archivo.

Implementar botones lógicos de control.

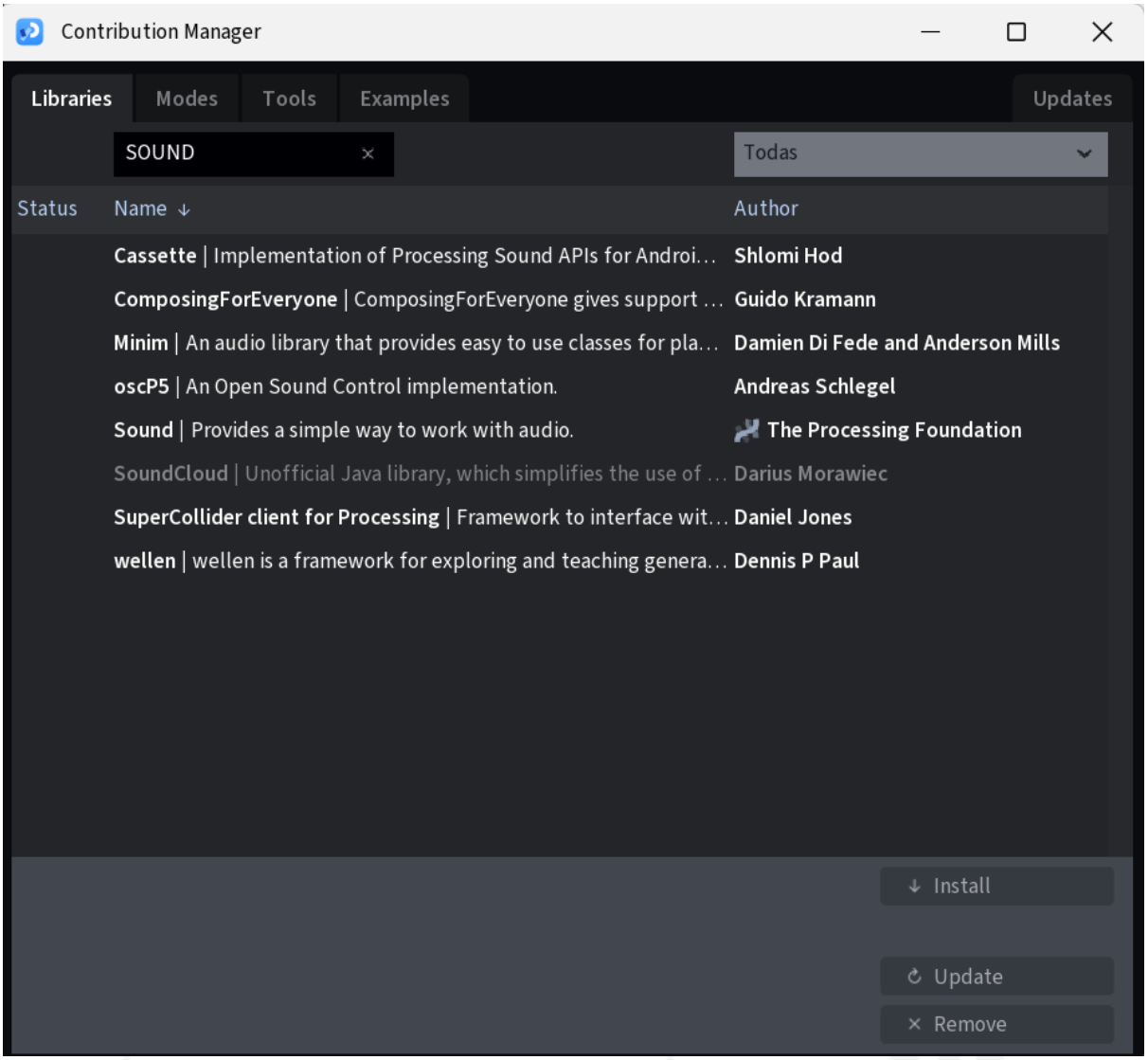
(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Desarrollo



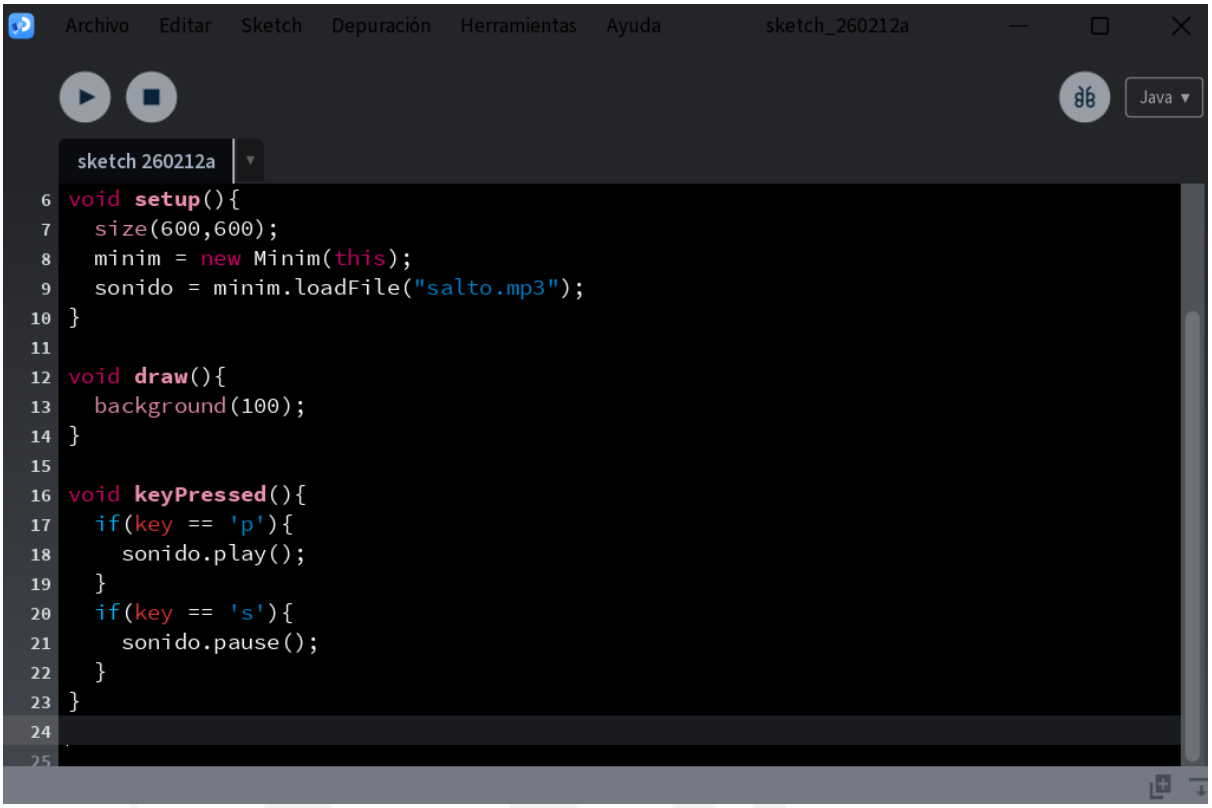
NOTA: Integración de bibliotecas

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Implementación de la biblioteca de sonidos.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: implementación del código par el funcionamiento correcto del sonido.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

```
import ddf.minim.*;

Minim minim;
AudioPlayer sonido;

Jugador principal;
Obstaculo[] enemigos = new Obstaculo[5];

void setup() {
  size(600, 600);
  minim = new Minim(this);
  sonido = minim.loadFile("salto.mp3");

  principal = new Jugador();


  for (int i = 0; i < enemigos.length; i++) {
    enemigos[i] = new Obstaculo();
  }
}
```

NOTA: Configuración inicial del programa donde se importan las librerías de audio, se cargan los recursos sonoros y se instancian tanto el objeto del jugador como un arreglo de cinco obstáculos mediante un bucle iterativo.

```
void draw() {
  background(50);

  principal.actualizar();
  principal.dibujar();

  for (int i = 0; i < enemigos.length; i++) {
    enemigos[i].caer();
    enemigos[i].dibujar();


    float d = dist(principal.x, principal.y, enemigos[i].x, enemigos[i].y);
    if (d < 40) {
      if (!sonido.isPlaying()) {
        sonido.rewind();
        sonido.play();
      }
      fill(255, 0, 0, 100);
      rect(0, 0, width, height);
    }
  }
}
```


(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

NOTA: Ciclo principal que actualiza y dibuja al jugador junto con un arreglo de obstáculos en movimiento constante. Incluye la detección de colisiones por distancia para activar el sonido de alerta y un efecto visual rojo en pantalla.

```
class Jugador {
  float x, y;

  Jugador() {
    x = width/2;
    y = height/2;
  }

  void actualizar() {
    x = mouseX;
    y = mouseY;
  }

  void dibujar() {
    fill(0, 255, 150);
    ellipse(x, y, 40, 40);
  }
}
```

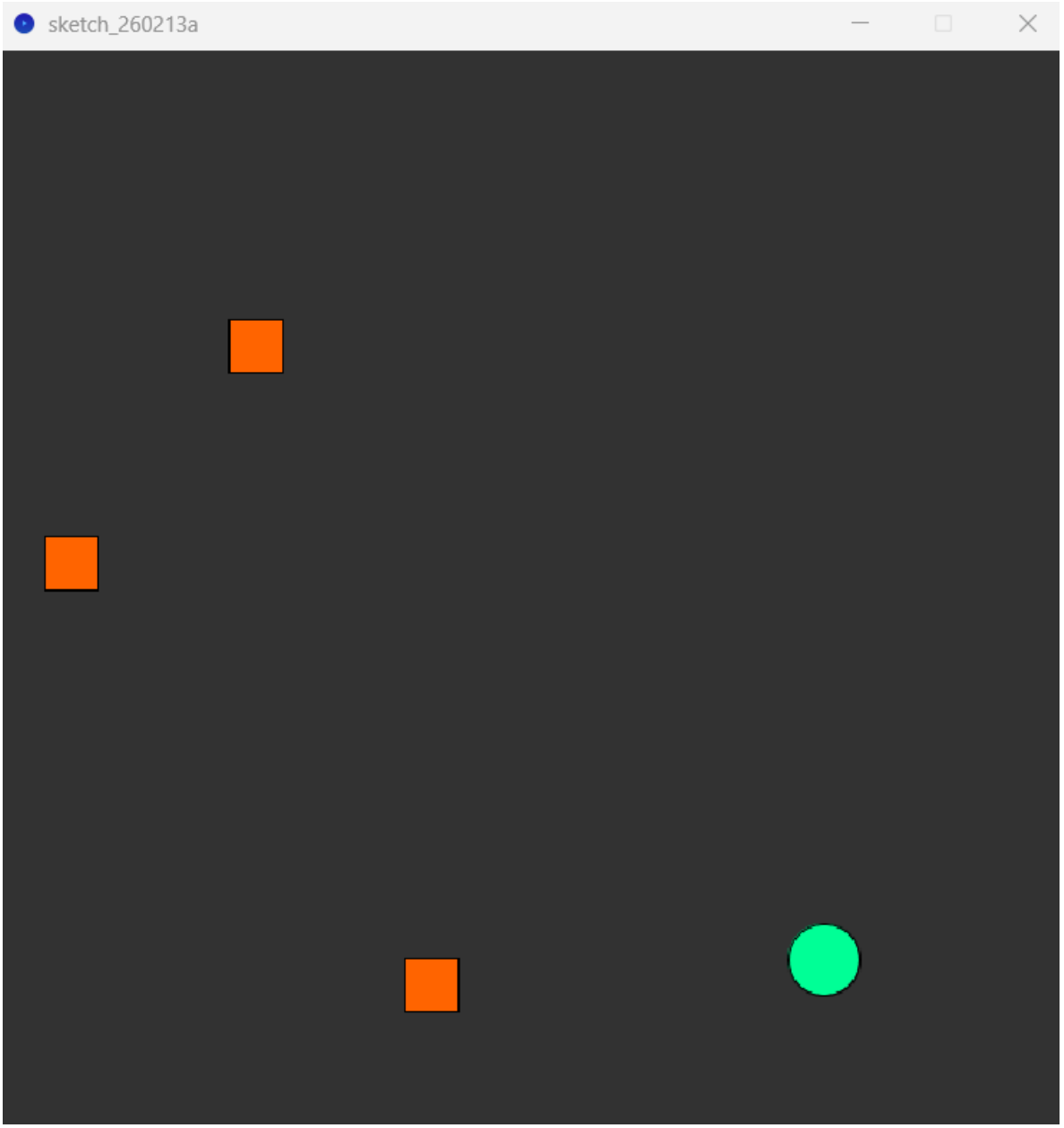
NOTA: Definición de la clase Jugador que establece su posición inicial en el centro del lienzo. Incluye métodos para sincronizar sus coordenadas con el mouse y renderizar su forma geométrica.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

```
class Obstaculo {
  float x, y, velocidad;

  Obstaculo() {
    reset();
  }

  void reset() {
    x = random(width);
    y = random(-500, -50);
    velocidad = random(2, 5);
  }

  void caer() {
    y += velocidad;
    if (y > height) {
      reset();
    }
  }

  void dibujar() {
    fill(255, 100, 0);
    rect(x, y, 30, 30);
  }
}
```

NOTA: Clase que gestiona el comportamiento de los enemigos, asignándoles posiciones y velocidades aleatorias mediante un método de reinicio. Define la lógica de caída vertical continua y el reciclaje del objeto cuando este supera el límite inferior del lienzo.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC



NOTA: Aplicación de algoritmo funcional.

Resultados esperados

Al concluir la actividad, el estudiante será capaz de integrar correctamente un archivo de sonido dentro de Processing utilizando la biblioteca Minim, aplicando métodos específicos para reproducir, pausar y detener el audio según sea necesario. Se espera que logre estructurar su código de forma modular mediante funciones dedicadas al control sonoro, comprendiendo el funcionamiento básico de los objetos de audio y su ciclo de vida. Asimismo, el estudiante adquirirá una base sólida para implementar efectos auditivos y sistemas de sonido en proyectos interactivos o videojuegos.

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC