# Tecnológico Nacional de México
# Instituto Tecnológico de Toluca

## Asignatura: Graficación

## Carrera: Ingeniería en Sistemas Computacionales

## Grupo: 184500

## Documento proyecto final

## Tema 5: Introducción a la animación por computadora

## Profesor: M. C. C. Rocío Elizabeth Pulido Alba

**Integrantes:**

**Esquivel Flores Jonathan 21280557**

**Garduño Rodríguez Jair 21281153**

**Millán Desales Raúl 21280585**

## Fecha de entrega: 17 de junio de 2024

## Índice


- Índice ..... 2
- Introducción .....4
- Planteamiento del problema.....5
  - Delimitación
  - Justificación
  - Objetivo general
  - Objetivos específicos
- Marco teórico ..... 7
  - Características del área de estudio
  - Soporte científico
  - Estadísticas mundiales al respecto
  - Videojuegos como herramienta de aprendizaje
  - Herramientas y entorno de desarrollo
    - Matrices 3D
    - OpenGL Python 3D
    - Texturización
    - Manejo de colisiones
    - Manejo de cámara
    - Sonido
    - Análisis FODA
- Marco metodológico .....20
  - Requerimientos funcionales
  - Requerimientos no funcionales
  - Ciclo de vida del software
  - Diagrama de flujo de actividades
  - Diagrama de caso de uso
  - Plantilla de caso de uso
  - Diagrama de paquetes

- Desarrollo de solución ..... 27
  - Historia del juego
  - Manera de juego
  - Descripción de los personajes
  - Personajes en 3D
  - Niveles por personaje
  - Como muestra la ayuda
  - Como muestra los créditos
  - Código relevante explicado
- Conclusiones.....34
- Referencias bibliográficas ..... 36
- Anexos ..... 37
  - Bocetos
  - Retroalimentación en clase tras la primera prueba
  - Revisiones de nivel

## Introducción

"Trash's Collector" es un apasionante juego de video que te transporta a un mundo lleno de retos y emociones, donde tu principal objetivo es limpiar y recolectar desechos en variados entornos, tanto urbanos como naturales. Desarrollado completamente en Python, este juego hace uso exhaustivo de sus propias librerías para la representación gráfica por computadora, ofreciendo así una experiencia visualmente impactante y una jugabilidad inmersiva.

En "Trash's Collector", te conviertes en un intrépido recolector de basura, asignado a la tarea de limpiar diversos escenarios plagados de residuos. Desde bulliciosas calles hasta exuberantes parques naturales, cada nivel te presenta nuevos desafíos y obstáculos que pondrán a prueba tus habilidades y destrezas.

El juego aprovecha al máximo las capacidades gráficas de Python, haciendo uso de sus librerías especializadas para crear entornos detallados y realistas. Los efectos visuales, que van desde la textura de la basura hasta la iluminación ambiental, están meticulosamente diseñados para sumergirte por completo en la experiencia de limpiar y recolectar.

Además de la impresionante calidad visual, "Trash's Collector" ofrece una jugabilidad adictiva y desafiante. Con controles intuitivos y mecánicas de juego fluidas, te enfrentarás a una amplia variedad de tareas de recolección de basura, como clasificar desechos, resolver acertijos ambientales y superar obstáculos mientras avanzas a través de los distintos niveles.

La historia de "Trash's Collector" te embarca en un viaje emocionante y educativo sobre la importancia de cuidar el medio ambiente y mantener nuestro mundo limpio. A medida que progresas en el juego, aprenderás acerca de la gestión de residuos, la relevancia del reciclaje y cómo pequeñas acciones pueden tener un impacto significativo en la preservación del medio ambiente.

Con su combinación única de acción, aprendizaje y conciencia ambiental, "Trash's Collector" no solo proporciona una experiencia de juego emocionante, sino que también motiva a los jugadores a tomar medidas en la vida real para proteger nuestro planeta. ¡Prepárate para sumergirte en esta emocionante aventura de recolección de basura y descubrir el defensor ambiental que llevas dentro!

## Planteamiento del problema

Se busca realizar un videojuego de 3 niveles enfocados en la recolección de basura. El jugador deberá que recolectar las bolsas de basura en un límite de tiempo a través de todo el escenario, en caso de error el juego terminara ahí.

## Delimitación

El escenario del juego solo será el límite de la pantalla, el primer nivel contara con 8 bolsas de basura a recolectar y cada nivel incluirá un porcentaje mas de bolsas que el nivel anterior decrementando el limite de tiempo.

## Justificación

En las últimas décadas debido a la producción en masa de productos envasados en plásticos y diferentes contenedores desechables, la producción de basura se ha vuelto un problema de índole mundial, ya que, al no llevar a cabo una correcta clasificación de estos desechos, comúnmente se encuentra en cualquier lugar: jardines, suelo, ríos, etc. .Nuestro videojuego tiene como objetivo hacer conciencia de este problema además de que se les dará noción de los desechos que se genera en tres diferentes entornos.

## Objetivo General

El objetivo general de nuestro videojuego es sensibilizar a los jugadores sobre el problema global de la producción masiva de basura y la importancia de una correcta clasificación de los desechos. A través de la experiencia interactiva de clasificar basura en tres entornos diferentes (jardín, salón de clases y fábrica), buscamos educar y concienciar a los jugadores sobre la necesidad de tomar medidas concretas para gestionar los desechos de manera responsable y preservar el medio ambiente.

## Objetivos específicos

- Proporcionar a los jugadores conocimientos sobre la variedad de desechos presentes en entornos habituales como jardines, aulas y fábricas.
- Generar conciencia entre los jugadores sobre las consecuencias adversas de la producción descontrolada de basura y la falta de su adecuada clasificación, tanto en el medio ambiente como en la salud humana.
- Promover entre los jugadores la adopción de prácticas responsables en la gestión de residuos, como el reciclaje y la minimización de desechos, mediante una experiencia de juego interactiva y atractiva.
- Desarrollar habilidades de toma de decisiones rápidas y precisas en los jugadores, al enfrentar situaciones realistas de clasificación de basura dentro de límites de tiempo.

- Motivar a los jugadores a llevar a cabo acciones concretas en su vida diaria para abordar el problema de la basura, como participar en actividades de limpieza comunitaria o adoptar comportamientos de consumo más sostenibles.

# Marco teórico

## Características del área de estudio

El proyecto se encuentra inmerso en la elaboración de una serie de videojuegos educativos, los cuales están diseñados con el propósito específico de sensibilizar y educar sobre la importancia fundamental del reciclaje y la correcta separación de residuos. Estos juegos fusionan de manera ingeniosa y creativa elementos lúdicos con mensajes pedagógicos, lo que les permite ofrecer a los jugadores una experiencia interactiva sumamente enriquecedora y didáctica.

A través de estos videojuegos, se busca no solo captar la atención y el interés del público, sino también transmitir de manera efectiva conocimientos y valores relacionados con la preservación del medio ambiente y la sostenibilidad. Desde desafíos que involucran la clasificación adecuada de diferentes tipos de residuos hasta la gestión responsable de recursos naturales, cada juego está cuidadosamente diseñado para ofrecer una experiencia de aprendizaje envolvente y significativa.

Al combinar la diversión propia de los videojuegos con la importancia de la educación ambiental, estos juegos representan una poderosa herramienta para fomentar conductas responsables y promover un mayor compromiso con la protección de nuestro entorno. En resumen, se trata de una iniciativa innovadora y relevante que aprovecha el potencial del entretenimiento digital para generar un impacto positivo en la sociedad y en el cuidado del planeta.

## Soporte científico

La investigación respalda ampliamente la idea de que los videojuegos pueden ser herramientas efectivas para el aprendizaje, especialmente en áreas como la educación ambiental. La interacción y la inmersión que caracterizan a los videojuegos pueden mejorar la retención de información y fomentar actitudes positivas entre los jugadores.

Los videojuegos permiten que los jugadores participen activamente en el proceso de aprendizaje, lo que puede aumentar su compromiso y motivación para absorber conocimientos. Además, la inmersión ofrecida por los juegos crea experiencias de aprendizaje envolventes y memorables, facilitando una comprensión más profunda de los temas tratados.

Otro aspecto relevante es que los videojuegos pueden influir en los comportamientos de los jugadores al presentar desafíos que requieren la aplicación de prácticas sostenibles, como el reciclaje o la conservación de la energía. La retroalimentación inmediata proporcionada por los juegos permite a los jugadores comprender las consecuencias de sus acciones y ajustar su comportamiento en consecuencia.

## Estadísticas mundiales al respecto

De acuerdo con estadísticas a nivel mundial, la cantidad de residuos generados por persona ha experimentado un notable aumento en las últimas décadas, lo que ha contribuido significativamente al problema de la contaminación ambiental. En este contexto, la educación ambiental, que incluye la promoción de prácticas como la separación adecuada de residuos, se vuelve fundamental para hacer frente a esta situación y fomentar comportamientos más sostenibles.

El incremento en la generación de residuos por persona ha ejercido una presión considerable sobre los ecosistemas naturales y los sistemas de gestión de residuos, exacerbando los problemas de contaminación del suelo, el agua y el aire. La educación ambiental desempeña un papel crucial al concienciar a las personas sobre el impacto de sus acciones en el medio ambiente y al promover cambios en su comportamiento para reducir la cantidad de residuos generados y gestionarlos de manera más responsable.

En particular, la separación adecuada de residuos es una práctica fundamental que puede ayudar a minimizar el impacto ambiental de los desechos. Al separar los residuos en diferentes categorías, como orgánicos, reciclables y residuos peligrosos, se facilita su posterior tratamiento y reciclaje, lo que contribuye a reducir la cantidad de desechos que terminan en vertederos o incineradoras.

## Videojuegos como herramienta de aprendizaje

Los videojuegos ofrecen un entorno interactivo y motivador que facilita el aprendizaje. Al combinar aspectos educativos con la jugabilidad, estos juegos ayudan a los jugadores a adquirir conocimientos y habilidades de manera efectiva y atractiva. La interactividad permite que los jugadores participen activamente en el proceso de aprendizaje, mientras que la jugabilidad envolvente garantiza una experiencia entretenida y estimulante. En resumen, los videojuegos representan una valiosa herramienta educativa que hace que el aprendizaje sea divertido y efectivo.

## Herramientas y entorno de desarrollo

Desarrollar un videojuego interactivo utilizando Visual Studio Code como entorno y Python como lenguaje de programación implica una serie de pasos y consideraciones específicas. En primer lugar, se comienza por diseñar la mecánica del juego y establecer los objetivos de aprendizaje que se desean alcanzar. Luego, se procede a la programación de las diferentes funciones y características del juego, como la interfaz de usuario, la lógica del juego, la gestión de eventos y la integración de elementos educativos.

Visual Studio Code ofrece un conjunto de herramientas poderosas para el desarrollo de software, incluyendo extensiones específicas para Python que facilitan la

escritura de código, el depurado y la gestión de proyectos. Además, su integración con sistemas de control de versiones como Git permite un desarrollo colaborativo y un seguimiento eficiente de los cambios realizados en el código.

En cuanto a Python, su sintaxis clara y legible lo convierte en una excelente opción para el desarrollo de videojuegos educativos, especialmente para aquellos que no requieren gráficos 3D avanzados. Python ofrece una amplia gama de bibliotecas y frameworks para el desarrollo de juegos, como Pygame o Arcade, que proporcionan herramientas para la creación de gráficos, sonido y física, así como para la gestión de entrada del usuario y la lógica del juego.

En comparación con el uso de motores gráficos como Unity o Unreal Engine, el desarrollo de un videojuego en Python con Visual Studio Code puede ofrecer una mayor flexibilidad y control sobre el proceso de desarrollo. Sin embargo, los motores gráficos suelen ofrecer características avanzadas para la creación de gráficos 3D, efectos visuales y optimización de rendimiento que pueden ser necesarios para ciertos tipos de juegos.

## Matrices 3D

Estas matrices se pueden presentar individualmente o de forma combinada (traslación, rotación, escalado o transformación de mundo).

En álgebra lineal, las transformaciones lineales pueden representarse con matrices. Generalmente se utilizan matrices 4x4 para representar transformaciones 3D:

$$\begin{bmatrix} m & m & m & m \\ m & m & m & m \\ m & m & m & m \\ m & m & m & m \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ \end{bmatrix}$$

Utilizamos matrices 4x4 en lugar de 3x3 porque algunas transformaciones especiales, como la traslación, requiere una columna/fila extra para realizarse correctamente.

Para permitir la multiplicación matriz-vector añadiremos un componente w al vector al que por ahora daremos el valor 1:

Estas matrices se utilizan comúnmente en computación grafica para transformaciones geométricas, como traslación, rotación, escalación, entre otras, que son esenciales para posicionar y darle forma a los objetos en 3D.

Las operaciones básicas en las matrices 3D incluyen la multiplicación de matrices para combinar transformaciones, como rotaciones y traslaciones a fin de enfocarse en los vértices del objeto para obtener su posición y orientación finales en el espacio 3D.

## OpenGL Python 3D

OpenGL es una API (Interfaz de Programación de Aplicaciones) gráfica multiplataforma utilizada para renderizar gráficos 2D y 3D en aplicaciones. Cuando

se combina con Python, especialmente con bibliotecas como PyOpenGL, se puede utilizar para crear gráficos tridimensionales de manera eficiente.

## Texturización

Texturizar modelos 3D es “darles vida” con colores, opacidades, brillos y otras variaciones en su superficie. Se trata de pasar de capas simples y uniformes —como las que se podrían pintar con un selector de relleno en Paint—, a otras que imiten la realidad o cuenten una historia.

Las texturas también tienen en cuenta aspectos como la suavidad o la rugosidad de un elemento, los reflejos que produce y, por supuesto, el “material” que lo compone. Se aplica tanto en el modelado orgánico 3D como en el modelado hard surface, y constituye una profesión en sí misma.

Dicho de forma simple: texturizar modelos 3D es añadirle detalles a un objeto tridimensional. El proceso tiene cuatro etapas, que van desde la elaboración de las texturas hasta el pulido del proyecto:

- **Confección de texturas:** lo primero es crear la textura que vas a aplicarle al objeto o personaje. Como veremos más adelante, puedes hacerlo tú mismo/a o dejar que se encarguen los programas de diseño 3D.
- **Aplicación del diseño:** cuando tenemos la capa con la que vamos a texturizar nuestros modelos 3D, hay que aplicarla. Lo que suele hacerse es mapear el elemento en cuestión para que cada detalle se visualice en la zona correcta.
- **Luces y sombras:** es el momento de llevar la iluminación a escena. Determina los puntos de los que procede y define cómo incide la luz en los objetos que estás texturizando.
- **Ultimación de los detalles:** puedes pintar directamente sobre los modelos 3D para texturizar los últimos detalles o añadir pequeñas rugosidades o abolladuras que le den más realismo.

## Manejo de colisiones

La elección de un método adecuado para resolver colisiones es tan importante como la elección de una buena función hash. Cuando la función hash obtiene una misma dirección para dos claves diferentes, se está ante una colisión.

Algunos métodos más utilizados para resolver colisiones son los siguientes:

• Reasignación
• Arreglos anidados
• Áreas de desborde

• **Reasignación**

Existen varios métodos que trabajan bajo el principio de comparación y reasignación de elementos. Se analizarán tres de ellos:

• Prueba lineal
• Prueba cuadrática
• Doble dirección hash

a) **Prueba lineal**
Consiste en que una vez detectada la colisión se debe de recorrer el arreglo secuencialmente a partir del punto de colisión, buscando al elemento. El proceso de búsqueda concluye cuando el elemento es hallado, o bien cuando se encuentra una posición vacía. Se trata al arreglo como a una estructura circular: el siguiente elemento después del último es el primero. La principal desventaja de este método es que puede haber un fuerte agrupamiento alrededor de ciertas claves, mientras que otras zonas del arreglo permanecerían vacías. Si las concentraciones de claves son muy frecuentes, la búsqueda será principalmente secuencial perdiendo así las ventajas del método hash.

b) **Prueba Cuadrática**
Este método es similar al de la prueba lineal. La diferencia consiste en que en el cuadrático las direcciones alternativas se generan como D + 1, D + 4, D + 9, . . ., D + $i^2$ en vez de D + 1, D + 2,...,D + i. Esta variación permite una mejor distribución de las claves colisionadas. La principal desventaja de este método es que pueden quedar casillas del arreglo sin visitar. Además, como los valores de las direcciones varían en $1^2$ unidades, resulta difícil determinar una condición general para detener el ciclo. Este problema podría solucionarse empleando una variable auxiliar, cuyos valores dirijan el recorrido del arreglo de tal manera que garantice que serán visitadas todas las casillas.

c) **Doble dirección hash**
Consiste en que una vez detectada la colisión se debe generar otra dirección aplicando la función hash a la dirección previamente obtenida. El proceso se detiene cuando el elemento es hallado, o bien cuando se encuentra una posición vacía: D = H(K), D' = H(D), D'' = H(D').
La función hash que se aplique a las direcciones puede o no ser la misma que originalmente se aplicó a la clave. No existe una regla que permita decidir cuál será la mejor función a emplear en el cálculo de las sucesivas direcciones.

### Arreglos anidados

Este método consiste en que cada elemento del arreglo tenga otro arreglo en el cual se almacena los elementos colisionados. Si bien la solución parece ser sencilla, es claro también que resulta ineficiente. Al trabajar con arreglos se depende del espacio que se allá asignado a este lo cual conduce a un nuevo problema difícil de solucionar: elegir un tamaño adecuado de arreglo que permita el equilibrio entre el coste de memoria y el número de valores colisionados que pudiera almacenar.

### Encadenamiento

Consiste en que cada elemento del arreglo tenga un apuntador a una lista ligada, la cual se ira generando e ira almacenando los valores colisionados a medida que se requiera.

## Manejo de cámara

El manejo de cámaras en OpenGL con Python en entornos 3D es esencial para controlar la vista y perspectiva de la escena renderizada. Aquí tienes una breve descripción del manejo de cámaras en este contexto:

**1. Posición y Orientación:** La cámara en OpenGL se representa mediante su posición y orientación en el espacio tridimensional. La posición determina desde dónde se observa la escena, y la orientación define hacia dónde apunta la cámara.

**2. Matrices de Vista (View Matrix):** En OpenGL, la vista de la cámara se controla a través de una matriz de vista, que transforma los objetos desde el sistema de coordenadas del mundo al sistema de coordenadas de la cámara. Esta matriz tiene en cuenta la posición y orientación de la cámara.

**3. Matrices de Proyección (Projection Matrix):** La matriz de proyección controla cómo se proyecta la escena 3D en la pantalla 2D. Define el tipo de proyección (perspectiva, ortográfica), el campo de visión y la relación de aspecto.

**4. Movimiento de la Cámara:** Puedes controlar el movimiento de la cámara cambiando su posición y orientación en respuesta a eventos del usuario, como teclas o el movimiento del ratón. Esto puede incluir desplazamientos (traslación) y rotaciones.

**5. Control de la Perspectiva:** Puedes implementar diferentes tipos de perspectivas, como perspectiva o ortográfica, según los requisitos de tu aplicación. La perspectiva simula cómo percibimos el mundo en la vida real, mientras que la ortográfica proporciona una proyección más uniforme.

**6. Cámaras Libres o Fijas:** Dependiendo de tu aplicación, puedes implementar cámaras que el usuario pueda controlar libremente o cámaras fijas que sigan un camino predefinido. Las cámaras libres son comunes en videojuegos, mientras que las cámaras fijas pueden ser útiles en presentaciones controladas.

**7. Actualización de las Matrices:** Las matrices de vista y proyección deben actualizarse en cada cuadro de renderización para reflejar los cambios en la posición y orientación de la cámara. Esto se hace típicamente en el ciclo de renderización de tu aplicación.

**8. Interacción con OpenGL:** El manejo de cámaras en OpenGL es esencial para asegurar que la escena se visualice correctamente. Las matrices de vista y proyección se multiplican con las matrices de modelo para transformar los objetos en el espacio 3D antes de su renderización.

## Sonido

OpenGL en sí mismo no maneja directamente el sonido, ya que está diseñado principalmente para gráficos en 3D. Sin embargo, si estás trabajando en una aplicación que implica tanto gráficos en 3D como sonido, puedes utilizar bibliotecas adicionales junto con OpenGL en Python para gestionar la parte de sonido. Un ejemplo común es la combinación de PyOpenGL con una biblioteca de audio como Pygame o Pyglet.

**1. Bibliotecas de Audio:** Utiliza bibliotecas como Pygame o Pyglet que permiten la reproducción de archivos de audio en Python. Estas bibliotecas proporcionan funciones para cargar, reproducir y manipular archivos de sonido.

**2. Integración con OpenGL:** Puedes utilizar OpenGL para renderizar tu escena 3D, mientras que la biblioteca de audio se encarga de la reproducción del sonido. Asegúrate de que ambas bibliotecas se integren bien y funcionen simultáneamente en tu aplicación.

**3. Sincronización:** Es importante tener en cuenta la sincronización entre la reproducción de sonido y la renderización gráfica para garantizar una experiencia multimedia fluida. Ajusta los tiempos y eventos para que el sonido y los gráficos estén sincronizados correctamente.

**4. Carga de Archivos de Sonido:** Usa las funciones proporcionadas por la biblioteca de audio para cargar archivos de sonido en formatos comunes como WAV, MP3 u otros, según la compatibilidad de la biblioteca.

**5. Control de Volumen y Efectos de Sonido:** Muchas bibliotecas de audio permiten ajustar el volumen, aplicar efectos de sonido y realizar otras manipulaciones en tiempo real. Puedes controlar estos aspectos para mejorar la experiencia auditiva de tu aplicación.

**6. Manejo de Eventos:** Implementa la lógica necesaria para activar eventos de sonido en respuesta a ciertas acciones en tu aplicación. Por ejemplo, podrías reproducir un sonido específico cuando un objeto en la escena colisiona con otro.

## Análisis FODA

1. Subway Surfers

| Fortalezas | Oportunidades |
|---|---|
| **Popularidad Global:** Subway Surfers ha ganado una gran base de fans en todo el mundo, lo que se refleja en su enorme cantidad de descargas y en su presencia en las listas de juegos más populares en múltiples plataformas móviles.<br>**Jugabilidad Sencilla:** El juego presenta controles simples e intuitivos que son fáciles de aprender, lo que lo hace accesible para jugadores de todas las edades y niveles de habilidad. La mecánica básica de correr, saltar y deslizarse es fácil de entender y ofrece una experiencia de juego rápida y gratificante.<br>**Actualizaciones Constantes:** Los desarrolladores continúan proporcionando actualizaciones regulares que introducen nuevos contenidos, personajes, tablas de surf y eventos especiales, lo que mantiene fresca la experiencia de juego y fomenta la participación continua de los jugadores. | **Expansión de la Monetización:** A través de la implementación de estrategias de monetización adicionales, como la inclusión de más opciones de personalización para los personajes o la introducción de elementos premium, el juego podría aumentar sus ingresos sin comprometer la experiencia del jugador.<br>**Exploración de Nuevas Plataformas:** Subway Surfers ha demostrado ser exitoso en dispositivos móviles, pero existe la oportunidad de expandirse a otras plataformas, como consolas de videojuegos o PC, lo que podría atraer a una audiencia más amplia y diversa. |
| **Debilidades** | **Amenazas** |
| **Repetitividad:** A pesar de su popularidad, algunos jugadores pueden encontrar que la jugabilidad de correr sin fin se vuelve repetitiva con el tiempo, ya que las mecánicas básicas del juego no cambian significativamente a lo largo de la experiencia.<br>**Dependencia de Microtransacciones:** Aunque el juego es gratuito para jugar, depende en gran medida de las microtransacciones para generar ingresos. Esto puede alienar a algunos jugadores que prefieren experiencias de juego completamente gratuitas o que sienten que están siendo presionados para gastar dinero en el juego. | **Competencia en el Mercado:** La industria de los juegos móviles es altamente competitiva, y existen numerosos competidores que ofrecen juegos similares de correr sin fin. La aparición de nuevos competidores podría reducir la cuota de mercado de Subway Surfers.<br>**Fatiga del Jugador:** A medida que pasa el tiempo, los jugadores pueden cansarse del juego si no se introducen cambios significativos o nuevas características con regularidad. La falta de innovación podría llevar a una disminución en el compromiso de los jugadores y, en última instancia, a una disminución en la popularidad del juego. |

2. 60 Segundos

| Fortalezas | Oportunidades |
|---|---|
| **Concepto Único:** "60 Seconds!" ofrece un enfoque único para la supervivencia en un entorno post-apocalíptico, con su mecánica de juego dividida en la fase de recolección y la fase de sobrevivencia. Este concepto distintivo lo diferencia de otros juegos de supervivencia.<br>**Rejugabilidad:** La naturaleza aleatoria de las situaciones y eventos en el juego proporciona una alta rejugabilidad. Cada partida ofrece una experiencia diferente, lo que anima a los jugadores a volver a jugar para explorar diferentes decisiones y resultados.<br>**Toma de Decisiones Significativas:** El juego presenta decisiones significativas y ramificaciones narrativas basadas en las elecciones del jugador durante la fase de recolección y la fase de sobrevivencia. Esto proporciona una sensación de agencia al jugador y aumenta la inmersión en la experiencia del juego. | **Expansión de Contenido:** Hay oportunidades para expandir el juego mediante la adición de nuevos eventos, situaciones y elementos de juego que aumenten la variedad y la profundidad de la experiencia del jugador.<br>**Exploración de Plataformas Adicionales:** "60 Seconds!" ha sido bien recibido en plataformas como PC y consolas, pero podría explorar la posibilidad de lanzar versiones para dispositivos móviles, lo que podría aumentar su alcance y su base de jugadores. |
| **Debilidades** | **Amenazas** |
| **Curva de Aprendizaje Pronunciada:** Para algunos jugadores, la curva de aprendizaje puede ser empinada, especialmente al principio del juego, ya que se enfrentan a la necesidad de tomar decisiones rápidas y estratégicas con limitada información y tiempo.<br>**Gráficos Simplistas:** Aunque el estilo artístico del juego tiene su encanto, los gráficos simplistas pueden no ser del agrado de todos los jugadores, y algunos podrían considerar que carece de la calidad visual que se encuentra en otros juegos modernos. | **Competencia en el Mercado:** El mercado de los juegos de supervivencia está saturado con una variedad de títulos, lo que significa que "60 Seconds!" compite con otros juegos similares por la atención de los jugadores. La aparición de nuevos competidores podría afectar su cuota de mercado.<br>**Fatiga del Género:** Existe el riesgo de que los jugadores experimenten fatiga del género si sienten que están viendo las mismas mecánicas y temas repetidos en varios juegos de supervivencia. Esto podría llevar a una disminución en el interés en "60 Seconds!" y juegos similares. |

3. Vector

| Fortalezas | Oportunidades |
|---|---|
| **Controles Intuitivos:** Vector cuenta con controles intuitivos que son fáciles de entender y dominar, lo que permite a los jugadores sumergirse rápidamente en la acción sin sentirse abrumados.<br>**Progresión del Personaje:** A medida que los jugadores avanzan en el juego, tienen la oportunidad de desbloquear nuevas habilidades y mejoras para su personaje, lo que proporciona un incentivo adicional para seguir jugando y mejorar. | **Eventos Especiales y Temporadas:** Implementar eventos especiales o temporadas temáticas podría aumentar la participación de los jugadores y mantener el interés a largo plazo, ofreciendo recompensas exclusivas y desafíos únicos durante períodos específicos.<br>**Expansión a Nuevas Plataformas:** Explorar la posibilidad de lanzar Vector en nuevas plataformas, como consolas de videojuegos o PC, podría aumentar su alcance y llegar a una audiencia más amplia que podría no estar presente en dispositivos móviles. |
| **Debilidades** | **Amenazas** |
| **Limitaciones de Personalización:** Aunque Vector ofrece cierta personalización a través de la progresión del personaje, algunos jugadores podrían desear más opciones de personalización, como la capacidad de modificar la apariencia o el equipo del personaje de manera más significativa.<br>**Repetitividad en Niveles:** Algunos jugadores podrían encontrar que los niveles se vuelven repetitivos después de un tiempo, ya que los desafíos y obstáculos pueden comenzar a sentirse familiares. Esto podría afectar la experiencia general del juego para algunos jugadores. | **Cambios en las Tendencias de Juegos Móviles:** Las tendencias en los juegos móviles están en constante evolución, y Vector podría enfrentarse a la amenaza de volverse obsoleto si no puede adaptarse a los cambios en las preferencias de los jugadores o las innovaciones en la industria.<br>**Expectativas de la Comunidad:** Si las expectativas de la comunidad de jugadores no se satisfacen o si hay una falta de comunicación con los desarrolladores sobre actualizaciones y mejoras, esto podría resultar en una disminución en la participación y el interés en el juego. |

4. Temple Run

| Fortalezas | Oportunidades |
|---|---|
| **Actualizaciones Constantes:** El juego ha recibido actualizaciones regulares que agregan nuevos contenidos, características y desafíos, lo que mantiene la experiencia fresca y emocionante para los jugadores a largo plazo.<br>**Reconocimiento de Marca:** Temple Run se ha convertido en una marca reconocida y establecida en la industria de los juegos móviles, lo que puede generar confianza y lealtad entre los jugadores que buscan experiencias de calidad. | **Expansión a Nuevas Plataformas:** Aunque Temple Run se lanzó inicialmente en dispositivos móviles, existe la oportunidad de expandirse a otras plataformas, como consolas de videojuegos, PC o incluso realidad virtual, para llegar a una audiencia más amplia.<br>**Marketing Colaborativo:** La colaboración con otras marcas o franquicias populares para eventos promocionales especiales podría aumentar la visibilidad de Temple Run y atraer a nuevos jugadores que estén interesados en las colaboraciones temáticas. |
| **Debilidades** | **Amenazas** |
| **Limitaciones de Variedad:** A pesar de su jugabilidad adictiva, algunos jugadores pueden encontrar que la falta de variedad en los niveles y los obstáculos puede hacer que el juego se vuelva repetitivo después de un tiempo, lo que podría afectar la retención a largo plazo.<br>**Monetización Intrusiva:** Aunque el juego es gratuito para jugar, la presencia de anuncios publicitarios y las compras dentro de la aplicación pueden resultar intrusivas para algunos jugadores, lo que podría afectar negativamente la experiencia del usuario. | **Tendencias de Juegos Emergentes:** La popularidad de nuevos géneros o estilos de juego podría desviar la atención de los jugadores de Temple Run, especialmente si no puede adaptarse para mantenerse relevante en un mercado en constante evolución.<br>**Cambios en la Tecnología:** Los avances tecnológicos en dispositivos móviles pueden generar expectativas más altas entre los jugadores en términos de gráficos, rendimiento y características, lo que podría poner a Temple Run en desventaja si no puede mantenerse al día con estas demandas. |

5. Jetpack JoyRide

| Fortalezas | Oportunidades |
|---|---|
| **Diseño Visual Atractivo:** Jetpack Joyride cuenta con gráficos vibrantes y coloridos que son atractivos para los jugadores. El diseño visual contribuye a una experiencia de juego inmersiva y emocionante.<br>**Personalización del Personaje:** Los jugadores tienen la capacidad de personalizar a su personaje con una variedad de disfraces y accesorios desbloqueables. Esta característica añade un elemento de diversión y permite a los jugadores expresar su estilo único mientras juegan. | **Colaboraciones y Crossovers:** Existen oportunidades para colaboraciones con otras marcas o franquicias populares, lo que podría resultar en eventos especiales temáticos o contenido exclusivo que atraiga a nuevos jugadores y mantenga el interés de los existentes.<br>**Integración de Funciones Sociales:** La adición de características sociales, como la capacidad de compartir logros en las redes sociales o desafiar a amigos a competiciones, podría fomentar la interacción entre los jugadores y aumentar el compromiso a largo plazo. |
| **Debilidades** | **Amenazas** |
| **Limitaciones de Innovación:** Aunque Jetpack Joyride ha sido exitoso, algunas personas podrían argumentar que el juego no ha innovado significativamente desde su lanzamiento inicial, lo que podría llevar a una pérdida de interés por parte de los jugadores que buscan experiencias nuevas y emocionantes.<br>**Dependencia de Compras Dentro de la Aplicación:** La dependencia de las compras dentro de la aplicación como principal fuente de ingresos puede generar críticas de algunos jugadores que prefieren una experiencia de juego completamente gratuita o que se sienten frustrados por las mecánicas de monetización del juego. | **Cambios en las Preferencias del Jugador:** Los cambios en las preferencias de los jugadores o las tendencias del mercado podrían afectar la popularidad de Jetpack Joyride, especialmente si los jugadores se inclinan hacia nuevos géneros o estilos de juego.<br>**Desarrollo de Competidores:** La aparición de nuevos competidores con juegos similares o innovadores en el mercado podría representar una amenaza para Jetpack Joyride al competir por la atención de los jugadores y la cuota de mercado en el género de los juegos de correr sin fin. |

6. Trash's Collector

| Fortalezas | Oportunidades |
|---|---|
| **Conciencia Ambiental:** El juego ofrece una plataforma efectiva para crear conciencia sobre la importancia de la gestión de residuos y el impacto positivo que puede tener en el medio ambiente. Esta conciencia puede llevar a una mayor adopción de prácticas responsables por parte de los jugadores en la vida real.<br>**Potencial de Colaboraciones:** Existe la oportunidad de colaborar con organizaciones ambientales, instituciones educativas y otras entidades afines para amplificar el alcance del juego y aumentar su impacto en la sociedad. Las colaboraciones pueden proporcionar recursos adicionales y conocimientos especializados para mejorar el contenido del juego. | **Personalización y Progresión:** Agregar elementos de personalización para el personaje del recolector de basura y la capacidad de desbloquear mejoras a medida que los jugadores avanzan en el juego podría aumentar la participación y el compromiso de los jugadores a largo plazo.<br>**Expansión a Plataformas Móviles:** Considerar la adaptación del juego para dispositivos móviles podría aumentar significativamente su accesibilidad y llegar a una audiencia más amplia, incluidos los jugadores más jóvenes que son una demografía clave para la educación ambiental. |
| **Debilidades** | **Amenazas** |
| **Limitaciones Tecnológicas:** Dependiendo de las capacidades de las librerías de Python utilizadas para el desarrollo del juego, podrían surgir limitaciones en términos de gráficos, rendimiento y compatibilidad con diferentes sistemas operativos, lo que podría afectar la experiencia del usuario.<br>**Competencia con Otras Formas de Entretenimiento:** El juego podría enfrentarse a la competencia de otras formas de entretenimiento, como aplicaciones educativas, libros y programas de televisión, que también buscan educar sobre la gestión de residuos y la protección del medio ambiente. | **Cambios en las Políticas Ambientales:** Los cambios en las políticas gubernamentales y las regulaciones ambientales podrían afectar la percepción del problema de la basura y la necesidad de soluciones educativas como "Trash's Collector". Adaptarse a estos cambios podría ser necesario para mantener la relevancia del juego.<br>**Desinterés del Público:** Existe el riesgo de que el público objetivo pierda interés en el juego si no se mantiene actualizado con contenido nuevo y emocionante, o si no se comunica eficazmente su valor educativo y ambiental. |

## Marco metodológico

## Requerimientos funcionales

- **Interactividad educativa**
  Esto implica que el juego debe incluir elementos de juego que presenten situaciones realistas donde los jugadores tengan que tomar decisiones sobre cómo clasificar y manejar diferentes tipos de residuos. Por ejemplo, podrían presentarse escenarios donde los jugadores deben seleccionar los contenedores de reciclaje adecuados para diferentes materiales, como papel, plástico o vidrio. Además, el juego podría proporcionar retroalimentación inmediata sobre la precisión de las elecciones del jugador, destacando los impactos positivos o negativos de sus acciones en el entorno virtual del juego. Además de las actividades directas de clasificación de residuos, el juego también podría incluir mini-juegos o desafíos que enseñen conceptos específicos relacionados con el reciclaje y la gestión de residuos, como la importancia de reducir el uso de plásticos de un solo uso o la reutilización de materiales. Estos elementos interactivos no solo mantienen a los jugadores comprometidos, sino que también refuerzan los conocimientos adquiridos de una manera práctica y memorable.

- **Mecánicas de juego**
  Estas mecánicas de juego pueden variar en complejidad y dificultad a medida que avanza el juego. Por ejemplo, al principio, los desafíos pueden ser simples, con pocos tipos de residuos y contenedores disponibles. Conforme avance el juego, los desafíos podrían volverse más difíciles, presentando una mayor variedad de materiales y contenedores, así como obstáculos adicionales, como límites de tiempo o penalizaciones por clasificar incorrectamente los residuos, el juego puede proporcionar retroalimentación inmediata sobre las elecciones del jugador, destacando si clasificaron correctamente los residuos o no, y brindando información adicional sobre por qué ciertos materiales deben ir en contenedores específicos.

- **Progresión de niveles**
  El juego debe ofrecer una estructura de progresión en la que los niveles se vuelvan más desafiantes a medida que los jugadores avanzan. Esto garantiza que la experiencia de juego sea gradualmente más exigente y estimulante, manteniendo el interés y la motivación de los jugadores a medida que exploran y aprenden sobre el reciclaje y la separación de residuos.
  La progresión de niveles puede lograrse mediante diversos medios, como aumentar la complejidad de los desafíos de clasificación de residuos, introducir nuevos tipos de materiales o contenedores, agregar obstáculos

adicionales, como límites de tiempo o restricciones en los movimientos del jugador, o presentar mecánicas de juego completamente nuevas que requieran una mayor habilidad y estrategia por parte del jugador.
La clave de este requerimiento es mantener un equilibrio entre desafío y accesibilidad, asegurándose de que los niveles se vuelvan gradualmente más difíciles sin resultar abrumadores para los jugadores. Esto garantiza una experiencia de juego satisfactoria y estimulante que motive a los jugadores a seguir avanzando y aprendiendo sobre el tema del reciclaje y la separación de residuos a lo largo del juego.

## Requerimientos no funcionales

- **Rendimiento del juego**
  El juego debe ser compatible y funcionar sin problemas en una variedad de dispositivos y sistemas operativos, garantizando una experiencia fluida y sin interrupciones para todos los jugadores. Esto requiere optimización del rendimiento y pruebas exhaustivas en diferentes plataformas durante el desarrollo del juego.

- **Estética visual y sonora**
  El juego consta de baja estática visual y sonora, debido al entorno de desarrollo “primitivo” y a las limitaciones del mismo no hay tanta estética como para considerarse atractivo, cuenta con las formas básicas, colores, movimientos poco libres, y en cuanto a sonido, cuenta con sonidos de las acciones que realizan los personajes, así como una que otra melodía de fondo, con posible Copyright.

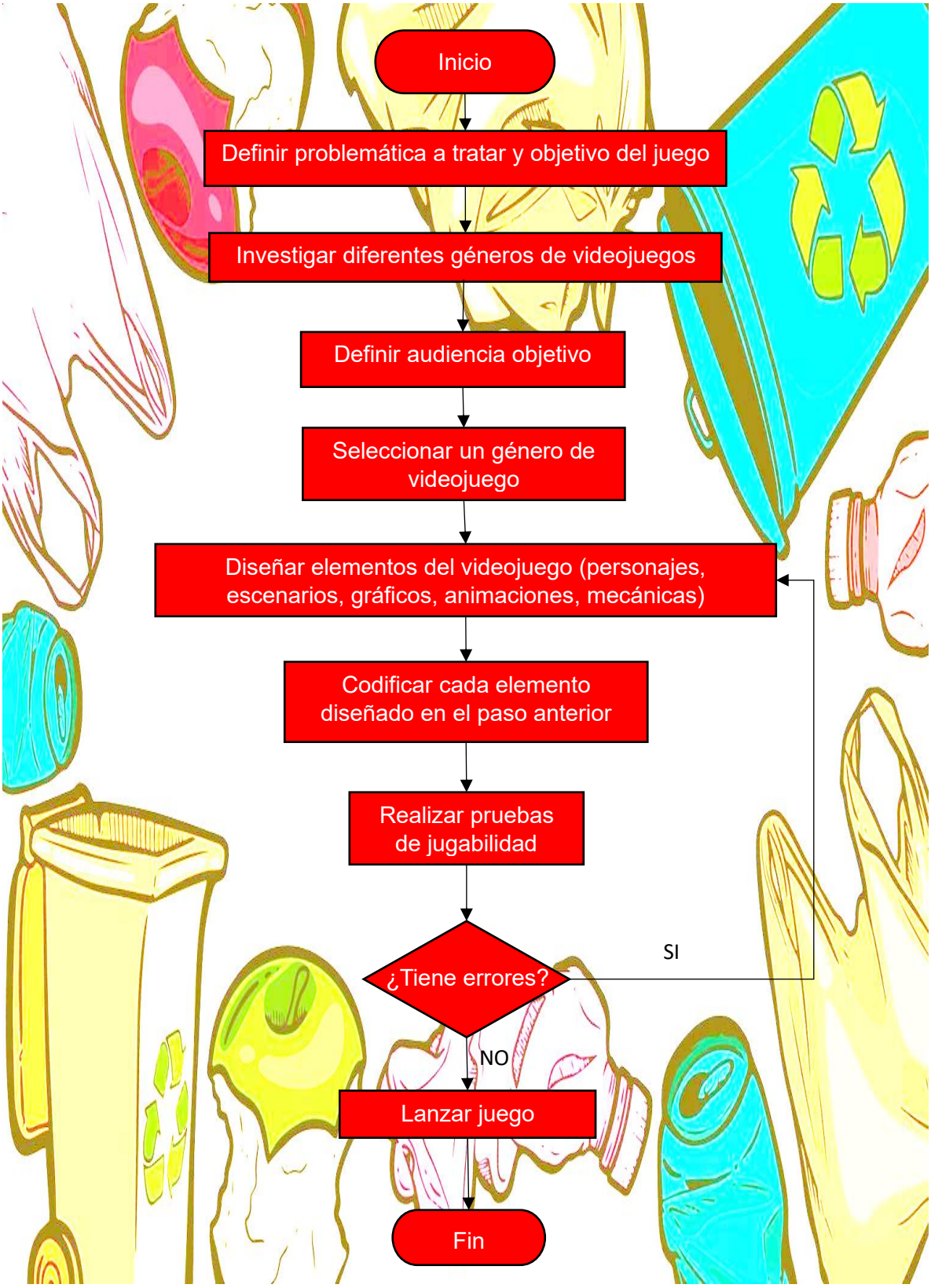
## Ciclo de vida de software

Contaría con el clásico ciclo de vida:

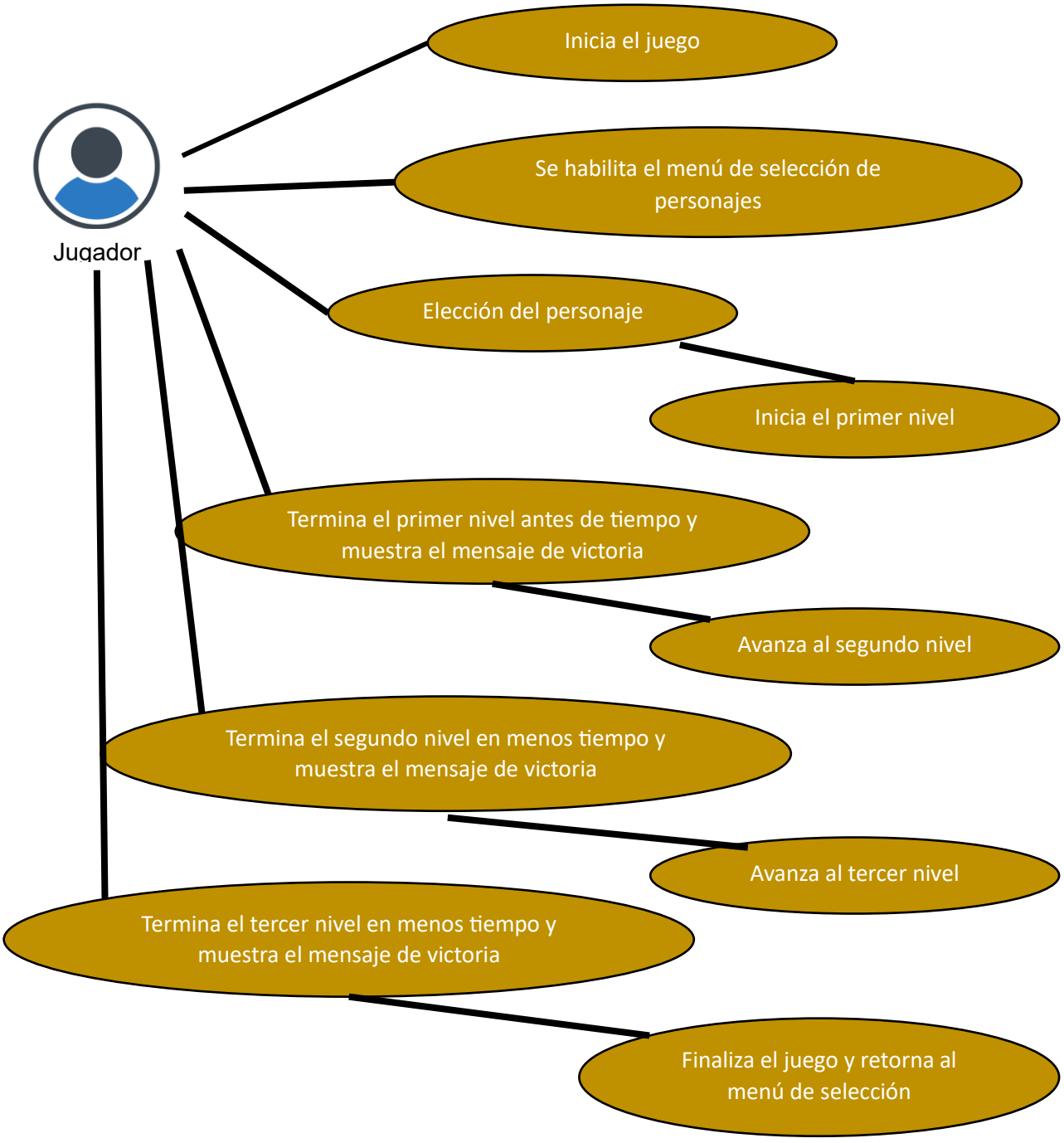
1. Planificación: esta etapa comenzó a mediados del curso, cuando se dieron propuestas y se tomó la decisión de el tipo y tema a desarrollar del videojuego, así como los personajes que se utiizarán/desarrollarán, se vieron las alternativas para realizar bocetos, librerías necesarias, códigos de colores, etc.
2. Análisis: se evaluaron y optaron por la mejor opción para poder realizar los bocetos de los personajes, tales como GeoGebra, Paint 3D, Python o dibujo, y de ahí partir de ahí buscar el método para importar ese boceto a modelo 3D, así como la realización de las funciones en código que nos ayudaran a generar las figuras y la optimización del código, en base a esos rubros se elegirá la opción optima.

3. Diseño: se comenzarán con el uso de la mejor plataforma para generar los bocetos y visualizar como quedaran los personajes en 3D, así como poder graficarlo en 3 dimensiones, asignarle coordenadas y poderlos escalar o hacer las transformaciones necesarias; además, se contemplarán también los escenarios que se usaran para los diferentes niveles.
4. Codificación: se hará uso de Visual Studio Code para llevar a código Python todo lo tratado en la fase de diseño, desde la interfaz mostrada al usuario, las funciones que crearan las figuras para generar a los personajes, le generación de escenarios, sonidos, iluminación.
5. Pruebas: se ejecuta el código para probar todas las funciones del juego y comprobar que no exista error alguno, que en la interfaz grafica la interacción con el usuario sea correcta, se ejecuten los movimientos, los sonidos, puntuaciones, etc.
6. Implementación: se guarda como ejecutable para que pueda correr en cualquier computadora y sea utilizada por otros usuarios ajenos al equipo de desarrollo
7. Mantenimiento: seguimiento periódico para erradicar errores, actualizaciones, etc.

## Diagrama de flujo de actividades



## Diagrama de caso de uso

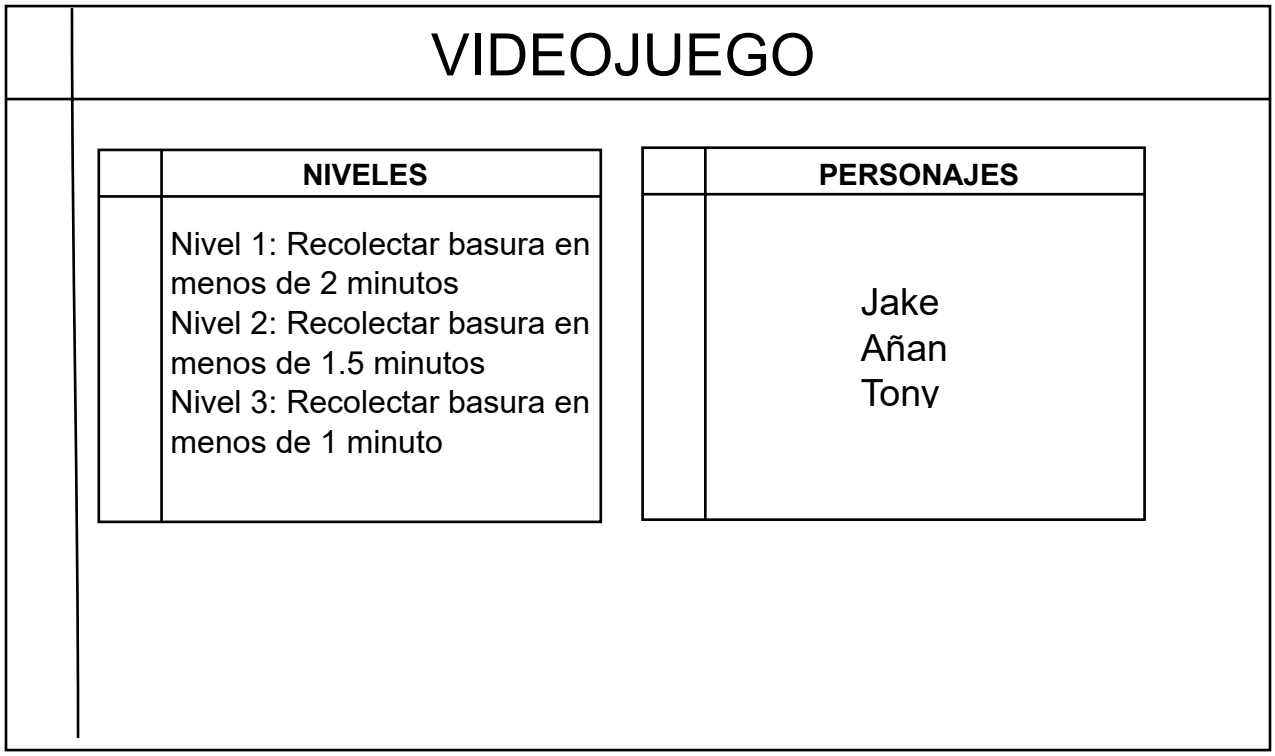


## Plantilla de caso de uso

| | | |
|---|---|---|
| **JBA0001** | | Jugabilidad |
| **Versión** | | Betha 1.0, 5 de mayo de 2024 |
| **Autores** | | Jugador |
| **Fuentes** | | Python, Visual Studio Code |
| **Objetivos asociados** | | El jugador deberá terminar cada uno de los niveles sin sobrepasar el limite de tiempo |
| **Descripción** | | Se describe la secuencia de acciones que realiza el jugador para iniciar el juego, seleccionar un personaje, progreso y funcionamiento de niveles, así como las ocasiones en las cuales se regrese al menú principal. |
| **Precondición** | | El juego se instala y funciona correctamente<br>Existen 3 niveles jugables<br>Al iniciar el juego se accede al menú de personajes |
| **Secuencia Normal** | | |
| | **Actor** | **Sistema** |
| **1** | Inicia el juego | Muestra a los personajes |
| **2** | Accede al menú de personajes | Muestra el nombre de los personajes y a los mismos |
| **3** | Elije un personaje | Inicia el nivel según sea el caso |
| **4** | Inicia el primer nivel | Se muestra un temporizador para el nivel |
| **5** | Completa el primer nivel | Muestra un mensaje: ¡YOU WIN!, y retorna al menú de personajes para iniciar el siguiente nivel |
| **Excepciones** | | |
| | **Actor** | **Sistema** |
| **1.1** | El jugador presiona le tecla “ESC” o al botón de encendido | El juego se cierra o se apaga el equipo |
| **2.1** | No logra coleccionar todos los objetos del nivel | El progreso del jugador lo regresara al nivel uno |
| **3.1** | El límite de tiempo se acaba | Retorna el jugado al menú principal |

| Rendimiento | Paso | Cota de tiempo |
|---|---|---|
| **Inicialización del juego** | 1 | 3 segundos |
| **Elección del personaje** | 2 | 2 segundos |
| **Salir del juego** | 3 | 4 segundos |
| **Frecuencia esperada** | 3/una ejecución | |
| **Importancia** | | |
| **Postcondición** | El jugador puede seleccionar cualquier personaje para jugar cualquier nivel, con la característica de que como complete niveles, el tiempo límite ira reduciendo | |
| **Urgencia** | Ingresar a los niveles y regresar al menú principal es instantáneo | |
| **Comentarios** | | |

## Diagrama de paquetes



## Desarrollo de solución

### Historia del juego

Como se mencionó en el planteamiento del problema, el juego consiste en asumir el papel de un personaje encargado de limpiar y recoger la basura que se encuentra en el entorno. Al principio de cada nivel, el jugador se situara a mitad del escenario y recorrerá el mismo para recolectar las bolsas de basura, cada vez que recolecte una un contador en la parte superior ira incrementando, todo esto antes de que se agote el tiempo establecido para el nivel.

El desafío radica en la capacidad del jugador para recolectar rápidamente las bolsas de basura dentro del límite de tiempo dado. Si el jugador no logra recolectar todas las bolsas dentro del límite de tiempo, el juego termina y debe intentarlo de nuevo desde el principio del nivel.

### Manera de juego

### Descripción de los personajes y personajes en 3D

Añan es un pequeño panda de un azul vibrante y pelaje cuadrado, una rareza entre su especie. Sus líneas rectas y ángulos definidos le otorgan una apariencia única y distintiva. Su figura es compacta pero ágil, con movimientos que denotan una gracia natural a pesar de su forma poco convencional.

Sus ojos son grandes y expresivos, reflejando una curiosidad insaciable por el mundo que lo rodea. Su hocico cuadrado se curva en una sonrisa amistosa, y sus orejas cuadradas se inclinan hacia adelante cuando está concentrado en algo.

A pesar de su inusual forma, Añan tiene un corazón valiente y una determinación inquebrantable. Siempre está listo para embarcarse en nuevas aventuras y ayudar a sus amigos cuando lo necesitan. Su singularidad lo convierte en un compañero valioso y entrañable en cualquier situación.

Tony es un personaje humanoide equipado con una armadura que recuerda a Iron Man. Esta armadura presenta una paleta de colores llamativa de rojos, blancos y amarillos, complementada por un casco en tonos de gris y dorado. Tony posee la capacidad de mover las piernas, levantar ambos brazos, girar la cabeza y mirar hacia arriba. Además, sus rasgos faciales son expresivos, permitiendo una variedad de gestos y expresiones.

Jake, versión en un mundo paralelo de Jake el perro de la caricatura Hora de Aventura, a diferencia del original este hace uso de vestimenta, tales como pantalón, zapatos y sombrero, cuenta con las habilidades de saltar, quitarse el sombrero, agacharse, contando con varias expresiones como guiño, sonrisa, enojo, etc.

**Niveles por personaje**

En los 3 diferentes niveles, se puede elegir al personaje que se desee, pero al tener progreso cada nivel disminuirá el temporizador, es decir, desde el nivel uno se tendrá un límite de tiempo de 2 minutos para recolectar todos los objetos que se encuentran por todo el escenario, can la posibilidad de moverse por todo del escenario (enfrente, atrás, derecha, izquierda) así como de mover la cámara para visualizar todo el escenario y percatarse de los objetos repartidos por el mismo.

Al recoger un objeto no desaparecerá, sin embargo, hará un sonido aumentando el contador y si se vuelve a pasar por la misma posición, este no afectara debido a que se ha recolectado, cuando termine de recoger todos los objetos antes de que se agote el tiempo el nivel acabara mostrando un mensaje de “YOU WIN” y retornándolo a la pantalla de selección de personaje al oprimir “F12” para seleccionar el mismo u otro diferente para el siguiente nivel, pero esta vez con menos tiempo para recolectar los objetos.

En caso de que no logre recolectar todos los objetos en el límite de tiempo estipulado retornara al jugador al menú de selección y comenzara de nuevo en el nivel 1 sin importar el nivel en el que haya perdido, manteniendo la idea de los clásicos juegos arcade de 1 vida

**Como muestra la ayuda**

Esta información se mostrará con la etiqueta “i) información” en la esquina de la pantalla donde al pulsarla se mostrará las teclas para poder jugar.

**Como muestra los créditos**


**E**n el menú principal se habilito la opción de poder mostrar los créditos del juego, es decir, la información acerca de los desarrolladores del juego, también puede ocultarse para una mejor visualización del menú

## Código relevante explicado

```
MENU(0, 0, 0)
if per == 1:
    if puntos != 8:
        tiempo += 1
    dibujar1(0, 0, 0)
elif per == 2:
    if puntos != 8:
        tiempo += 1
    dibujar2(0, 0, 0)
elif per == 3:
    dibujar3(0, 0, 0)
    if puntos != 8:
        tiempo += 1
elif per == 5:
    esc = 0
    band = 0
    per = 0
    jake = 0
    panda = 0
    pos1 = [0, 0, 0]
    tiempo = 0
    puntos = 0
    colisiones_detectadas = []
    MENU(0, 0, 0)
```

Dentro del bucle while mandamos a llamar la función del menú, para que al ejecutar el programa como pantalla inicial se muestre el menú para permitir a usuario elegir el personaje de su gusto, al elegir el personaje se le asigna un valor específico a la bandera per, donde después de la llamada la función menú, se verifica el valor que tiene asignada esta bandera y según el dato que tenga se llama una función en la cual se mostrará escenario las figuras de residuos el personaje elegido un cronómetro un texto donde se visualiza la puntuación y una opción de información.

```
tiempo = 0
```

Para el manejo del cronómetro creamos una variable llamada tiempo inicializada en cero, La cual aumenta en uno en el bucle while justo después de mandar a llamarla que dibuja el escenario y el personaje, esta bandera contabiliza los milisegundos transcurridos mediante la ejecución del juego, para dar formato al texto que mostrará el cronómetro en minutos segundo si milisegundos se usó la función formatoTime.

```
def formatoTime(tiempo):
    minutos = tiempo // 600  # 600 frames = 1 minuto
    segundos = (tiempo // 10) % 60  # 10 frames = 1 segundo
    milisegundos = (tiempo % 10) * 100  # 1 frame = 100 milisegundos
    texto_cronometro = "{:02}:{:02}:{:03}".format(minutos, segundos, milisegundos)
    return texto_cronometro
```

```python
import random
from a_escenario import load_texture

puntos = [None] * 8  # Inicializa la lista p con 8 elementos None

class Basura:
    def __init__(self):
        self.planos_basura = []

    def crear_planos_laterales_random(self):
        for _ in range(8):  # Crear 8 planos
            x = random.uniform(-60, 55)  # Rango de -60 a 55 para que el plano tenga una anchura de 5 unidades
            z = random.uniform(-60, 60)  # Valor constante para z
            plano = [
                (x, -7.5, z),
                (x + 5, -7.5, z),
                (x + 5, -2.4, z),
                (x, -2.5, z)
            ]
            puntos[_] = (x + 2.5, 0, z)
            self.planos_basura.append(plano)

    def dibujar_planos_basura(self, filename):
        glEnable(GL_TEXTURE_2D)
        glBindTexture(GL_TEXTURE_2D, load_texture(filename))
        glBegin(GL_QUADS)
        glColor(1, 1, 1)
        for plano in self.planos_basura:
            glTexCoord2f(0, 0)
            glVertex3f(*plano[0])
            glTexCoord2f(1, 0)
            glVertex3f(*plano[1])
            glTexCoord2f(1, 1)
            glVertex3f(*plano[2])
            glTexCoord2f(0, 1)
            glVertex3f(*plano[3])
        glEnd()

obj = Basura()

# Llamar a esta función para generar un nuevo plano de basura
obj.crear_planos_laterales_random()

def crea_basura():
    # Dibujar otros objetos si es necesario
    obj.dibujar_planos_basura("Imagenes/basura.jpg")
```

Esta función recibe como parámetro nuestra variable de tiempo, y con este dato realiza las operaciones correspondientes para calcular los minutos, segundos y los segundos retornando un texto con el formato y valores deseados.

Para hacer posible la visualización de los que representarán los residuos de la basura dentro de nuestro escenario se tuvo que hacer uso de una clase llamada basura, donde se usa como atributo una lista vacía llamada planos_basura, posteriormente en la función de crear los planos se establecen valores directorios a las variables X, Z, las cuales establecerán las posiciones de los planos que contendrán las texturas de nuestros reside, una vez que se tiene establecida la posición de estos planos, es almacenada en la lista planos_basura.

Ya que la lista tiene almacenadas todas las posiciones de los planos que aparecerán en pantalla, en la función dibujar_planos_basura se aplican las funciones correspondientes para su gratificación y texturizado.

Posteriormente la función crea_basura es la encargada de mandar a llamar la función que grafica estos planos.

```
415
416 def drawObj1():
417     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
418     glEnable(GL_DEPTH_TEST)
419     glPushMatrix()
420     p.crea_basura()
421     glPopMatrix()
422
```

Dentro de nuestro archivo main, se crea una función llamada drawObj1, en la cual se manda a llamar la función crea_basura Para hacer posible la gratificación de nuestros residuos.

Esta función drawObj1 es llamada dentro de las funciones en dónde se grafican cada uno de los personajes, haciendo así posible que en pantalla se muestre el personaje, escenario y residuos.

```
    # ----------Colisiones----------
    if per == 1:
        for i in range(8):
            if (
                col.check_collision_inters(pos1, p.puntos[i])
                and i not in colisiones_detectadas
            ):
                txt.texto(" +1 ", -13, 12 - 3, 0, 50, 0, 122, 204, 255, 255, 255)
                colisiones_detectadas.append(i)
                puntos += 1
                py.mixer.music.load("Sonidos/bonus.wav")
                py.mixer.music.play()

    if per == 2:
        posJake = [
            pos1[0] - 3,
            pos1[1],
            pos1[2] - 6.5,
        ]  # Ajustar la posición según la traslación
        for i in range(8):
            if (
                col.check_collision_inters(posJake, p.puntos[i])
                and i not in colisiones_detectadas
            ):
                txt.texto(" +1 ", -13, 12 - 3, 0, 50, 0, 122, 204, 255, 255, 255)
                colisiones_detectadas.append(i)
                puntos += 1
                py.mixer.music.load("Sonidos/bonus.wav")
                py.mixer.music.play()

    if per == 3:
        for i in range(8):
            if (
                col.check_collision_inters(pos1, p.puntos[i])
                and i not in colisiones_detectadas
            ):
                txt.texto(" +1 ", -13, 12 - 3, 0, 50, 0, 122, 204, 255, 255, 255)
                colisiones_detectadas.append(i)
                puntos += 1
                py.mixer.music.load("Sonidos/bonus.wav")
                py.mixer.music.play()
```

```
def dibujar2(x, y, z):
    pos1[0] += x  # Actualizar la coordenada x
    pos1[1] += y  # Actualizar la coordenada y
    pos1[2] += z  # Actualizar la coordenada z

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    drawObj1()

    es.piso("Imagenes/piso.jpg")
    es.techo("Imagenes/cielo.jpg")

    if esc in (0, -1):
        es.escenario("Imagenes/ciudad.jpg")
def dibujar3(x, y, z):
    pos1[0] += x  # Actualizar la coordenada x
    pos1[1] += y  # Actualizar la coordenada y
    pos1[2] += z  # Actualizar la coordenada z

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    drawObj1()

    es.piso("Imagenes/piso.jpg")
    es.techo("Imagenes/cielo.jpg")
```

Para la acumulación de puntaje mediante la obtension de residuos que vaya adquiriendo el personaje, hicimos uso de colisiones, de igual manera se implementó el uso de una lista para registrar todas las colisiones detectadas.

Dentro de nuestro bucle while se verifica si el personaje hace intersección con un residuo , entonces es detectada la colisión y registrada en la lista colisiones_detectadas .

Estas condicionales reciben como parámetro las posiciones actualizadas de cada personaje y la lista que contiene las posiciones de los residuos, con estos datos verifican si hay alguna colisión detectada y además si esa colisión detectada no ha sido detectada antes, si no ha sido detectada antes entonces esta colisión es válida de lo contrario no se detectará la colisión, esto nos permite que cada residuo sea colisionado sólo una vez para que nuestro puntaje no se vea afectado.

```
puntos = 0
```

Para la acumulación del puntaje hicimos uso de una variable llamar a puntos, la cual está inicializada con el valor de cero, y esta aumenta en uno cada que hay una colisión detectada, además se reproduce un sonido cada que se detecta una colisión, es decir se escucha un sonido al sumar un punto.

Ahora para mostrar en pantalla el puntaje, el cronómetro y la información de los controles, realizamos la función llamada draw_panel, en la cual se guardan las configuraciones actuales de las matrices de proyección y modelo de vista posteriormente se crea un estilo de panel en 2 dimensiones invisible, esto nos permitirá que los textos que queramos mostrar en pantalla se mantengan fijos en una sola posición y no se muevan junto con el movimiento de la cámara aplicada al juego. Dentro de esta función hacemos varias condicionales como:

sí nuestra bandera per es diferente de cero, o sea que hay un personaje seleccionado se mostrarán los textos que indican la puntuación, el cronómetro, y la

opción de información, de lo contrario si la variable per tiene asignado el valor de cero, es decir, no hay ningún personaje seleccionado y aún se encuentra el usuario en el menú se van a mostrar los textos que indican los nombres de cada personaje además de la tecla utilizar para seleccionar a cada uno de estos personajes.

En otra condicional se verifica si la bandera jake, panda o band (banderas que controlan los movimientos y gestos de cada personaje), tienen asignado el valor de 12, se mostrarán los controles de cada personaje.

Posteriormente se verifica que si la variable que acumula los puntos que se van obteniendo mediante la ejecución del juego tiene asignado el valor de 8, quiere decir que se han recolectado todos los puntos posibles entonces el personaje ha cumplido su fin y ha ganado el juego. Entonces el personaje será regresado a su posición original y aparecerá un texto que dice: ! YOU WIN ¡.

```
if (formatoTime(tiempo)) == "02:00:000" and puntos < 8:
    per = 5
    resetear_camara()
if puntos == 8:
    resetear_camara()
draw_panel()
```

Por último, en la parte final dentro de nuestro bucle while, se establece un tiempo límite de 2 minutos para recolectar los 8 puntos posibles y ganar el juego, si se ha llegado a los 2 minutos y no se han obtenidos todos los puntos, entonces automáticamente el nivel se cerrará y te regresará al menú principal además de reiniciar a la cámara, y también se verifica si la variable puntos ha llegado a 8, es decir que se ha ganado el juego también se reiniciara la cámara.

```
def resetear_camara():
    global rotaciones_acumuladas
    for angle in rotaciones_acumuladas[::-1]:
        glRotatef(-angle, 0, 1, 0)
    rotaciones_acumuladas = []
```

```
if x != 0 and per != 0:
    glRotate(x, 0, 1, 0)
    rotaciones_acumuladas.append(x)
    py.mouse.set_pos(display[0] // 2, display[1] // 2)  # Centramos el raton
    # glRotatef(1, 0, 0, 0)
```

Para lograr resetear la cámara y que ésta regresa su posición original, implementamos una lista en la cual después de cada movimiento rotatorio en la cámara, este será registrado en la lista.

Entonces esta lista tiene almacenados todos los movimientos registrados, y al resetear la camara estos movimientos son aplicados de forma inversa, logrando asi que regrese a su posicion inicial.

## Conclusiones

Los videojuegos educativos pueden ser una herramienta poderosa para concienciar sobre la contaminación causada por diferentes tipos de residuos. Su capacidad para involucrar activamente a los jugadores les permite experimentar directamente las consecuencias de sus acciones, lo que puede sensibilizarlos sobre la importancia de gestionar los residuos de manera responsable.

Estos juegos ofrecen la posibilidad de explorar diversos aspectos del problema, como sus causas, impactos y posibles soluciones, de forma interactiva y dinámica. Esto facilita la comprensión de conceptos complejos y estimula un pensamiento crítico sobre el tema.

Además, los videojuegos educativos tienen la ventaja de llegar a un público diverso, lo que los convierte en una herramienta educativa versátil que puede utilizarse en diferentes contextos, como la educación formal y la informal. **– Raúl Millán Desales**

En conclusión, los videojuegos educativos se han consolidado como una herramienta invaluable en el panorama educativo contemporáneo. Su capacidad para fusionar entretenimiento y aprendizaje los convierte en una opción atractiva y eficaz tanto para educadores como para estudiantes. La inmersión interactiva que ofrecen permite a los usuarios sumergirse en experiencias de aprendizaje envolventes y significativas, lo que facilita el desarrollo de una amplia gama de habilidades cognitivas, sociales y emocionales.

Al aprovechar la tecnología digital, los videojuegos educativos pueden presentar conceptos académicos de una manera innovadora y atractiva, lo que resulta en un mayor compromiso y motivación por parte de los estudiantes. Además, ofrecen la oportunidad de aprender a través del error y la experimentación, lo que puede promover un enfoque más práctico y reflexivo del aprendizaje.

Sin embargo, para que los videojuegos educativos sean efectivos, es fundamental un diseño cuidadoso que integre los principios pedagógicos con la diversión del juego. Esto implica la creación de experiencias de juego equilibradas que desafíen y estimulen a los estudiantes mientras les proporcionan retroalimentación significativa y oportunidades de aprendizaje autodirigido. **- Jair Garduño Rodríguez**

El desarrollo de un videojuego con el propósito de sensibilizar sobre el reciclaje representa un valioso esfuerzo que combina entretenimiento y educación de manera efectiva. Este tipo de juego no solo busca entretener, sino también educar sobre la importancia del reciclaje, la correcta clasificación de residuos y la preservación del medio ambiente. Aprovechar la popularidad y accesibilidad de los videojuegos brinda la oportunidad de llegar a una audiencia más amplia y comprometerla activamente en la adopción de prácticas sostenibles.

Los videojuegos han surgido como una herramienta valiosa para promover el aprendizaje en diversos campos. Su capacidad para sumergir al jugador en entornos interactivos y atractivos ofrece una experiencia de aprendizaje inmersiva y memorable. En el caso específico de los videojuegos de concienciación sobre el reciclaje, esta inmersión puede ser particularmente efectiva al permitir que los jugadores experimenten directamente los conceptos y desafíos relacionados con la gestión de residuos.

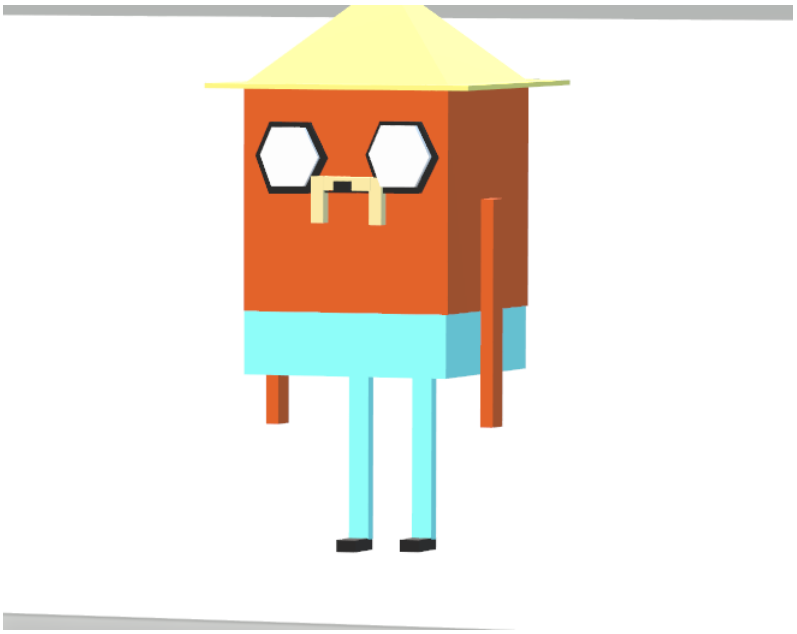
El uso del lenguaje de programación Python, reconocido por su simplicidad, versatilidad y potencia, se presenta como una opción ideal para el desarrollo de este tipo de videojuegos. Su sintaxis clara y legible simplifica la creación de código, lo que resulta especialmente beneficioso para desarrolladores novatos o aquellos que buscan un enfoque más accesible. Además, la variedad de bibliotecas y herramientas disponibles en Python, como Pygame, ofrece recursos robustos para diseñar y desarrollar juegos interactivos y dinámicos. **– Jonathan Esquivel Flores**

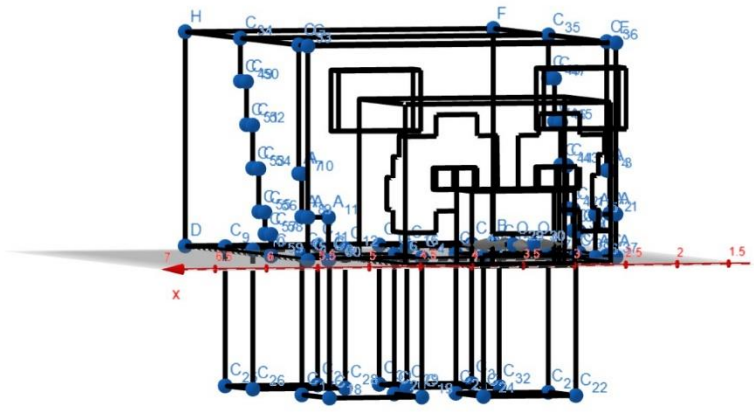
## Referencias bibliográficas

• Anonimo. (2022). Dibujando el espacio 3D con OpenGL. HektorDocs. Recuperado el 28 de marzo de 2024 de: https://shorturl.at/sBHS9
• Anonimo. (2024). Camera. Learn OpenGL. Recuperado el 28 de febrero de 2024 de: https://shorturl.at/kBFZ9
• Saltwash in Technology. (2015). Camera pose using OpenCV and OpenGL. WordPress. Recuperado el 28 de marzo de 2024 de: https://shorturl.at/osDNO
• Anónimo. (2018). Tutorial 6: Teclado y ratón. OpenGL-Tutorial. Recuperado el 27 de marzo de 2024 de: https://shorturl.at/pryF3
• Agusti Melchor. M. (2017). Ejemplos de aplicaciones 3D interactivas con OpenGL. Universidad Politécnica de Valencia. Recuperado el 27 de marzo de 2024 de: https://shorturl.at/ksHMQ
• Anonimo. (2022). Codigo de colores para la clasificación de basura. Reko. Recuperado el 3 de abril de 2024 de: https://shorturl.at/qCGX7
• García-Bullé, S. (2019). Videojuegos: una herramienta educativa en potencia. Observatorio. Recuperado el 3 de abril de 2024 de: https://shorturl.at/yOY79
• Guzmán H., C. (2022). Matrices de transformación, Hector Docs. Recuperado el 14 de abril de 2024 de: https://shorturl.at/egsW6
• López T., B. (2020). Métodos de tratamiento de colisiones. TecNM. Recuperado el 16 de abril de 2024 de: https://shorturl.at/iHY59

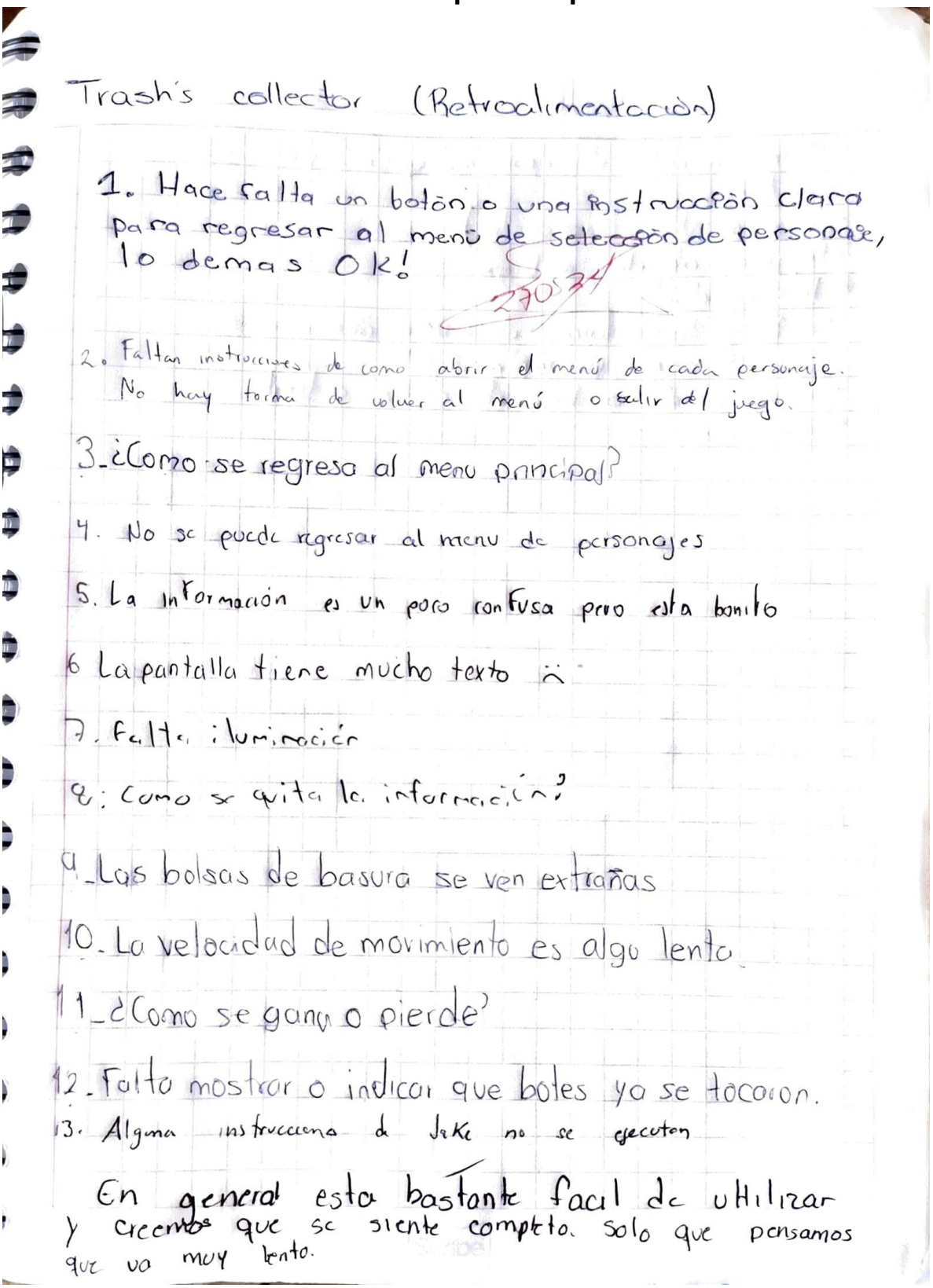
# Anexos

## Bocetos





## Retroalimentación en clase tras primera prueba



Trash's collector (Retroalimentación)

1. Hace falta un botón o una instrucción clara para regresar al menú de selección de personaje, lo demas Ok!

270:34

2. Faltan instrucciones de como abrir el menú de cada personaje. No hay forma de volver al menú o salir del juego.

3. ¿Como se regresa al menu principal?

4. No se puede regresar al menu de personajes

5. La información es un poco confusa pero esta bonito

6 La pantalla tiene mucho texto :(

7. Falta iluminación

8: Como se quita la información?

9. Las bolsas de basura se ven extrañas

10. La velocidad de movimiento es algo lenta

11. ¿Como se gana o pierde?

12. Falto mostrar o indicar que botes ya se tocaron.

13. Alguna instruccion de Jake no se ejecuten

En general esta bastante facil de uHilizar y creemos que se siente completo. solo que pensamos que va muy lento.

## Revisiones de nivel

**Revisión Nivel 1 –** Revisión 1.docx

**Revisión Nivel 2 -** Revision 2.docx

**Revisión Nivel 3 –** Revisión 3.docx