



Entrega del Reto. Ciclo 2.

Nombre del proyecto: Viaje Intermatemático

Equipo 3

Nicole Kapellmann Lepine A01664563

Arely Yael Villatoro Amador A01663303

Pablo Ricardo Duran Sanchez A01663878

Dana Mendoza Palacios Roji A01658253

2/05/2024

Profesores: Sergio Ruiz, Christian Rolando & José Angel Martínez

Construcción de Software y Toma de Decisiones (Grupo 442)

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México

INTRODUCCIÓN

Definición del problema:

Para abordar la necesidad de fortalecer los temas de Matemáticas en todos los niveles de primaria, resulta necesario implementar actividades didácticas interactivas y lúdicas. Estas estrategias no solo permiten a los alumnos repasar los temas de manera más dinámica y atractiva, sino que también fomentan un mayor compromiso y participación por parte de los estudiantes. Al integrar elementos de este estilo, se crea un ambiente propicio para el aprendizaje activo, donde los alumnos pueden explorar conceptos matemáticos de manera práctica.

Contexto mundial:

A nivel mundial, los desafíos en la enseñanza y aprendizaje de las matemáticas en la educación primaria son variados y se enfrentan en diferentes contextos educativos. Según la UNESCO el 56% de los niños en educación primaria no alcanzan los conocimientos mínimos de matemáticas. Algunos países han implementado enfoques innovadores en la enseñanza de las matemáticas, centrándose en la comprensión conceptual y el desarrollo de habilidades prácticas, mientras que otros aún se adhieren a métodos más tradicionales. La necesidad de mejorar la competencia matemática desde una edad temprana es reconocida como crucial para el éxito educativo y profesional en el futuro.

Contexto en México:

De acuerdo con la aplicación de la prueba de aprendizaje del Plan Nacional para la Evaluación de los Aprendizajes (Planea) se estima que “el 65% de alumnos de secundaria son incapaces de solucionar problemas matemáticos de quinto grado de primaria y solo 14% de secundaria” (Staff, 2022). Esta baja comprensión de las matemáticas nace desde la educación primaria, y deben ser atendidas desde ese entonces o impedirá el aprendizaje de conceptos más avanzados.

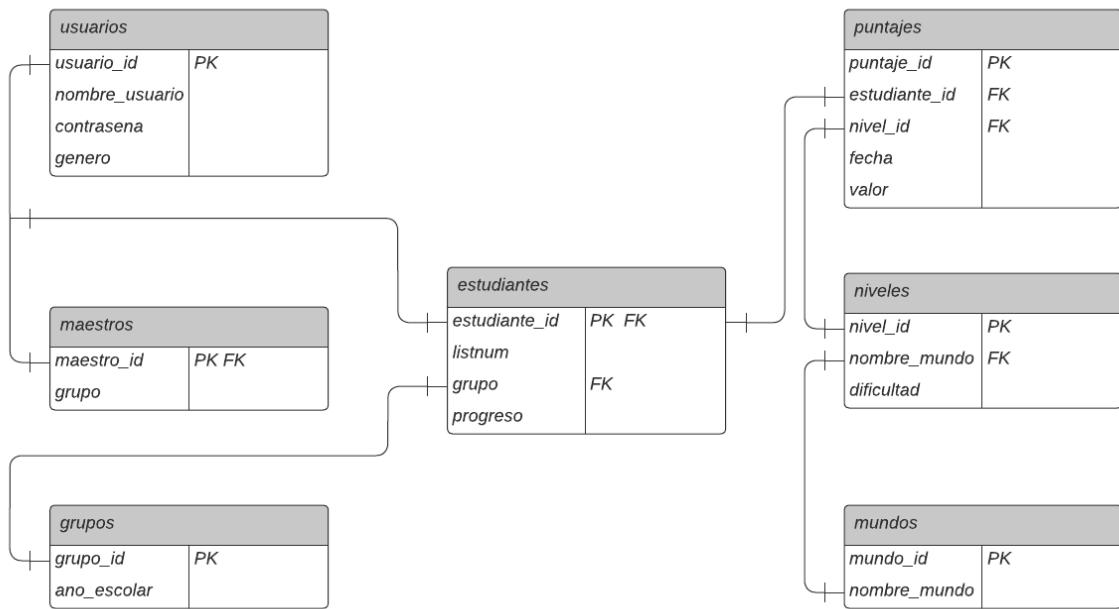
1. MODULO 1

1.1 Modelo relacional de la base de datos.

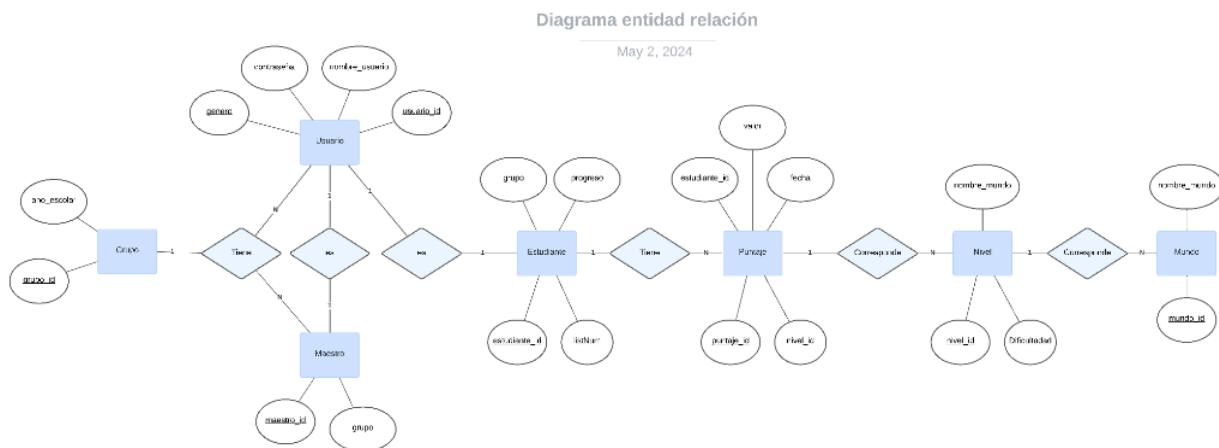
1.1.1 MER

Diagrama ER para base de datos

Ricardo Duran | April 24, 2024



1.1.2 DER



1.2 APIs

Endpoint: /dashboard/maestros

Descripción

El endpoint `/dashboard/maestros` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es recopilar y

combinar una variedad de estadísticas y datos relacionados con los maestros, estudiantes, puntajes, avances globales, avances por género, estadísticas globales de jugadores y grupos, y luego devolver estos datos combinados al cliente en formato JSON. Aquí se detallan sus funciones y uso:

- **Recopilación de Datos:** Al recibir una solicitud, el endpoint invoca varios métodos de controladores específicos (`MaestroController`, `EstudianteController`, `PuntajeController`, `MundoController`, `GrupoController`) para recopilar datos relevantes. Esto incluye listar maestros, estudiantes, los puntajes más altos, el avance global de los mundos completados, el avance por género, estadísticas globales de jugadores y listar grupos y puntajes.
- **Combinación de Datos:** Los datos recopilados de cada controlador se combinan en una única lista de listas (`estadisticas_combinadas`). Esta estructura de datos permite agrupar y organizar los diferentes conjuntos de datos de manera coherente.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve los datos combinados al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```
@app.route("/dashboard/maestros", methods=["GET", "POST"])
def datos_dashboard_maestros():
    maestros = MaestroController().listarMaestros()
    estudiantes = EstudianteController().listarEstudiantes()
    toppuntajes = PuntajeController().top_puntajes()
    avance_global = MundoController().mundos_completados()
    avance_genero = EstudianteController().listarEstudiantesGenero()
    estadisticas_jugadores = EstudianteController().estadisticas_globales()
    grupos = GrupoController().listarGrupos()
    puntajes = PuntajeController().listarPuntajes()

    # Combina los resultados en una lista de listas
    estadisticas_combinadas = [
        maestros,
        estudiantes,
        toppuntajes,
        avance_global,
        avance_genero,
        estadisticas_jugadores,
        grupos,
        puntajes,
    ]

    # Retorna los resultados como JSON
    return jsonify(estadisticas_combinadas)
```

Endpoint: /dashboard/estudiante

Descripción

El endpoint `/dashboard/estudiante` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es proporcionar información detallada sobre un estudiante específico, incluyendo sus datos personales y sus puntajes, basándose en el `estudiante_id` proporcionado en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato JSON. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de `estudiante_id`:** El endpoint extrae el `estudiante_id` del cuerpo de la solicitud. Este identificador único es necesario para buscar y recuperar la información del estudiante correspondiente.
- **Creación del Objeto Estudiante:** Con el `estudiante_id` obtenido, se crea un objeto `Estudiante` que se utilizará para buscar la información del estudiante en el sistema.
- **Recuperación de Información del Estudiante:** Se invocan dos métodos del controlador `EstudianteController`: `datos_de_estudiante` para obtener los datos personales del estudiante y `puntajes_de_estudiante` para obtener los puntajes asociados al estudiante. Ambos métodos retornan objetos que luego se serializan para convertirlos en formatos adecuados para la respuesta JSON.
- **Preparación de la Respuesta:** Los datos del estudiante y sus puntajes se combinan en un diccionario (`resultado`), que luego se convierte en una respuesta JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```

@app.route("/dashboard/estudiante", methods=["GET", "POST"])
def datos_dashboard_alumnos():
    data = request.json
    if data is None:
        return jsonify({"message": "Datos faltantes"})

    estudiante_id = data.get("estudiante_id")

    if estudiante_id is None:
        return jsonify({"message": "Datos faltantes"})

    estudiante = Estudiante(estudiante_id=estudiante_id)

    info_estudiante = EstudianteController().datos_de_estudiante(estudiante).serialize()
    info_puntajes = [
        puntaje.serialize()
        for puntaje in EstudianteController().puntajes_de_estudiante(estudiante)
    ]
    # Crear un diccionario con las variables info_estudiante y puntaje_estudiante
    resultado = {"estudiante_info": info_estudiante, "puntajes": info_puntajes}

    # Retorna los resultados como JSON
    return jsonify(resultado)

```

Endpoint: /usuario/crear

Descripción

El endpoint `/usuario/crear` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es crear un nuevo usuario en el sistema, permitiendo que el usuario sea un estudiante o un maestro, dependiendo de si se proporciona un `listNum` en la solicitud. Aquí se detallan sus funciones y uso:

Recepción de Datos: Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato JSON. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.

Extracción de Datos del Usuario: El endpoint extrae varios campos del cuerpo de la solicitud, incluyendo `nombre_usuario`, `contrasena`, `genero`, `grupo`, y opcionalmente `listNum`. Estos campos son necesarios para crear un nuevo usuario y determinar si será un estudiante o un maestro.

Creación del Objeto Usuario: Con los datos obtenidos, se crea un objeto **Usuario** que se utilizará para intentar crear un nuevo usuario en el sistema.

Intento de Creación del Usuario: Se invoca el método `crearUsuario` del controlador **UsuarioController**, pasando el objeto **Usuario** creado. Este método intenta crear un nuevo usuario en el sistema y retorna un **id** único para el usuario creado.

Verificación de la Creación del Usuario: Si la creación del usuario es exitosa (es decir, el **id** retornado no es **False**), el endpoint verifica si se proporcionó un **listNum**. Si se proporcionó, se asume que el usuario es un estudiante y se crea un objeto **Estudiante** con el **id** del usuario, **listNum**, y **grupo** proporcionados. Si no se proporcionó **listNum**, se asume que el usuario es un maestro y se crea un objeto **Maestro** con el **id** del usuario y **grupo** proporcionados.

Creación del Estudiante o Maestro: Dependiendo de si se proporcionó **listNum**, se invoca el método `crearEstudiante` del controlador **EstudianteController** o `crearMaestro` del controlador **MaestroController**, pasando el objeto **Estudiante** o **Maestro** creado. Estos métodos intentan crear un nuevo estudiante o maestro en el sistema y retornan los datos del estudiante o maestro creado en formato JSON.

Manejo de Errores: Si la creación del usuario no es exitosa, se devuelve un mensaje de error en formato JSON indicando que hubo un error al crear el usuario.

```

@app.route("/usuario/crear", methods=["GET", "POST"]) # WEB
def usuario_crear():
    data = request.json
    if data is None:
        return jsonify({"message": "Datos faltantes"})

    nombre_usuario = data.get("nombre_usuario")
    contrasena = data.get("contrasena")
    genero = data.get("genero")
    grupo = data.get("grupo")
    listNum = data.get("listNum")

    if nombre_usuario is None or contrasena is None or genero is None:
        return jsonify({"message": "Datos faltantes"})

    user = Usuario(nombre_usuario=nombre_usuario, contrasena=contrasena, genero=genero)
    id = UsuarioController().crearUsuario(user)

    if id is not False: # Verifica si la creación del usuario fue exitosa
        if listNum is not None:
            estudiante = Estudiante(estudiante_id=id, listNum=listNum, grupo=grupo)
            return jsonify(EstudianteController().crearEstudiante(estudiante))
        else:
            maestro = Maestro(maestro_id=id, grupo=grupo)
            return jsonify(MaestroController().crearMaestro(maestro))
    else:
        return jsonify({"error": "Error al crear el usuario ", "code": -1})

```

Endpoint: /usuario/validar

Descripción

El endpoint `/usuario/validar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es validar la información de un usuario existente en el sistema, basándose en el `nombre_usuario` y la `contrasena` proporcionados en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato JSON. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Usuario:** El endpoint extrae los campos `nombre_usuario` y `contrasena` del cuerpo de la solicitud. Estos campos son necesarios para identificar y autenticar al usuario en el sistema.
- **Verificación de Datos:** Antes de proceder con la validación, el endpoint verifica si alguno de los datos (`nombre_usuario` o `contrasena`) es `None`. Si alguno de los campos está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- **Creación del Objeto Usuario:** Con los datos obtenidos, se crea un objeto `Usuario` que se utilizará para intentar validar al usuario en el sistema.
- **Determinación del Tipo de Usuario:** Se invoca el método `tipo` del controlador `UsuarioController`, pasando el objeto `Usuario` creado. Este método determina el tipo de usuario (por ejemplo, estudiante, maestro) basándose en la información del usuario.
- **Validación del Usuario:** Se invoca el método `validar` del controlador `UsuarioController`, pasando el objeto `Usuario` creado. Este método realiza la validación del usuario, verificando si el `nombre_usuario` y la `contrasena` proporcionados coinciden con los registros existentes en el sistema.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve el tipo de usuario y el resultado de la validación al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```

@app.route("/usuario/validar", methods=["GET", "POST"]) # WEB
def usuario_validar():
    # Acceder a los datos enviados como JSON
    data = request.json
    if data is None:
        return jsonify({"message": "Datos faltantes"})

    # Extraer los valores de 'nombre_usuario' y 'contrasena' del JSON
    nombre_usuario = data.get("nombre_usuario")
    contrasena = data.get("contrasena")

    # Verificar si los datos requeridos están presentes
    if nombre_usuario is None or contrasena is None:
        return jsonify({"message": "Datos faltantes"})

    # Crear el objeto User
    user = Usuario(nombre_usuario=nombre_usuario, contrasena=contrasena)
    tipo = UsuarioController().tipo(user)

    return jsonify(tipo, UsuarioController().validar(user))

```

Endpoint: /maestro/modificar

Descripción

El endpoint `/maestro/modificar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es modificar la información de un maestro específico en el sistema, permitiendo actualizar el `grupo` del maestro basándose en los datos proporcionados en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato JSON. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Maestro:** El endpoint extrae los campos `maestro_id` y `grupo` del cuerpo de la solicitud. Estos campos son necesarios para identificar al maestro específico cuya información se desea modificar y para determinar el nuevo valor para `grupo`.

- **Verificación de Datos:** Antes de proceder con la modificación de la información del maestro, el endpoint verifica si alguno de los datos (`maestro_id` o `grupo`) es `None`. Si alguno de los campos está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- **Creación del Objeto Maestro:** Con los datos obtenidos, se crea un objeto `Maestro` que se utilizará para intentar modificar la información del maestro en el sistema.
- **Modificación de la Información del Maestro:** Se invoca el método `modificarMaestro` del controlador `MaestroController`, pasando el objeto `Maestro` creado. Este método realiza la modificación de la información del maestro, actualizando el registro del maestro con el nuevo valor de `grupo` proporcionado.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve el resultado de la modificación de la información del maestro al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```

@app.route("/maestro/modificar", methods=["GET", "POST"]) # UNITY
def maestro_modificar():
    # Acceder a los datos enviados como formulario
    data = request.json
    if not data:
        return jsonify({"message": "Datos faltantes"})

    maestro_id = data.get("maestro_id")
    grupo = data.get("grupo")

    if maestro_id is None:
        return jsonify({"message": "Datos faltantes"})

    maestro = Maestro(maestro_id=maestro_id, grupo=grupo)

    return jsonify(MaestroController().modificarMaestro(maestro))

```

Endpoint:/grupos/listar

Descripción

El endpoint `/grupos/listar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es listar todos los grupos disponibles en el sistema, devolviendo esta información en formato JSON. Aquí se detallan sus funciones y uso:

- **Invocación del Controlador:** Al recibir una solicitud, el endpoint invoca el método `listarGrupos` del controlador `GrupoController`. Este método es responsable de recuperar y preparar la lista de grupos para ser devuelta al cliente.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve la lista de grupos en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```
@app.route("/grupos/listar", methods=["GET", "POST"]) # WEB
def listar_grupos():
    return jsonify(GrupoController().listarGrupos())
```

Endpoint:/grupo/agregar

Descripción

El endpoint `/grupo/agregar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es agregar un nuevo grupo al sistema, permitiendo a los usuarios o sistemas externos introducir un nuevo grupo mediante la provisión de un `grupo_id` en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato JSON. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Grupo:** El endpoint extrae el campo `grupo_id` del cuerpo de la solicitud. Este campo es necesario para identificar el nuevo grupo que se desea agregar al sistema.

- **Verificación de Datos:** Antes de proceder con la adición del grupo, el endpoint verifica si el `grupo_id` es `None`. Si el campo está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- **Creación del Objeto Grupo:** Con el `grupo_id` obtenido, se crea un objeto `Grupo` que se utilizará para intentar agregar el nuevo grupo al sistema.
- **Adición del Grupo:** Se invoca el método `agregarGrupo` del controlador `GrupoController`, pasando el objeto `Grupo` creado. Este método realiza la adición del nuevo grupo al sistema, almacenando el `grupo_id` proporcionado.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve el resultado de la adición del grupo al sistema al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```
@app.route("/grupo/agregar", methods=["GET", "POST"]) # WEB
def agregar_grupos():
    data = request.json
    if data is None:
        return jsonify({"message": "Datos faltantes"})
    # Extraer los valores del JSON
    grupo_id = data.get("grupo_id")

    if grupo_id is None:
        return jsonify({"message": "Datos faltantes"})

    grupo = Grupo(grupo_id=grupo_id)
    return jsonify(GrupoController().agregarGrupo(grupo))
```

Endpoint:/grupo/eliminar

Descripción

El endpoint `/grupo/eliminar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es eliminar un grupo

existente del sistema, permitiendo a los usuarios o sistemas externos especificar un `grupo_id` en la solicitud para identificar el grupo que se desea eliminar. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato JSON. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Grupo:** El endpoint extrae el campo `grupo_id` del cuerpo de la solicitud. Este campo es necesario para identificar el grupo específico que se desea eliminar del sistema.
- **Verificación de Datos:** Antes de proceder con la eliminación del grupo, el endpoint verifica si el `grupo_id` es `None`. Si el campo está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos específicos.
- **Creación del Objeto Grupo:** Con el `grupo_id` obtenido, se crea un objeto `Grupo` que se utilizará para intentar eliminar el grupo del sistema.
- **Eliminación del Grupo:** Se invoca el método `eliminarGrupo` del controlador `GrupoController`, pasando el objeto `Grupo` creado. Este método realiza la eliminación del grupo del sistema, eliminando el registro asociado al `grupo_id` proporcionado.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve el resultado de la eliminación del grupo al sistema al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```
@app.route("/grupo/eliminar", methods=["GET", "POST"]) # WEB
def eliminar_grupos():
    data = request.json
    if data is None:
        return jsonify({"message": "Datos faltantes"})
    # Extraer los valores del JSON
    grupo_id = data.get("grupo_id")

    if grupo_id is None:
        return jsonify({"message": "Datos faltantes específicos"})

    grupo = Grupo(grupo_id=grupo_id)
    return jsonify(GrupoController().eliminarGrupo(grupo))
```

2. MODULO 2

2.1 Requerimientos funcionales

2.1.1 Del Videojuego

| Identificador del requerimiento | Descripción | Estado |
|---------------------------------|--|------------|
| RFV01 | El videojuego permitirá a los usuarios jugar y avanzar a través de diferentes niveles y desafíos relacionados con conceptos matemáticos, como áreas y volúmenes, fracciones, sumas y restas. | Completado |
| RFV02 | El videojuego permitirá a los usuarios jugar y avanzar a través de diferentes niveles y desafíos relacionados con conceptos matemáticos, como áreas y volúmenes, fracciones, sumas y restas. | Completado |
| RFV03 | El videojuego proporcionará una interfaz de usuario intuitiva que permitirá a los niños seleccionar y jugar diferentes actividades matemáticas de forma fácil y divertida. | Completado |
| RFV04 | Antes de cada actividad, el videojuego verificará que los conceptos matemáticos abordados en el nivel no sean demasiado avanzados para el niño, ajustando la dificultad según el progreso y habilidades del jugador. | Completado |
| RFV05 | El videojuego permitirá a los usuarios registrar y gestionar sus perfiles de jugador, incluyendo opciones para personalizar avatares, ver estadísticas de juego y guardar el progreso. | Completado |
| RFV06 | El videojuego ofrecerá retroalimentación inmediata y positiva cuando los niños completen actividades correctamente, utilizando animaciones, efectos de sonido y recompensas visuales para reforzar el aprendizaje. | Completado |
| RFV07 | El videojuego incluirá una variedad de actividades y mini-juegos matemáticos que mantendrán la experiencia de juego fresca y emocionante, fomentando así el interés y la participación continua. | Completado |
| RFV08 | El videojuego incluirá una función de seguimiento del progreso del jugador, que permitirá a los padres y maestros monitorear el rendimiento y el tiempo dedicado al juego, así como identificar áreas de mejora. | Completado |

2.1.2 De la web

| Identificador del requerimiento | Descripción | Estado |
|--|--|---------------|
| RFW01 | Permitirá a los usuarios (alumnos) acceder a sus perfiles personales mediante un inicio de sesión seguro. | Completado |
| RFW02 | Mostrará a los usuarios (alumnos) un resumen visual de su progreso individual, incluyendo estadísticas de rendimiento en actividades y juegos educativos. | Completado |
| RFW03 | Permitirá a los usuarios (alumnos) ver un registro detallado de las actividades y lecciones completadas, así como los puntajes obtenidos en cada una. | Completado |
| RFW04 | Ofrecerá a los usuarios (maestros) acceso a un panel de control centralizado donde puedan ver el progreso y desempeño del grupo de alumnos. | Completado |
| RFW05 | Permitirá a los usuarios (maestros) generar informes personalizados sobre el rendimiento de los alumnos, incluyendo estadísticas de grupo y comparaciones individuales. | Completado |
| RFW06 | Facilitará a los usuarios (maestros) la comunicación con los alumnos a través de la plataforma, permitiendo el envío de mensajes y recordatorios relacionados con el progreso académico. | Completado |
| RFW07 | Integrará funciones de seguimiento de objetivos y logros, donde tanto los alumnos como los maestros puedan establecer y monitorear el progreso hacia metas específicas de aprendizaje. | Completado |
| RFW08 | Integrará funciones de seguimiento de objetivos y logros, donde tanto los alumnos como los maestros puedan establecer y monitorear el progreso hacia metas específicas de aprendizaje. | Completado |

2.2 Requerimientos no funcionales

2.2.1 Del videojuego

| Identificador del requerimiento | Descripción | Estado |
|--|--|---------------|
| RNFV01 | El videojuego deberá tener una interfaz intuitiva y atractiva que sea fácil de entender y utilizar para niños de diferentes edades y niveles de habilidad. | Completado |

| | | |
|--------|--|------------|
| RNFV02 | Los temas del videojuego tendrán que ser investigados previamente para que se ajusten a las necesidades del cliente | Completado |
| RNFV03 | Los datos personales de los usuarios, como los perfiles de jugador y el progreso del juego, deberán estar protegidos mediante medidas de seguridad adecuadas, como la encriptación de datos y el acceso restringido. | Completado |
| RNFV04 | El acceso a funciones administrativas del videojuego, como la gestión de perfiles de jugador y estadísticas de juego, deberá estar restringido y requerirá autenticación adecuada. | Completado |
| RNFV05 | El videojuego deberá poder actualizarse para añadir más temas si es que se necesita en un futuro. | Completado |
| RNFV06 | El videojuego tendrá una temática amigable para la audiencia a la que va dirigida. | Completado |
| RNFV07 | Los tiempos de respuesta del videojuego para acciones como seleccionar actividades, cargar niveles y guardar progreso deberán ser mínimos, asegurando una experiencia de juego receptiva y fluida. | Completado |
| RNFV08 | El videojuego deberá ser capaz de manejar un gran número de usuarios concurrentes sin experimentar caídas en el rendimiento o tiempos de espera prolongados. | Completado |
| RNFV09 | El videojuego deberá estar disponible para jugar las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad mínimo para mantenimiento programado. | Completado |
| RNFV10 | Se implementarán medidas de recuperación de datos para garantizar la disponibilidad continua del progreso del jugador en caso de fallos del sistema. | Completado |
| RNFV11 | La arquitectura del videojuego será escalable para adaptarse a un aumento en el número de usuarios y funciones en el futuro. | Completado |
| RNFV12 | El videojuego será compatible con una variedad de dispositivos y sistemas operativos, incluyendo PC, consolas y dispositivos móviles, para maximizar su accesibilidad. | Completado |
| RNFV13 | Se realizarán pruebas exhaustivas de compatibilidad para garantizar que el videojuego funcione correctamente en diferentes configuraciones de hardware y software. | Completado |
| RNFV14 | El código del videojuego seguirá las mejores prácticas de programación y estará bien documentado para facilitar futuras actualizaciones y mantenimiento. | Completado |

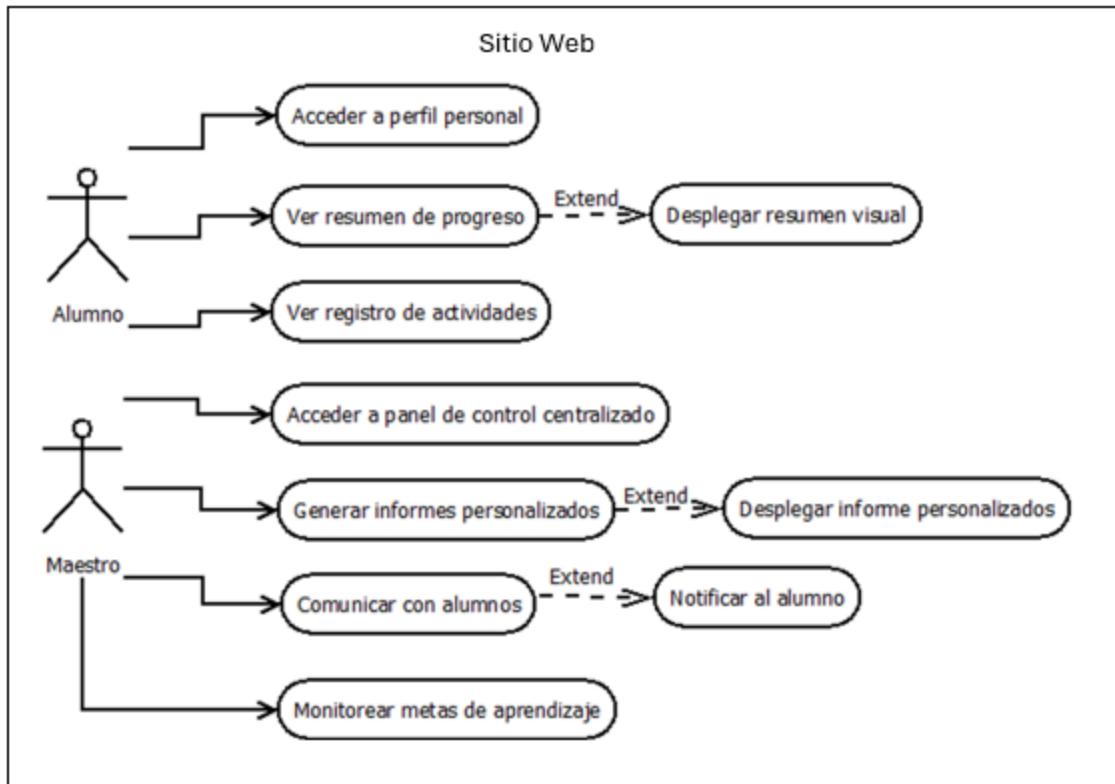
| | | |
|--------|---|------------|
| RNFV15 | Se establecerán procesos y herramientas adecuadas para la gestión de versiones y el control de cambios en el desarrollo del videojuego. | Completado |
|--------|---|------------|

2.2.2 De la web

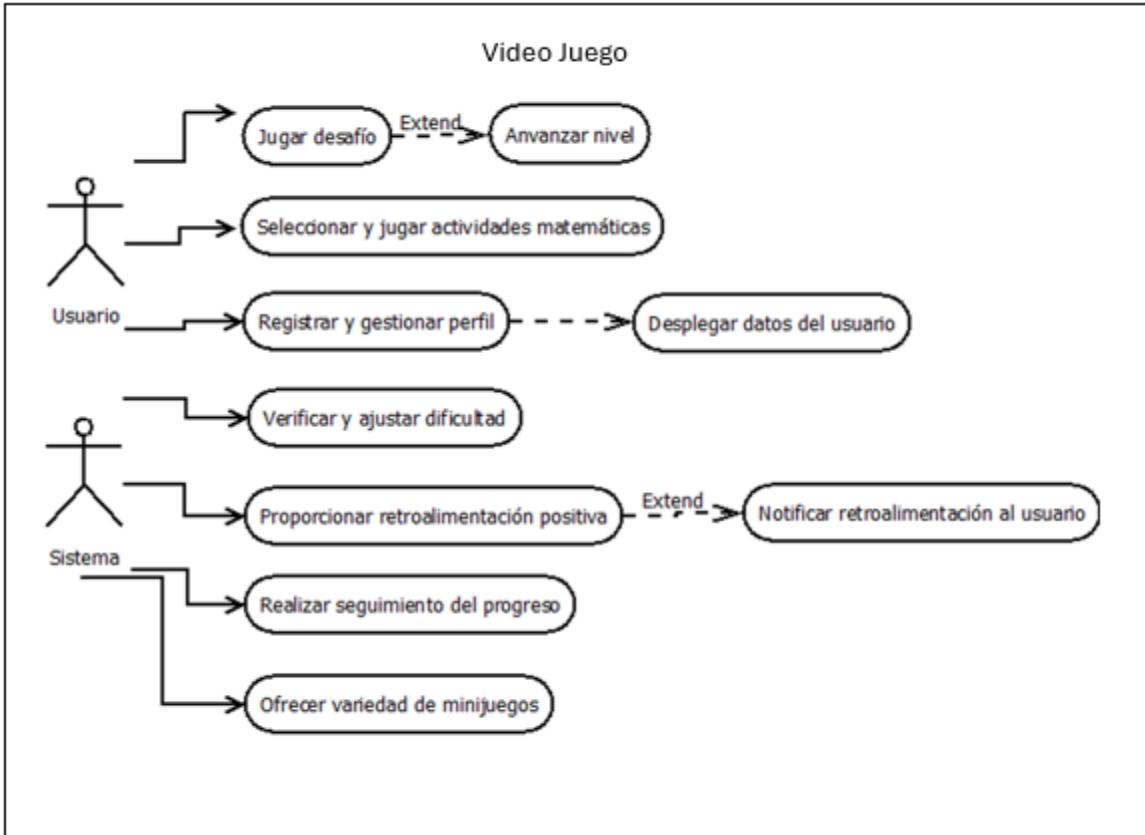
| Identificador del requerimiento | Descripción | Estado |
|---------------------------------|--|------------|
| RNFW01 | La página web deberá ser fácil de navegar y usar tanto para alumnos como para maestros, con una interfaz intuitiva y amigable. | Completado |
| RNFW02 | Los tiempos de carga de la página y la visualización de datos deberán ser rápidos y eficientes, no mayor a 5 segundos, para proporcionar una experiencia fluida y sin interrupciones. | Completado |
| RNFW03 | El sistema deberá ser capaz de manejar múltiples sesiones de usuarios simultáneas sin experimentar retrasos en la respuesta o caídas en el rendimiento. | Completado |
| RNFW04 | La seguridad de los datos de los usuarios deberá ser una prioridad, utilizando medidas de protección adecuadas como cifrado de datos y permisos de usuario. | Completado |
| RNFW05 | La página web deberá estar disponible las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad mínimo programado para mantenimiento. | Completado |
| RNFW06 | Se deberán implementar medidas de respaldo y recuperación de datos para garantizar la integridad y disponibilidad de la información del progreso de los alumnos. | Completado |
| RNFW07 | El diseño de la página web deberá ser responsive y compatible con una variedad de dispositivos y navegadores web, para garantizar una experiencia consistente en diferentes plataformas. | Completado |
| RNFW08 | Se deberá establecer procedimientos de autenticación seguros para garantizar que solo los usuarios autorizados tengan acceso a la información del progreso de los alumnos. | Completado |

2.3 Casos de uso

2.3.1 Web

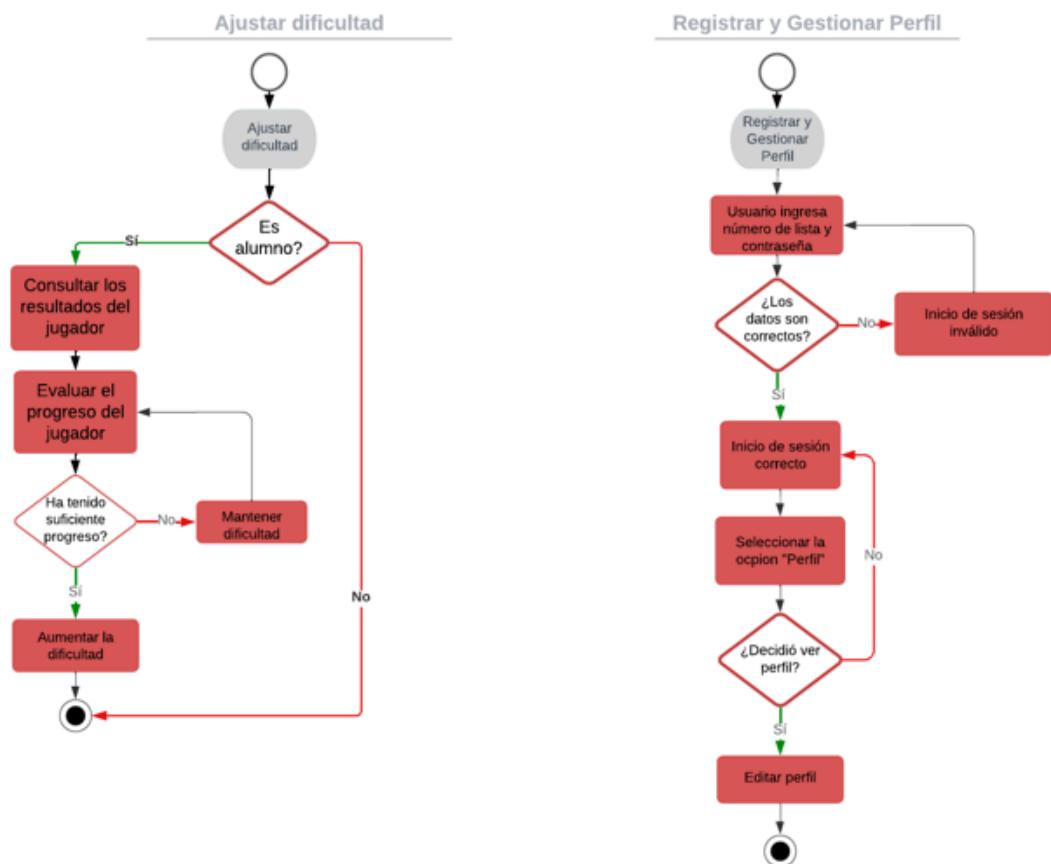


2.3.2 Videojuego

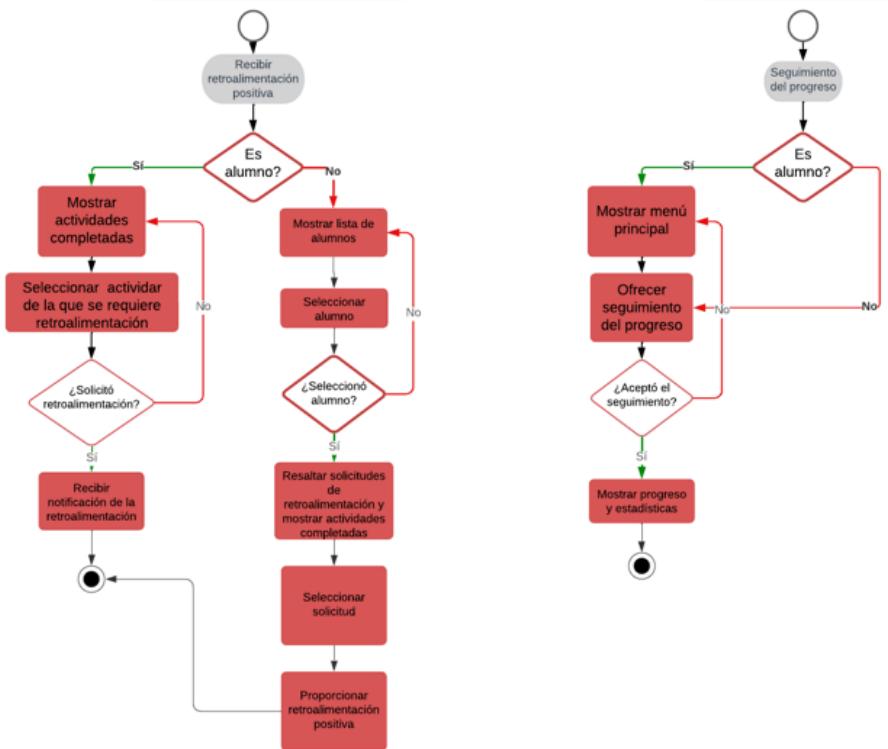


2.4 Diagramas de actividad

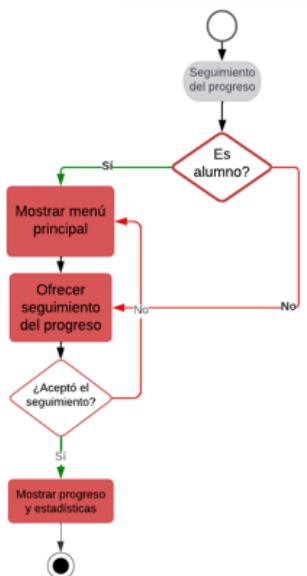
2.4.1 Web



Recibir retroalimentación positiva



Seguimiento del proceso

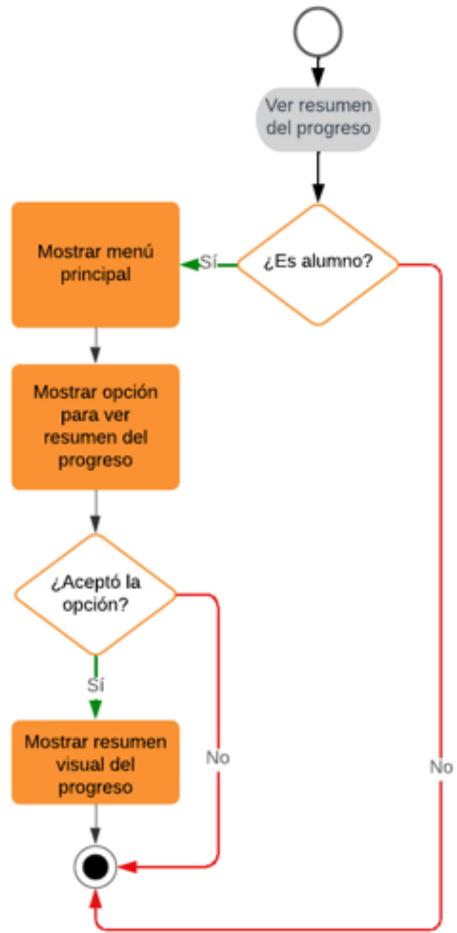


2.4.2 Videojuego

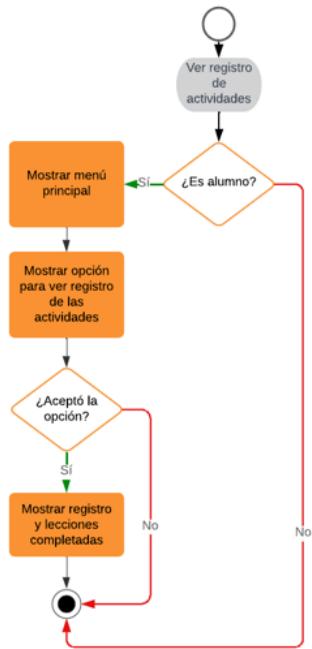
Acceder al perfil



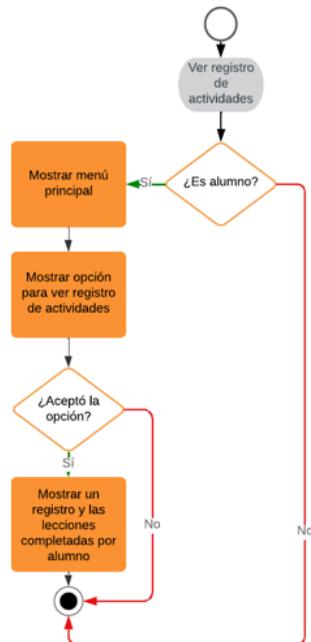
Ver resumen del progreso



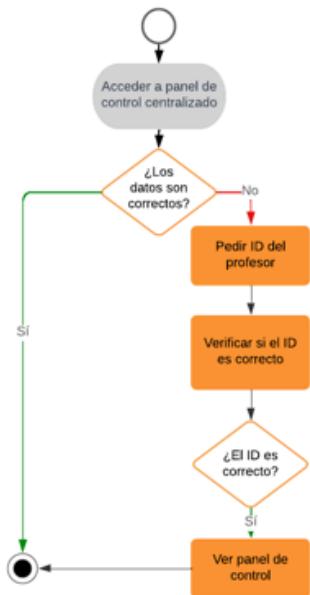
Ver registro de actividades



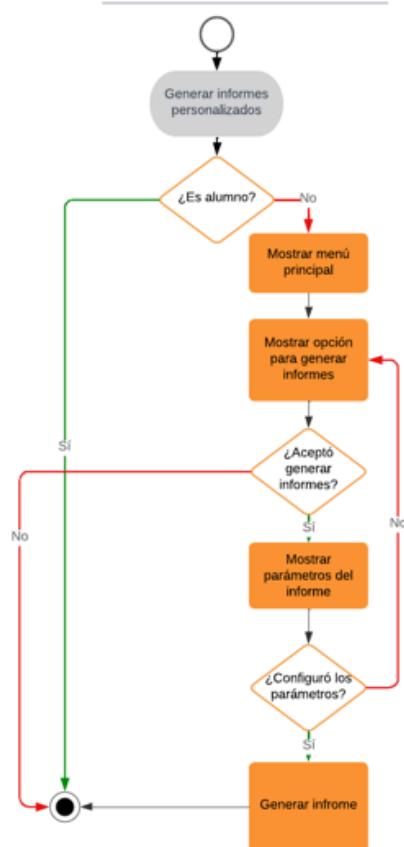
Ver registro de actividades

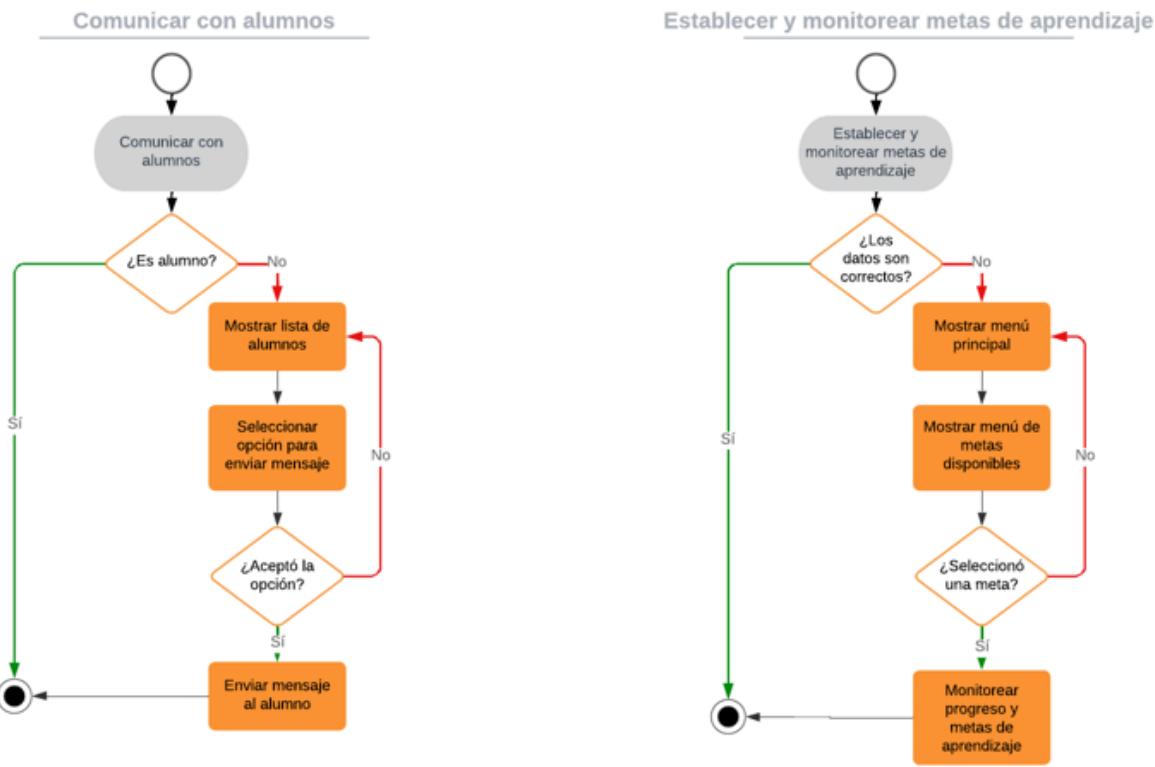


Acceder a panel de control centralizado



Generar informes personalizados

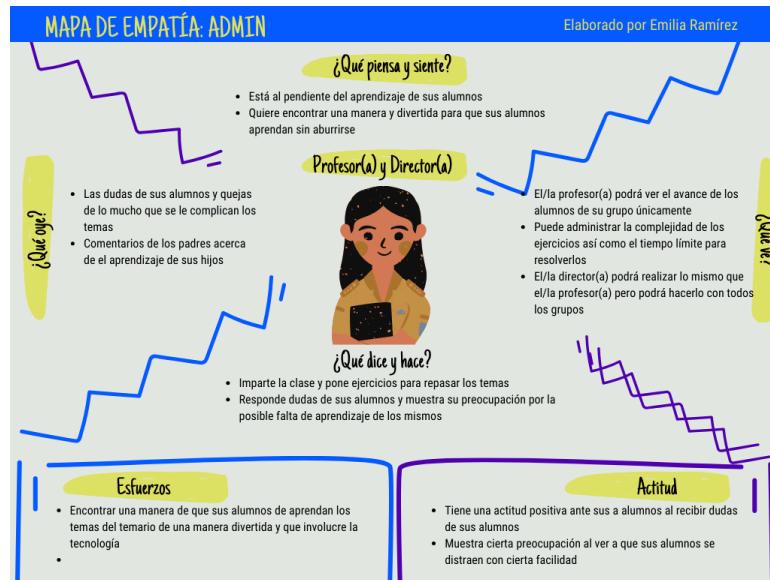




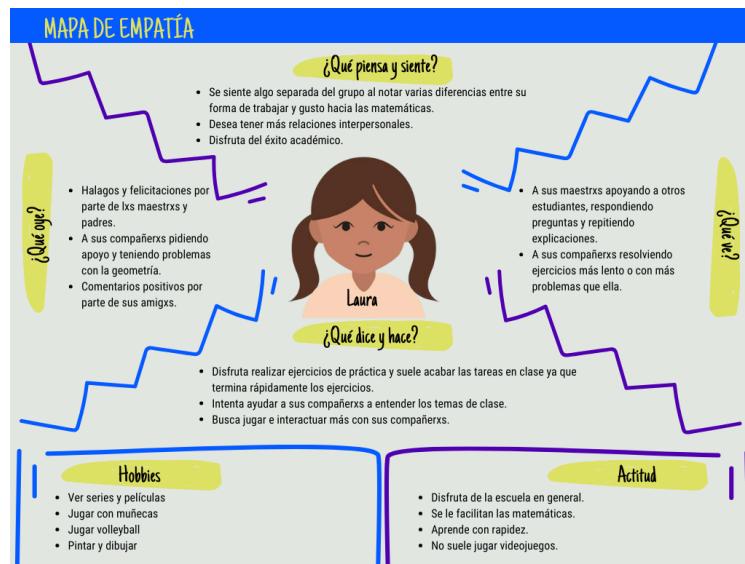
2.5 Historias de usuario

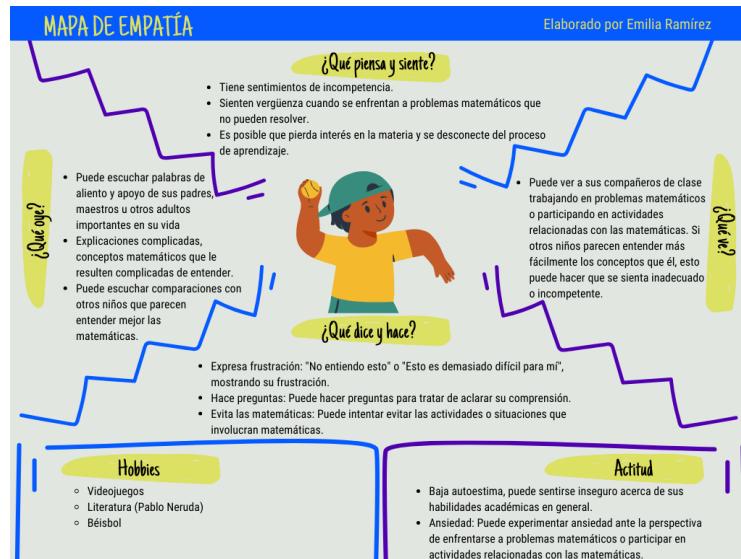
2.5.1 Web





2.5.1 Videojuego





2.6 Journey Maps

2.6.1 Web

| Web Experience Map | | | |
|--|--|--|---|
|  <p>Laura Maestra de Sto de primaria con pasión por la enseñanza</p> <p>Edad: 42 Género: Femenino Ocupación: Maestra</p> | | | |
| Fases | Login | Preview general | Vista filtrada |
| Necesidades | <ul style="list-style-type: none"> - Que ingrese a su cuenta fácil y rápidamente | <ul style="list-style-type: none"> - Que la información sea fácil de entender y manipular - Poder obtener información sobre el aprendizaje de sus alumnos | <ul style="list-style-type: none"> - Poder ver la información y avances de ciertos alumnos en específico - Poder ver la información del desempeño de sus alumnos por temas en específico |
| Acciones | <ul style="list-style-type: none"> - Da click en maestra - Ingresa sus datos de inicio de sesión | <ul style="list-style-type: none"> - Ver las gráficas del desempeño de sus alumnos - Ver el tiempo de juego de sus alumnos - Ver el desempeño de sus alumnos por tema/nivel | <ul style="list-style-type: none"> - Selecciona los filtros a aplicar - Ve los puntajes que pasen el filtro - Ver datos sobre el desempeño de sus alumnos en los temas seleccionados |
| Herramientas | <ul style="list-style-type: none"> - Computadora - Internet | <ul style="list-style-type: none"> - Computadora - Internet | <ul style="list-style-type: none"> - Computadora - Internet |
| Storyboard |  |  |  |
| Experiencia | 😊 😐 😢 |  | |
| Problemas | <ul style="list-style-type: none"> • Mala conexión • Problemas al iniciar sesión | <ul style="list-style-type: none"> • Problemas para visualizar las gráficas • Dispositivo lento • Dificultad para entender la interfaz | <ul style="list-style-type: none"> • Que la aplicación de filtros sea compleja de entender • Que las queries tarden tiempo en completarse |
| Ganancias | <ul style="list-style-type: none"> - Puede acceder a las estadísticas | <ul style="list-style-type: none"> - Aprendió sobre las fortalezas y áreas de oportunidad de sus alumnos | <ul style="list-style-type: none"> - Logró identificar las áreas de oportunidad para casos específicos - Logró ver el desempeño de sus alumnos en cada tema |

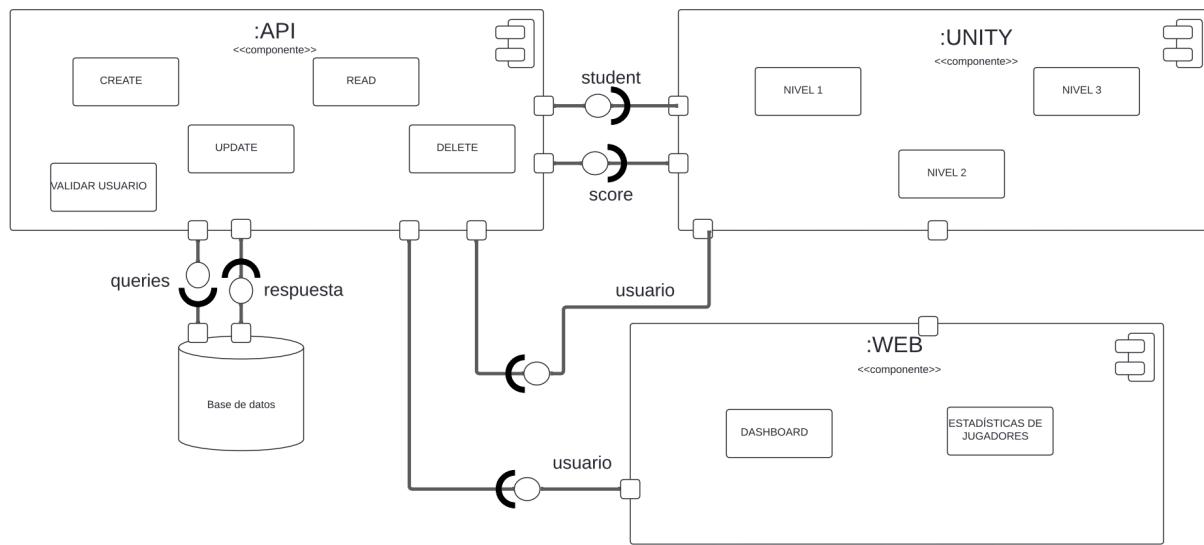
| <h1 style="color: white; margin: 0;">Web Experience Map</h1> <div style="background-color: #0070C0; padding: 5px; display: flex; align-items: center;"> <div style="flex-grow: 1; text-align: center;"> <p>Diego Suárez</p> <p>Estudiante de 5to de primario, motivado por encontrar nuevas maneras de aprendizaje y sus logros.</p> <p>Edad: 10 Género: Masculino Ocupación: Estudiante</p> </div> </div> | | | |
|---|---|---|---|
| Fases | Login | Preview general | Vista filtrada |
| Necesidades | <ul style="list-style-type: none"> - Entender fácilmente cómo entrar correctamente a su cuenta - Recordar rápidamente sus datos de inicio | <ul style="list-style-type: none"> - Obtener una visualización amplia de su desempeño, casi inmediatamente - Tener claridad de qué áreas ameritan mayor exploración dentro de las estadísticas - Entender qué pasos seguir para navegar la página como desea | <ul style="list-style-type: none"> - Explorar más a fondo sus fortalezas y áreas de oportunidad en los temas - Ver los resultados concisos de su esfuerzo - Sentir un sentimiento de competencia sana |
| Acciones | <ul style="list-style-type: none"> - Da click estudiante - Pregunta cómo ingresar correctamente sus datos - Ingrresa sus datos de inicio | <ul style="list-style-type: none"> - Preguntar qué hacen las secciones de la página y dónde encontrar lo que busca - Compartir con sus compañerxs sus resultados - Dar click en los botones que le van a mostrar diferentes secciones | <ul style="list-style-type: none"> - Dar click en el tipo de filtrado que desea observar - Compartir con sus compañerxs sus resultados - Anotar qué es lo que quiere trabajar más - Comentar sobre el mini juego que disfrutó más jugar |
| Herramientas | <ul style="list-style-type: none"> - Computadora - Internet | <ul style="list-style-type: none"> - Computadora - Internet | <ul style="list-style-type: none"> - Computadora - Internet |
| Storyboard |  |  |  |
| Experiencia | <p>😊</p> <p>😐</p> <p>😢</p> |  | |
| Problemas | <ul style="list-style-type: none"> • Problemas con la autenticación del usuario • Algún atraso de respuesta de la página | <ul style="list-style-type: none"> • Algun botón que no responda correctamente • Que no se muestre la información correspondiente al alumnx • Que no resulte intuitivo para el alumnx navegar | <ul style="list-style-type: none"> • Que la información no se despliegue o tenga un atraso • Que los números y porcentajes resulten muy abrumadores para el alumnx • Poca iniciativa para checar resultados y buscar mayor información |
| Ganancias | <ul style="list-style-type: none"> - Acceso a la visualización de sus resultados | <ul style="list-style-type: none"> - Un panorama general de cómo fue su desempeño - Evidencia de sus logros que puede compartir - Conocimiento sobre el manejo adecuado de una página web sencilla | <ul style="list-style-type: none"> - Un recuento específico de su experiencia en el videojuego - Herramientas que ilustran sus temas de fortaleza y de área de oportunidad - Motivación para seguir aprendiendo de diversas maneras |

2.6.1 Videojuego

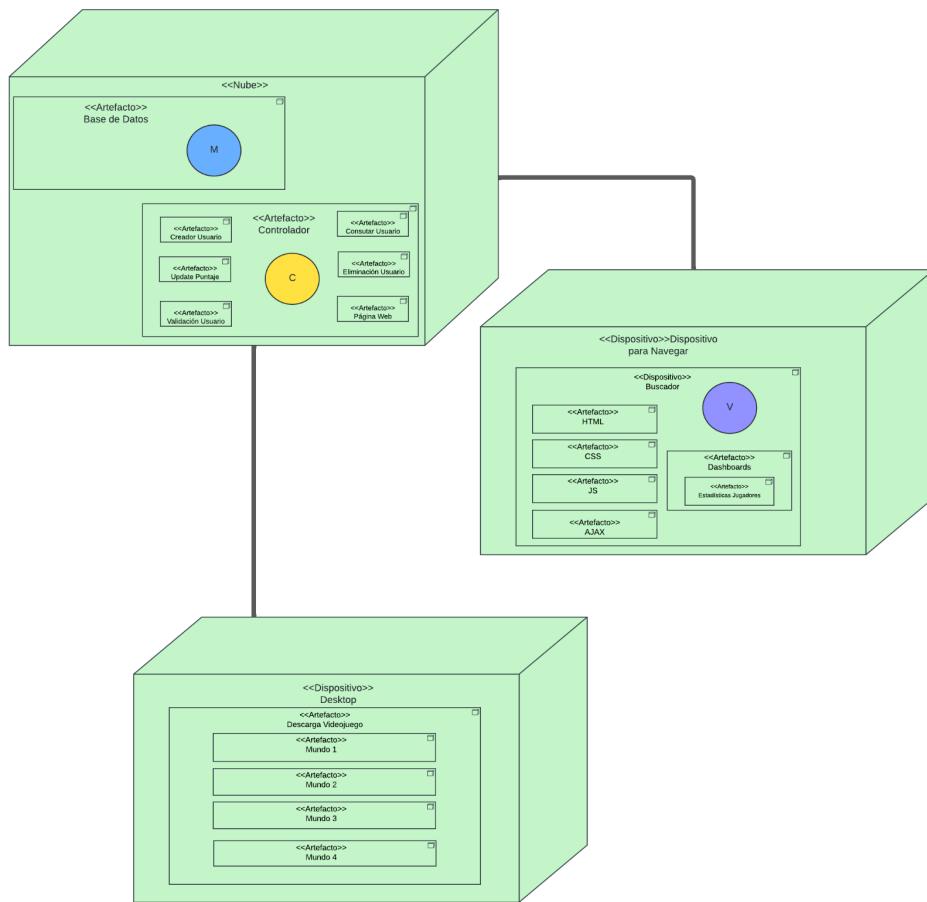
| Videojuego Experience Map | | | | | | | |
|---------------------------|--|--|---|---|---|--|---|
| Fases | Login | Preview de los mundos | Juega 1er mundo | Juega 2do mundo | Juega 3er mundo | Juega 4to mundo | Termina el juego |
| Necesidades | <ul style="list-style-type: none"> - Que el inicio de sesión sea corto, simple y fácil de entender - Que se vea atractivo el juego | <ul style="list-style-type: none"> - Que el mapa se vea atractivo - Que la navegación en el juego sea fácil e intuitiva | <ul style="list-style-type: none"> - Entender la dinámica del mini juego - Divertirse y aprender | <ul style="list-style-type: none"> - Entender la nueva dinámica - Divertirse - Mantenerse entretenida | <ul style="list-style-type: none"> - Entender la nueva dinámica - Manejar los controles con facilidad - Mantenerle el paso al juego | <ul style="list-style-type: none"> - Entender la nueva dinámica - Mantenerle el paso al juego - Aprender divirtiéndose | <ul style="list-style-type: none"> - Ver su avance - Sentir que el juego y ayudó a desarrollar sus habilidades - Ser reconocida por su esfuerzo |
| Acciones | <ul style="list-style-type: none"> - Ingresar su grupo - Ingresar su número de lista - Da click en comenzar | <ul style="list-style-type: none"> - Da click en el primer mundo | <ul style="list-style-type: none"> - Seleccionar las figuras que pide el juego - Subir de dificultad | <ul style="list-style-type: none"> - Mover los pedazos de aristas - Formar el contorno de figuras (cada vez más complejas) - Sumar la longitud de los pedazos de aristas | <ul style="list-style-type: none"> - Seleccionar el cubo con la fórmula correcta para el área del pie que debe atrapar - Seleccionar el cubo con el valor correcto del área que se solicita | <ul style="list-style-type: none"> - Seleccionar la fórmula o volumen correcto - Calcular el volumen de diferentes figuras geométricas (cada vez más complejas y a mayor velocidad) | <ul style="list-style-type: none"> - Ver su puntaje - Cerrar el juego |
| Herramientas | - Computadora | - Computadora | - Computadora | - Computadora - Papel y lápiz (tal vez) | - Computadora - Papel y lápiz (tal vez) | - Computadora - Papel y lápiz (tal vez) | - Computadora |
| Storyboard | | | | | | | |
| Experiencia | | | | | | | |
| Problemas | <ul style="list-style-type: none"> - Error al ingresar su grupo - Tardanza en el inicio de sesión por parte del juego | <ul style="list-style-type: none"> - Problemas para visualizar las gráficas - Dispositivo lento | <ul style="list-style-type: none"> - Que el juego se sienta muy simple al ser el primer nivel - Que al no sentirse desafiada se aburre - Que el juego se trabe | <ul style="list-style-type: none"> - Problemas para mover los pedazos de aristas - Dificultad para entender la jugabilidad | <ul style="list-style-type: none"> - Problemas para entender lo que se supone que hace - Tardarse en realizar el cálculo de ciertas áreas | <ul style="list-style-type: none"> - Tardarse en realizar los cálculos de volumen - Tiempo muy corto | <ul style="list-style-type: none"> - Que lo haya acabado muy rápido |
| Ganancias | - Puede empezar a jugar el videojuego | <ul style="list-style-type: none"> - Tiene una idea inicial de cómo es el juego - Puede empezar a interactuar con los mundos y mini-juegos | <ul style="list-style-type: none"> - Repasa su conocimiento sobre las diferentes figuras geométricas - Desbloquea un nuevo mundo | <ul style="list-style-type: none"> - Repasa sus habilidades de suma y resta - Repasa y aprendió sobre el perímetro de diferentes figuras - Desbloquea el siguiente mundo | <ul style="list-style-type: none"> - Repasa sus habilidades de multiplicación - Repasa las fórmulas para calcular el área de diferentes figuras - Desbloquea el siguiente mundo | <ul style="list-style-type: none"> - Repasa y desarrolló sus habilidades de multiplicación rápida - Repasa y aprendió las fórmulas de volumen de diferentes figuras - Completó todos los mundos | <ul style="list-style-type: none"> - Repasó y aprendió sobre conceptos de geometría - Se divirtió y convivió con sus compañeros - Terminó el juego |

| Videojuego Experience Map | | | | | | | |
|---|---|--|---|---|---|---|---|
| Ariely Yael Villatoro Amador A0166303 Nicole Kapelman Lepine A01664563 | | | | | | | |
| Stages | Login | Preview de los mundos | Juega 1er mundo | Juega 2do mundo | Juega 3er mundo | Juega 4to mundo | Termina el juego |
| Necesidades | <ul style="list-style-type: none"> - Tener acceso rápido a su cuenta - Recordar con facilidad sus datos para entrar - Encontrar fácilmente los botones y campos para llenar - Comprender desde dónde se quedó antes de salir - Entender cuál es el primer paso | <ul style="list-style-type: none"> - Entender rápidamente cuál es el camino general que van a seguir - Obtener una idea de cuál es la temática general del juego - Aprender a identificar formas mediante el juego | <ul style="list-style-type: none"> - Generar dudas sobre el tema para poder entender mejor - Señalar entendiendo y fuera de la dinámica usual de la clase - Aprender a identificar formas mediante el juego | <ul style="list-style-type: none"> - Entender con facilidad qué es el área de una figura - Sentirse más competente en temas de matemáticas - Sentirse motivado para seguir aprendiendo con el videojuego | <ul style="list-style-type: none"> - Aprender a sacar el área de manera directa - Sentir que cada vez va más al ritmo de sus compañeros - Sentirse motivado para reparar los temas por su cuenta | <ul style="list-style-type: none"> - Aprender las fórmulas de volumen sin que sea algo tedioso - Sentir que si es capaz de resolver problemas de matemáticas - Sentirse satisfecho con su progreso | <ul style="list-style-type: none"> - Que se le de reconocimiento sobre su esfuerzo y logros - Visualizar cómo fue mejorando - Visualizar fortalezas y áreas de oportunidad |
| Acciones | <ul style="list-style-type: none"> - Ingresar su grupo y número de lista - Da click en Ingresar - Espera confirmación de sus credenciales | <ul style="list-style-type: none"> - Navega y explora la visualización de los mundos - Hace click por partes para ver qué es lo que funciona y qué no - Se impacienta por querer entender rápido y empieza antes que sus compañeros | <ul style="list-style-type: none"> - Seleccionar cada figura conforme se pide - Repetir ayuda al docente para entender errores - Repite los niveles varias veces para obtener mejor puntuación - Adaptares a la dificultad de cada nivel | <ul style="list-style-type: none"> - Formar los perímetros de las figuras con los pedazos - Repite los niveles varios menos para obtener mejor puntuación - Poner más atención desde el inicio y tomar notas - Hacer las sumas en papel de los pedazos que hacen el perímetro | <ul style="list-style-type: none"> - Seleccionar correctamente el anzuelo - Repetir cada vez menos los niveles - Pedir ayuda a sus compañeros para entender el tema | <ul style="list-style-type: none"> - Seleccionar el cuadrante con la respuesta correcta, cada vez más rápido - Repetir los niveles, por gusto o porque le falta poco para pasar - Comparar con sus amigos su avance - Pedir ayuda a sus compañeros y a su maestro | <ul style="list-style-type: none"> - Buscar su puntaje más alto y su puntaje más alto - Compartir sus hallazgos con sus compañeros - Checar qué "medallas" ganó - Finalizar el juego |
| Herramientas | - Computadora - Datos de acceso | - Computadora | - Computadora | - Computadora - Papel y lápiz | - Computadora - Papel y lápiz | - Computadora - Papel y lápiz | - Computadora |
| Storyboard | | | | | | | |
| Experiencia | | | | | | | |
| Problemas | <ul style="list-style-type: none"> - No se registra correctamente los datos - No encuentra los botones para terminar el login - No esté descargado correctamente el juego | <ul style="list-style-type: none"> - Surge bugs o el movimiento de la navegación no sea tan fluido - Surge frustración por no llegar rápidamente a la parte divertida | <ul style="list-style-type: none"> - Frustración por rendimiento menor a lo esperado - Dificultad para entender los comandos - Errores para registrar el puntaje correcto y/o en la secuencia esperada del juego - Copia de respuestas - Malo reconocimiento de figuras por competitividad con compañeros - Copia de respuestas | <ul style="list-style-type: none"> - Que las líneas no se arrastren con facilidad - Que tiene que hay un patrón de que no puede aprender matemáticas de ninguna manera - Errores para registrar el puntaje correcto y/o en la secuencia esperada del juego - Aburrimiento | <ul style="list-style-type: none"> - Bugs en la dinámica del juego y en el registro de puntajes - Confusión entre las fórmulas de perímetro y área - Confusión para usar las operaciones matemáticas adecuadas - Se han quedado sobre que realmente no ha aprendido mucho, a pesar de avanzar | <ul style="list-style-type: none"> - Confusión entre las fórmulas de perímetro, área y volumen - No logra las respuestas en el tiempo esperado - Bugs en la dinámica del juego y en el registro de puntos - Respuestas al azar para terminar el juego | <ul style="list-style-type: none"> - Que no se haya asignado los puntajes correctos - Que no se muestren los premios correctos - Que secale mucho después que sus compañeros o con menor puntaje, y se sienta incompetente |
| Ganancias | <ul style="list-style-type: none"> - Ganó acceso al videojuego con su perfil - Ganó incentivo para explorar las otras etapas del videojuego | <ul style="list-style-type: none"> - Obtuvo una idea de lo que se va a tratar | <ul style="list-style-type: none"> - Conocimiento sobre sus áreas de oportunidad - Familiaridad con el funcionamiento del juego - Aprendimiento divertido con el tema de identificar diferentes figuras - Avanza al siguiente mundo | <ul style="list-style-type: none"> - Práctica para hacer sumas y/o multiplicaciones - Técnicas para sacar el perímetro - Costumbres para tomar apuntes y formular dudas - Avanza al siguiente mundo | <ul style="list-style-type: none"> - Gana conocimiento sobre qué le funciona más para estudiar - Práctica para hacer las operaciones para sacar el área | <ul style="list-style-type: none"> - Aprendizaje sobre la diferencia entre perímetro, área y volumen - Aprendizaje sobre cómo sacar el volumen de diversas figuras - Recomendación de que puede afrontar nuevos retos - Termina todos los mundos | <ul style="list-style-type: none"> - Una nueva experiencia de aprendizaje - Evidencia de que puede con problemas de matemáticas y de otra naturaleza - Aprendizaje de figuras, perímetro, área y volumen |

2.7 Diagrama de componentes



2.8 Diagrama de despliegue



2.9 Pruebas dinámicas

2.9.1 Videojuego

| ID Escenario de prueba | 1-RAF | | ID Caso de prueba | v1 | | |
|------------------------|--|-------------------|---|---|-----------|--|
| Descripción | Verificar el sistema de vidas del Mundo 1 (Figuras) | | Prioridad | Alta | | |
| Probado por | Ricardo Durán | | Revisado por | Arely Amador | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Seleccionar una respuesta correcta | Input (Cuadrado) | Se aumenta el puntaje y las vidas se mantienen | Se aumenta el puntaje y las vidas se mantienen | Pass | Checar la posición de los corazones |
| 2 | Seleccionar respuesta incorrecta | Input (Pentágono) | Se disminuye el puntaje y las vidas se disminuyen | Se disminuye el puntaje y las vidas se disminuyen | Pass | Se puede usar esta lógica para el resto de los mundos |
| 3 | Checar en el subnivel 2 que la funcionalidad sea la misma | Input (Cuadrado) | Se aumentan o disminuyen correctamente las vidas, dependiendo de la respuesta | Se aumentan o disminuyen correctamente las vidas, dependiendo de la respuesta | Pass | Es posible inferir que no se va a tener mayor problema en el siguiente nivel |
| 4 | Checar que la funcionalidad también sea correcta en el último subnivel | Input (Triángulo) | Se aumentan o disminuyen correctamente las vidas, dependiendo de la respuesta | Se aumentan o disminuyen correctamente las vidas, dependiendo de la respuesta | Pass | Nunca se presentó algún problema de lógica de vidas en el juego |

| ID Escenario de prueba | 1-RAF | ID Caso de prueba | V2 | | | |
|------------------------|--|-----------------------------------|--|--|-----------|--|
| Descripción | Verificar la aleatoriedad con la que salen las figuras en cada ronda | Prioridad | Alta | | | |
| Probado por | Ricardo Durán | Revisado por | Arely Amador | | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Jugar la primera vez en el primer subnivel | Se presionan el botón de comenzar | Se instancia alguna figura de las que se definieron en el script | Se instancia la figura, pero no en la posición correcta | Fail | Para la estética, hay que cambiar la posición de la figura |
| 1.1 | Jugar la primera vez en el primer subnivel, después de modificar el vector de posición | Se presionan el botón de comenzar | Se instancia la figura, en una posición más visible | Se instancia la figura, en una posición más visible | Pass | Todo en orden |
| 2 | Se pierde deliberadamente para jugar segunda ronda | Input (Cuadrado) | La figura que se instancia es diferente a la que en la primera ronda | La figura que se instancia es la misma que en la primera ronda | Fail | Se debe checar el random generator |
| 3 | Se pierde deliberadamente para jugar segunda ronda | Input (Triángulo) | La figura que se instancia es diferente a la que en la primera ronda | La figura que se instancia es diferente a la que en la primera ronda | Pass | Es una posibilidad que se pueda escoger entre más figuras |

| ID Escenario de prueba | 1-RAF | | ID Caso de prueba | | V3 | |
|------------------------|---|-----------------------------------|--|---|--------------|---|
| Descripción | Encontrar la velocidad más apta para el jugador | | Prioridad | | Media | |
| Probado por | Ricardo Durán | | Revisado por | | Arely Amador | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Jugar el nivel fácil con la velocidad predeterminada | Se presionan el botón de comenzar | Las figuras se mueven a una velocidad ligeramente retadora | Las figuras se mueven a una velocidad en donde no se alcanza a percibir las figuras | Fail | Puede ser por la computadora |
| 1.1 | Jugar el nivel fácil, con una velocidad ligeramente más lenta | Se presionan el botón de comenzar | Las figuras se mueven a una velocidad ligeramente retadora | Las figuras se mueven a una velocidad en donde no se alcanza a percibir las figuras | Fail | Se necesita reducir la velocidad por mucho más |
| 1.2 | Jugar el nivel fácil, con una velocidad mucho más lenta | Se presionan el botón de comenzar | Las figuras se mueven a una velocidad ligeramente retadora | Las figuras se mueven a una velocidad ligeramente retadora | Pass | Va a tener que ser diferente en cada computadora, pero es ajustable |

| | | | |
|------------------------|-------|-------------------|----|
| ID Escenario de prueba | 2-DRP | ID Caso de prueba | v1 |
|------------------------|-------|-------------------|----|

| Descripción | | Checar el flujo de diálogo del personaje | | Prioridad | | Media | |
|-------------|--|--|--|--|-----------|---|--|
| Probado por | | Dana Mendoza | | Revisado por | | Ricardo Durán | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios | |
| 1 | Verificar el tiempo para leer el texto | Velocidad de diálogo x | El diálogo se muestra para poderlo leer correctamente | El diálogo se muestra para poderlo leer correctamente | Pass | Se puede avanzar a la siguiente prueba | |
| 2 | Verificar la destrucción de la caja de diálogo | Se da click en comenzar el juego | Se destruye la caja de diálogo después de que se acaba de mostrar el texto | Se destruye la caja de diálogo después de que se acaba de mostrar el texto | Pass | Se muestra muy fluida esta lógica | |
| 3 | Verificar que se pueda saltar el diálogo | Se da click en el diálogo que se quiere uno saltar | El diálogo deja de aparecer, y pasa al siguiente diálogo | El diálogo deja de aparecer, y pasa al siguiente diálogo | Pass | Tiene una funcionalidad perfecta, que podría ser utilizada en los otros niveles | |

| ID Escenario de prueba | | 2-DRP | | ID Caso de prueba | | V2 |
|------------------------|--------|--|-------------------|-------------------|-----------|--------------|
| Descripción | | Asegurar la correcta animación del personaje | | Prioridad | | Alta |
| Probado por | | Ricardo Durán | | Revisado por | | Arely Amador |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |

| | | | | | | |
|---|--|---|--|--|------|---|
| 1 | Se deja correr el programa para ver el movimiento del personaje mientras habla | Se presionan el botón de comenzar | El personaje se mueve constantemente, sin la necesidad de input del usuario, mientras termina su diálogo | El personaje se mueve constantemente, sin la necesidad de input del usuario, mientras termina su diálogo | Pass | Se ve estéticamente muy bien |
| 2 | Se mueve el personaje para observar su movimiento alrededor del mundo | Se presionan las teclas para dirigir al personaje | El personaje se mueve a la dirección correcta, mientras corre su animación | El personaje se mueve a la dirección correcta, mientras corre su animación | Pass | Se ahorró mucho tiempo en estas pruebas |

| ID Escenario de prueba | 2-DRP | ID Caso de prueba | V3 | | | |
|------------------------|--|--|---|--|-----------|----------------------------------|
| Descripción | Corregir el registro de respuestas correctas | Prioridad | Alta | | | |
| Probado por | Dana Mendoza | Revisado por | Ricardo Durán | | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Se juega para ver de qué color se ponen las respuestas correctas e incorrectas | Se coloca un cuadrado en el espacio de un rectángulo | La respuesta incorrecta se pone en rojo | La respuesta incorrecta se pone en verde | Fail | Es un simple cambio en el código |

| | | | | | | |
|-----|---|--|--|--|------|--|
| 1.1 | Se juega para ver de qué color se ponen las respuestas correctas e incorrectas, después de haber hecho correcciones | Se coloca un cuadrado en el espacio de un rectángulo | Las figuras se mueven a una velocidad ligeramente retadora | Las figuras se mueven a una velocidad ligeramente retadora | Pass | Ya solo falta checar si cambia el sistema de vidas |
| 2 | Verificar que se pierde una vida cada vez que se pone roja una respuesta | Se coloca un cuadrado en el espacio de un rectángulo | Se pierde una vida con cada respuesta incorrecta | Se pierde una vida con cada respuesta incorrecta | Pass | Perfecto |

| ID Escenario de prueba | 3-NAA | ID Caso de prueba | v1 | | | |
|------------------------|---|------------------------------|---|---|-----------|---|
| Descripción | Corregir materiales rosas para la importación | Prioridad | Alta | | | |
| Probado por | Nicole Kapellmann | Revisado por | Arely Amador | | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Se importan los assets de peces, tal y cómo están | Assets de peces con material | Exportación idéntica a la que se ve en la computadora | Los peces se tornan rojas en otras computadoras | Fail | Existen muchísimas posibles razones por las que se da este caso |

| | | | | | | |
|-----|--|--|---|---|------|--|
| 1.1 | Se quita la herencia dentro de los assets de peces | Assets de peces con material | Exportación idéntica a la que se ve en la compu | Los peces se tornan rozas en otras computadoras | Fail | Es urgente verificar que siempre se vayan a importar correctamente, o que se pueda agregar casi inmediatamente |
| 1.2 | Se investiga en internet solución | Assets de peces con material, con shader diferente | Exportación idéntica a la que se ve en la compu | Exportación idéntica a la que se ve en la compu | Pass | Resultó tener una solución mucho más fácil de la esperada |

| ID Escenario de prueba | 3-NAA | | ID Caso de prueba | V2 | | |
|------------------------|---|--|---|---|-----------|--|
| Descripción | | Corregir el sistema de vidas | Prioridad | Alta | | |
| Probado por | | Nicole Kapellmann | Revisado por | Arely Amador | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Se hace un arreglo de corazones de vida | Se selecciona una respuesta incorrecta | Se deshace el corazón de hasta la izquierda | Se deshace el corazón de hasta la derecha, y dos a la vez | Pass | Podría ser por el acomodo del arreglo |
| 2 | Se cambia el orden del arreglo de corazones de vida | Se selecciona una respuesta incorrecta | Se deshace el corazón de hasta la izquierda | Se deshace el corazón de hasta la izquierda, y dos a la vez | Fail | Sí se están destruyendo los corazones correctos, pero con la lógica correcta |

| | | | | | | |
|---|--|--|---|---|------|---|
| 3 | Se cambia la condición a la que entra la destrucción del corazón | Se selecciona una respuesta incorrecta | Se deshace el corazón de hasta la izquierda | Se deshace el corazón de hasta la izquierda | Pass | Fue un buen trabajo en equipo, ver esta situación |
|---|--|--|---|---|------|---|

| ID Escenario de prueba | 3-NAA | | ID Caso de prueba | V3 | | |
|------------------------|---|---------------------------------------|--|--|-----------|---|
| Descripción | Probar la efectividad de la caña de pescar | | Prioridad | Alta | | |
| Probado por | Nicole Kapellmann | | Revisado por | Arely Amador | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Se jugó el nivel básico para verificar que se detecten las colisiones indicadas | Se hace click en la tecla para pescar | Un debug que diga que se detectó la colisión | Un debug que diga que se detectó la colisión | Pass | Es importante que sea con OnTriggerStay() y no con OnTriggerEnter() |
| 2 | Se cuentan cuántos segundos se tarda la caña en destruir al pez | Se hace click en la tecla para pescar | Se destruye el pez prácticamente inmediatamente después de la colisión | Se destruye el pez con un atraso después de la colisión | Fail | Simplemente puede ser por del delay |
| 2.1 | Mover el delay de la destrucción del pez, a un número menor | Se hace click en la tecla para pescar | Se destruye el pez prácticamente inmediatamente después de la colisión | Se destruye el pez prácticamente inmediatamente después de la colisión | Pass | Probablemente sea necesario cambiar ese delay en otras computadoras |

| ID Escenario de prueba | 4-ANV | | ID Caso de prueba | v1 | | |
|------------------------|---|------------------------------------|---|---|-----------|--|
| Descripción | Probar las animaciones del dinosaurio | | | Prioridad | | |
| Probado por | Arely Amador | | | Revisado por | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Se juega una partida en el nivel básico, para observar cómo avanza el dinosaurio | Se hace click en play | El dinosaurio muestra la animación de caminar, hasta que se encuentra con su primera figura | El dinosaurio muestra la animación de caminar, hasta que se encuentra con su primera figura | Pass | Siempre hay que checar que todas las animaciones estén seleccionadas |
| 2 | Se juega una partida en el nivel básico, para observar cómo el dinosaurio realiza la acción de comer | Se ingresa la respuesta correcta | EL dinosaurio realiza la acción de comer con la colisión de la figura | EL dinosaurio realiza la acción de comer con la colisión de la figura | Pass | Continúa funcionando todo bien |
| 3 | Se juega una partida en el nivel básico, para observar cómo el dinosaurio realiza la acción de caerse | Se ingresa la respuesta incorrecta | EL dinosaurio realiza la acción de caerse con la colisión de la figura | EL dinosaurio realiza la acción de caerse con la colisión de la figura | Pass | No hubo mayor problema con estas animaciones |

| ID Escenario de prueba | | 4-ANV | | ID Caso de prueba | | V2 | |
|------------------------|---|---|---|--|-----------|------------------------------------|--|
| Descripción | | Verificar la velocidad con la que el jugador debe ingresar la respuesta | | Prioridad | | Alta | |
| Probado por | | Arely Amador | | Revisado por | | Nicole Kapellmann | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios | |
| 1 | Se jugó el nivel difícil para ver qué tan retadora es la velocidad del input | Se ingresa la respuesta con velocidad x | Hay tiempo de respuesta muy retador, pero suficiente | Hay tiempo de respuesta muy retador, y casi no es suficiente | Fail | Hay que cambiar el contador | |
| 1.1 | Se jugó el nivel difícil para ver qué tan retadora es la velocidad del input, después de haber corregido la velocidad | Se ingresa la respuesta con velocidad x - y | Hay tiempo de respuesta muy retador, pero suficiente | Hay tiempo de respuesta muy retador, pero suficiente | Pass | Perfecto | |
| 2 | Se jugó el nivel difícil para ver qué tan intuitivo es ingresar la respuesta correcta | Se ingresa la respuesta con velocidad x - y | La tecla de enter funciona para ingresar la respuesta | La tecla de enter funciona para ingresar la respuesta | Pass | No es necesario hacer algún cambio | |

| ID Escenario de prueba | | 4-ANV | | ID Caso de prueba | | V3 | |
|------------------------|---|---|--|--|-----------|-----------------------------------|--|
| Descripción | | Asegurarse de que la importación a 3D esté bien hecha | | Prioridad | | Alta | |
| Probado por | | Arely Amador | | Revisado por | | Nicole Kapellmann | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios | |
| 1 | Se exportó el juego hacia el resto de los juegos 3D | La exportación del juego | Un funcionamiento idéntico, moviendo el eje en z | Un funcionamiento idéntico, moviendo el eje en z | Pass | Hay que descargar el Package "2D" | |

2.9.2 Web

| ID Escenario de prueba | | 5-RDB | | ID Caso de prueba | | V3 | |
|------------------------|--------|--|-------------------|-------------------|-----------|--------------|--|
| Descripción | | Asegurarse de que se puedan crear nuevos usuarios a la base de datos | | Prioridad | | Alta | |
| Probado por | | Ricardo Durán | | Revisado por | | Dana Mendoza | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios | |

| | | | | | | |
|---|--|-----------------------|--|--|------|---------------------------|
| 1 | Se ingresa un nuevo usuario, desde la pestaña de agregar | Un usuario nuevo | El usuario se refleja en la base de datos y en la página web | El usuario se refleja en la base de datos y en la página web | Pass | Las conexiones están bien |
| 2 | Se hace la distinción entre maestro y alumno | Se ingresa un maestro | Se agrega un nuevo maestro en la base de datos | Se agrega un nuevo maestro en la base de datos | Pass | Las conexiones están bien |

| ID Escenario de prueba | 5-RDB | ID Caso de prueba | V2 | | | |
|------------------------|---|-------------------|--|--|-----------|--|
| Descripción | Verificar el registro de un nuevo grupo | Prioridad | Alta | | | |
| Probado por | Ricardo Durán | Revisado por | Dana Mendoza | | | |
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Se agrega un Nuevo grupo | Nuevo grupo | En la parte del maestro, está disponible el asignar el alumno a otro grupo | En la parte del maestro, está disponible el asignar el alumno a otro grupo | Pass | Se tiene que dar refresh para que se vea visualmente |

| ID Escenario de prueba | 5-RDB | ID Caso de prueba | V3 |
|------------------------|--|-------------------|------|
| Descripción | Verificar la conexión entre el videojuego y las gráficas | Prioridad | Alta |

| Probado por | | Dana Mendoza | | Revisado por | | Ricardo Durán |
|-------------|--|---|--|--|-----------|---|
| No. | Acción | Entradas | Salidas esperadas | Salidas obtenidas | Resultado | Comentarios |
| 1 | Se modifica la puntuación de un juego | Se ingresan todas las respuestas correctas del Subnivel 1 del Mundo 1 | Se muestra un 100% de respuestas correctas en los dashboards | Se muestra un 100% de respuestas correctas en los dashboards | Pass | Se tiene la conexión correcta |
| 2 | Se juegan diferentes niveles en diferentes mundos | Se ingresan diferentes respuestas en cada nivel | Se refleja en los dashboards, las diferentes participaciones que ha tenido el alumno | Se refleja la participación, excepto en un mundo | Fail | Probablemente es una query |
| 2.1 | Se juegan diferentes niveles en diferentes mundos, después de ajustar la query | Se ingresan diferentes respuestas en cada nivel | Se refleja en los dashboards, las diferentes participaciones que ha tenido el alumno | dashboards, las diferentes participaciones que ha tenido el alumno | Pass | Están ya todas las conexiones correctas |

2.10 Pruebas estáticas

PRUEBA: PE001

Descripción de la prueba: Probando el código “FishControllers.cs.” Probando la funcionalidad de creación y posicionamiento de peces en el entorno de Unity.

Criterios de aceptación:

- El sistema debe ser capaz de instanciar un número específico de peces basado en la variable `numberOfFish`.

- Los peces instanciados deben ser posicionados aleatoriamente dentro de un radio especificado por spawnRadius.
- Los peces instanciados deben tener un componente Rigidbody desactivado y un componente FishMovement añadido.
- El sistema debe ser capaz de manejar correctamente el caso en el que el número de peces a instanciar exceda el número de prefabs disponibles.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script FishController debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script FishController esté adjunto a un objeto en la escena de Unity.
2. Configurar el número de peces a instanciar (numberOfFish) y el radio de spawn (spawnRadius) en el Inspector de Unity.
3. Ejecutar la escena de Unity y observar la creación de los peces.

Datos de Prueba:

- numberOfFish: 5
- spawnRadius: 20f
- zPosition: -3.338617f
- fishPrefabs: 3 prefabs de peces diferentes

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- Cada pez debe ser posicionado aleatoriamente dentro de un radio de 20 unidades del objeto al que está adjunto el script FishController.
- Cada pez instanciado debe tener un componente Rigidbody desactivado y un componente FishMovement añadido.
- Si el número de peces a instanciar excede el número de prefabs disponibles, el sistema debe manejar este caso sin errores, posiblemente instanciando los peces disponibles y repitiendo los prefabs si es necesario.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: Niki

Revisado por: Niki

PRUEBA: PE002

Descripción de la prueba: Probando el código “FishMovement.cs” Probando la funcionalidad de movimiento y cambio de dirección de los peces en el entorno de Unity.

Criterios de aceptación:

- El sistema debe permitir a los peces moverse en una dirección inicial.
- Al colisionar con un objeto que tenga un componente Plane, el pez debe cambiar de dirección.
- La velocidad de movimiento del pez debe ser ajustable mediante la variable swimSpeed.

Precondiciones: El script FishMovement debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script FishMovement esté adjunto a un objeto en la escena de Unity.
2. Configurar la velocidad de movimiento (swimSpeed) en el Inspector de Unity.
3. Ejecutar la escena de Unity y observar el movimiento del pez.
4. Colocar un objeto con un componente Plane en la escena de Unity y observar si el pez cambia de dirección al colisionar con él.

Datos de Prueba:

- swimSpeed: 3f
- counter: 0

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- El pez debe moverse en la dirección inicial configurada por swimSpeed.
- Al colisionar con un objeto que tenga un componente Plane, el pez debe cambiar de dirección inmediatamente.
- La velocidad de movimiento del pez debe ser ajustable mediante la variable swimSpeed.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE003

Descripción de la prueba: Probando el código “HookController.cs”. Verificar la funcionalidad de cambio de dirección de los peces al colisionar con objetos que tengan un componente Plane en el entorno de Unity.

Criterios de aceptación:

- Al colisionar con un objeto que tenga un componente Plane, el pez debe cambiar de dirección.
- La dirección de movimiento del pez debe cambiar inmediatamente al colisionar con un objeto Plane.
- La velocidad de movimiento del pez debe invertirse al cambiar de dirección.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script FishMovement debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script FishMovement esté adjunto a un objeto en la escena de Unity.
2. Configurar la velocidad de movimiento (swimSpeed) en el Inspector de Unity.
3. Ejecutar la escena de Unity y observar el movimiento del pez.
4. Colocar un objeto con un componente Plane en la escena de Unity y observar si el pez cambia de dirección al colisionar con él.

Datos de Prueba:

- swimSpeed: 3f
- counter: 0

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- Al colisionar con un objeto que tenga un componente Plane, el pez debe cambiar de dirección inmediatamente, reflejando un cambio en su velocidad de movimiento.
- La velocidad de movimiento del pez debe invertirse al cambiar de dirección, lo que significa que si estaba moviéndose hacia la izquierda, después de la colisión debe comenzar a moverse hacia la derecha, y viceversa.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE005

Descripción de la prueba: Verificar la funcionalidad de pausa, reanudación y reintentos del juego en el entorno de Unity. Criterios de aceptación:

Criterios de aceptación:

- Al llamar al método `_pause_game()`, el juego debe pausarse, el HUD de pausa debe activarse, y el sonido del juego debe silenciarse.
- Al llamar al método `_resume()`, el juego debe reanudarse, el HUD de pausa debe desactivarse, y el sonido del juego debe reactivarse.
- Al llamar al método `_retrygame()`, el juego debe reiniciarse, el HUD de pausa debe desactivarse, y el sonido del juego debe reactivarse.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script `_pause` debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script `_pause` esté adjunto a un objeto en la escena de Unity.
2. Configurar el objeto `_pausehud` en el Inspector de Unity para que esté activo o inactivo según sea necesario.
3. Ejecutar la escena de Unity y observar el comportamiento del juego.
4. Llamar al método `_pause_game()` para pausar el juego y verificar que el HUD de pausa se active y que el sonido del juego se silencie.
5. Llamar al método `_resume()` para reanudar el juego y verificar que el HUD de pausa se desactive y que el sonido del juego se reactive.
6. Llamar al método `_retrygame()` para reiniciar el juego y verificar que el HUD de pausa se desactive, que el sonido del juego se reactive, y que el juego se reinicie correctamente.

Datos de Prueba:

- `_pausehud`: Objeto activo/inactivo según sea necesario.
- `_gamecontroll`: Componente que controla el juego, incluyendo el sonido y la lógica de reanudación.

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- Al pausar el juego, el tiempo del juego debe detenerse, el HUD de pausa debe activarse, y el sonido del juego debe silenciarse.
- Al reanudar el juego, el tiempo del juego debe retomar, el HUD de pausa debe desactivarse, y el sonido del juego debe reactivarse.
- Al reiniciar el juego, el juego debe reiniciarse desde el principio, el HUD de pausa debe desactivarse, y el sonido del juego debe reactivarse.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE006

Descripción de la prueba: Verificar la funcionalidad de actualización de estilos y colores de gemas en el entorno de Unity. Criterios de aceptación:

Criterios de aceptación:

- Al llamar al método updatestyle(), los colores de las líneas y las gemas deben actualizarse según el estilo seleccionado.
- El método update_gems() debe iterar sobre todas las gemas y cambiar su color según si son buenas o malas, basándose en el estilo seleccionado.
- El método color_lines() debe actualizar el color de todas las líneas según el estilo seleccionado.
- El método changecolor() debe devolver un sprite de color correspondiente a si la gema es buena o mala, basándose en el estilo seleccionado.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script _stage_design debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script _stage_design esté adjunto a un objeto en la escena de Unity.
2. Configurar la lista _styles_stage con diferentes estilos de diseño en el Inspector de Unity.
3. Ejecutar la escena de Unity y observar el comportamiento del juego.
4. Llamar al método updatestyle() para actualizar los estilos de las líneas y las gemas.
5. Verificar que los colores de las líneas y las gemas se actualicen correctamente según el estilo seleccionado.
6. Verificar que el método changecolor() devuelva el sprite de color correcto para las gemas buenas y malas.

Datos de Prueba:

- _styles_stage: Lista de estilos de diseño con diferentes colores para líneas y gemas.
- _style_selected: Índice del estilo seleccionado en la lista _styles_stage.
- _lines: Lista de objetos de línea en la escena.
- _gems: Lista de objetos de gemas en la escena.

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- Al llamar al método updatestyle(), los colores de las líneas y las gemas deben actualizarse según el estilo seleccionado.
- Las gemas buenas y malas deben tener colores diferentes según el estilo seleccionado.

- El método changecolor() debe devolver el sprite de color correcto para las gemas buenas y malas.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE007

Descripción de la prueba: Probando el código “_stage_design.cs”. Verificar la funcionalidad de actualización de estilos y colores de gemas en el entorno de Unity, incluyendo la interacción con el componente _gamecontroll

Criterios de aceptación:

- Al llamar al método updatestyle(), los colores de las líneas y las gemas deben actualizarse según el estilo seleccionado.
- El método update_gems() debe iterar sobre todas las gemas y cambiar su color según si son buenas o malas, basándose en el estilo seleccionado.
- El método color_lines() debe actualizar el color de todas las líneas según el estilo seleccionado.
- El método changecolor() debe devolver un sprite de color correspondiente a si la gema es buena o mala, basándose en el estilo seleccionado.
- El script debe interactuar correctamente con el componente _gamecontroll para obtener la lista de gemas.

Precondiciones: El script _stage_design debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script _stage_design esté adjunto a un objeto en la escena de Unity.
2. Configurar la lista _styles_stage con diferentes estilos de diseño en el Inspector de Unity.
3. Asegurarse de que el componente _gamecontroll esté adjunto al mismo objeto o a un objeto accesible desde el script _stage_design.
4. Ejecutar la escena de Unity y observar el comportamiento del juego.
5. Llamar al método updatestyle() para actualizar los estilos de las líneas y las gemas.
6. Verificar que los colores de las líneas y las gemas se actualicen correctamente según el estilo seleccionado.
7. Verificar que el método changecolor() devuelva el sprite de color correcto para las gemas buenas y malas.

Datos de Prueba:

- `_styles_stage`: Lista de estilos de diseño con diferentes colores para líneas y gemas.
- `_style_selected`: Índice del estilo seleccionado en la lista `_styles_stage`.
- `_lines`: Lista de objetos de línea en la escena.
- `_gems`: Lista de objetos de gemas en la escena.
- `_gamecontroll`: Componente que controla el juego, incluyendo la gestión de gemas.

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- Al llamar al método `updatestyle()`, los colores de las líneas y las gemas deben actualizarse según el estilo seleccionado.
- Las gemas buenas y malas deben tener colores diferentes según el estilo seleccionado.
- El método `changecolor()` debe devolver el sprite de color correcto para las gemas buenas y malas.
- El script debe interactuar correctamente con el componente `_gamecontroll` para obtener la lista de gemas.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE008

Descripción de la prueba: Probando el código “custom_vars.cs”. Verificar la funcionalidad de la clase `_custom_design` y `_gem_line` en el entorno de Unity, incluyendo la serialización y la asignación de valores.

Criterios de aceptación:

- La clase `_custom_design` debe permitir la configuración de estilos de niveles personalizados, incluyendo colores de gemas buenas y malas, velocidad de gemas, color de línea, puntuación máxima del nivel, y color del HUD.
- La clase `_gem_line` debe permitir la creación de una línea de gemas, asignando un array de transformaciones a la variable `_gems`.
- La serialización de las clases debe permitir la configuración de sus propiedades en el Inspector de Unity.

Precondiciones: El sistema debe estar en funcionamiento y accesible. Las clases `_custom_design` y `_gem_line` deben estar definidas en el proyecto de Unity.

Pasos de la Prueba:

1. Asegurarse de que las clases `_custom_design` y `_gem_line` estén definidas en el proyecto de Unity.
2. Crear un objeto en la escena de Unity y adjuntar un script que utilice estas clases.
3. Configurar las propiedades de `_custom_design` y `_gem_line` en el Inspector de Unity.
4. Ejecutar la escena de Unity y observar el comportamiento del juego.
5. Verificar que las propiedades de `_custom_design` y `_gem_line` se hayan configurado correctamente según lo establecido en el Inspector de Unity.

Datos de Prueba:

- `_custom_design`: Configuración de estilos de niveles personalizados.
- `_gem_line`: Configuración de una línea de gemas.

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- La clase `_custom_design` debe permitir la configuración de estilos de niveles personalizados a través del Inspector de Unity.
- La clase `_gem_line` debe permitir la creación de una línea de gemas, asignando un array de transformaciones a la variable `_gems`.
- La serialización de las clases debe permitir la configuración de sus propiedades en el Inspector de Unity.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE009

Descripción de la prueba: Probando el código “gamecontrollers.cs”. Verificar la funcionalidad del controlador de juego `_gamecontrol` en el entorno de Unity, incluyendo la gestión de niveles, gemas, y la interacción del usuario.

Criterios de aceptación:

- El método `startgame()` debe iniciar el juego, estableciendo `_create_gems` en `true`.
- El método `loadlevel()` debe cargar el nivel actual, actualizando el estilo del nivel, cambiando el color del HUD, y preparando el efecto de carga de nivel si es necesario.
- El método `effectloadlevel()` debe detener el juego, mostrar el efecto de carga de nivel, y luego reanudar el juego.

- El método `_retrygame()` debe reiniciar el juego, restableciendo el puntaje y el nivel, y preparando el juego para comenzar de nuevo.
- El método `_reset_gem()` debe restablecer las gemas a su estado incorrecto, cambiando su etiqueta y sprite.
- El método `_create_good_gem()` debe crear una gema correcta en una línea específica, ajustando su posición y etiqueta.
- El método `Update()` debe manejar la lógica del juego, incluyendo el movimiento de las gemas y la interacción del usuario.
- El método `_change_hud_color()` debe cambiar el color del HUD según el nivel actual.
- El método `_resetallgems()` debe restablecer todas las gemas a su estado incorrecto.
- El método `_good_selection()` debe manejar la selección correcta de una gema, incrementando el puntaje y cambiando el sprite de la gema.
- El método `_bad_selection()` debe manejar la selección incorrecta de una gema, disminuyendo los intentos y activando el GameOverHUD si es necesario.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script `_gamecontroll` debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script `_gamecontroll` esté adjunto a un objeto en la escena de Unity.
2. Configurar las propiedades necesarias en el Inspector de Unity, como las gemas, las líneas de gemas, y los elementos del HUD.
3. Ejecutar la escena de Unity y observar el comportamiento del juego.
4. Llamar al método `startgame()` para iniciar el juego y verificar que las gemas comiencen a moverse.
5. Verificar que el método `loadlevel()` cargue correctamente el nivel actual y actualice el estilo del nivel.
6. Interactuar con el juego seleccionando gemas correctas e incorrectas y verificar que el puntaje y los intentos se manejen correctamente.
7. Llamar al método `_retrygame()` para reiniciar el juego y verificar que el juego se reinicie correctamente.

Datos de Prueba:

- `_levelgame`: Nivel actual del juego.
- `_gems`: Lista de transformaciones de gemas en el juego.
- `_g_lines`: Lista de líneas de gemas.
- `_attempts`: Lista de objetos de intentos en el HUD.
- `_ui_score`: Texto del puntaje en el HUD.
- `_color`: Imagen del color actual en el HUD.
- `_levelup`: Imagen del efecto de carga de nivel.
- `_levelup_txt`: Texto del efecto de carga de nivel.

- `_score_to_loadlevel`: Puntaje necesario para cargar el siguiente nivel.
- `_hud_colors`: Array de imágenes del HUD que cambian de color.
- `_GameOverHUD`: Componente del HUD de fin de juego.

Resultado Esperado: Se instancian 5 peces en la escena de Unity.

- El juego debe iniciar correctamente y manejar la lógica de niveles, gemas, y la interacción del usuario.
- Las gemas deben moverse y restablecerse correctamente.
- El puntaje y los intentos deben incrementarse y disminuirse correctamente.
- El HUD debe cambiar de color según el nivel actual.
- El juego debe reiniciarse correctamente al llamar al método `_retrygame()`.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE010

Descripción de la prueba: Probando el código “gameover.cs”. Verificar la funcionalidad del script gameover en el entorno de Unity, específicamente el método `_g_over()` que actualiza el HUD con el puntaje final del juego.

Criterios de aceptación:

- El método `_g_over()` debe encontrar el componente `_stage_design` adjunto al objeto `_Game` en la escena.
- El método `_g_over()` debe obtener el puntaje final del juego desde el componente `_gamecontrol` y actualizar el texto del primer elemento del HUD con este puntaje.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script `_gamecontrol` debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script gameover esté adjunto a un objeto en la escena de Unity.
2. Configurar las propiedades necesarias en el Inspector de Unity, como el objeto `_Game` y el componente `_stage_design`.
3. Ejecutar la escena de Unity y observar el comportamiento del juego.
4. Llamar al método `_g_over()` para actualizar el HUD con el puntaje final del juego.

5. Verificar que el texto del primer elemento del HUD se actualice correctamente con el puntaje final del juego.

Datos de Prueba:

- `_Game`: Objeto en la escena de Unity que tiene el componente `_stage_design` adjunto.
- `_stage_design`: Componente que maneja el diseño del juego, incluyendo el HUD.
- `_gamecontroll`: Componente que maneja la lógica del juego, incluyendo el puntaje.
-
- `_GameOverHUD`: Componente del HUD de fin de juego.

Resultado Esperado:

- El método `_g_over()` debe encontrar correctamente el componente `_stage_design` adjunto al objeto `_Game`.
- El método `_g_over()` debe obtener el puntaje final del juego desde el componente `_gamecontroll` y actualizar el texto del primer elemento del HUD con este puntaje

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE011

Descripción de la prueba: Probando el código “startgame.cs”. Verificar la funcionalidad del script startgame en el entorno de Unity, específicamente el método playgame() que inicia el juego y el método disablemenu() que desactiva el menú de inicio después de un retraso.

Criterios de aceptación:

- El método `playgame()` debe activar el componente Animator del objeto `_startmenu`.
- El método `playgame()` debe iniciar una corutina `disablemenu()` que desactiva el menú de inicio después de un retraso de 1 segundo.
- La corutina `disablemenu()` debe desactivar el objeto padre del `_startmenu` y llamar al método `startgame()` del componente `_gamecontroll` adjunto al objeto `_Game`.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script startgame debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script startgame esté adjunto a un objeto en la escena de Unity.
2. Configurar las propiedades necesarias en el Inspector de Unity, como el objeto _startmenu.
3. Ejecutar la escena de Unity y observar el comportamiento del juego.
4. Llamar al método playgame() para iniciar el juego y verificar que el menú de inicio se desactive correctamente después de un retraso de 1 segundo.
5. Verificar que el juego comience correctamente después de que el menú de inicio se desactive.

Datos de Prueba:

- _startmenu: Objeto en la escena de Unity que representa el menú de inicio.
- _gamecontroll: Componente que maneja la lógica del juego, incluyendo el inicio del juego.

Resultado Esperado:

- El método playgame() debe activar el componente Animator del objeto _startmenu.
- La corutina disablemenu() debe desactivar el objeto padre del _startmenu y llamar al método startgame() del componente _gamecontroll adjunto al objeto _Game.
- El juego debe comenzar correctamente después de que el menú de inicio se desactive.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE012

Descripción de la prueba: Probando el código "AccesoMundo.cs" Verificar la funcionalidad del script AccesoMundos en el entorno de Unity, específicamente la gestión de la navegación entre mundos y el bloqueo/desbloqueo de mundos basado en las preferencias del jugador.

Criterios de aceptación:

- El método bloquear() debe cambiar el texto de los botones de los mundos 2, 3 y 4 a "Bloqueado".

- Los métodos mandaM1(), mandaM2(), mandaM3() y mandaM4() deben cargar la escena correspondiente basada en las preferencias del jugador almacenadas en PlayerPrefs.
- Los métodos desbloqueaMundo2(), desbloqueaMundo3() y desbloqueaMundo4() deben cambiar el texto de los botones de los mundos 2, 3 y 4 a "Entra" si el jugador ha desbloqueado el mundo correspondiente.
- El método modificaBotones() debe llamar a los métodos de desbloqueo correspondientes basados en las preferencias del jugador almacenadas en PlayerPrefs.
- El método Start() debe bloquear inicialmente los mundos y luego modificar los botones según las preferencias del jugador.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script AccesoMundos debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Configurar las propiedades necesarias en el Inspector de Unity, como los botones de los mundos.
2. Ejecutar la escena de Unity y observar el comportamiento del juego.
3. Verificar que los botones de los mundos 2, 3 y 4 se bloqueen inicialmente.
4. Cambiar las preferencias del jugador en PlayerPrefs para desbloquear mundos y verificar que los botones correspondientes cambien a "Entra".
5. Llamar a los métodos mandaM1(), mandaM2(), mandaM3() y mandaM4() para navegar entre mundos y verificar que las escenas correspondientes se carguen correctamente.

Datos de Prueba:

- PlayerPrefs: Almacenamiento de preferencias del jugador, incluyendo los mundos desbloqueados.
- SceneManager: Manejador de escenas de Unity.
- TextMeshProUGUI: Componente de texto utilizado para mostrar el estado de los mundos.

Resultado Esperado:

- Los mundos deben bloquearse inicialmente y desbloquearse según las preferencias del jugador almacenadas en PlayerPrefs.
- Los botones de los mundos deben cambiar a "Entra" si el jugador ha desbloqueado el mundo correspondiente.
- Al navegar entre mundos, las escenas correspondientes deben cargarse correctamente.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE013

Descripción de la prueba: Probando el código “BotonMulti.cs” Verificar la funcionalidad del script BotonMulti en el entorno de Unity, específicamente los métodos AvanzaNivelBoton() y RetryNivelBoton() que manejan la navegación entre niveles y la reintentación de niveles.

Criterios de aceptación:

- El método AvanzaNivelBoton() debe cargar la siguiente escena basada en la escena anterior almacenada en PlayerPrefs.
- El método RetryNivelBoton() debe cargar la escena anterior almacenada en PlayerPrefs, permitiendo al jugador reintentar el nivel.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script BotonMulti debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script BotonMulti esté adjunto a un objeto en la escena de Unity.
2. Configurar las propiedades necesarias en el Inspector de Unity, como los botones de navegación.
3. Ejecutar la escena de Unity y observar el comportamiento del juego.
4. Llamar al método AvanzaNivelBoton() para avanzar al siguiente nivel y verificar que la escena correspondiente se cargue correctamente.
5. Llamar al método RetryNivelBoton() para reintentar el nivel actual y verificar que la escena anterior se cargue correctamente.

Datos de Prueba:

- PlayerPrefs: Almacenamiento de preferencias del jugador, incluyendo la escena anterior.
- SceneManager: Manejador de escenas de Unity.

Resultado Esperado:

- Al llamar al método AvanzaNivelBoton(), la escena siguiente correspondiente a la escena anterior almacenada en PlayerPrefs debe cargarse correctamente.
- Al llamar al método RetryNivelBoton(), la escena anterior almacenada en PlayerPrefs debe cargarse correctamente, permitiendo al jugador reintentar el nivel.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE014

Descripción de la prueba: Probando el código “Camara.cs” Verificar la funcionalidad del script Camara en el entorno de Unity, específicamente la gestión de la cámara para moverse en el espacio 3D basado en las entradas del teclado.

Criterios de aceptación:

- La cámara debe moverse hacia adelante cuando se presiona la tecla 'W'.
- La cámara debe moverse hacia atrás cuando se presiona la tecla 'S'.
- La cámara debe moverse hacia la izquierda cuando se presiona la tecla 'A'.
- La cámara debe moverse hacia la derecha cuando se presiona la tecla 'D'.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script Camara debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script Camara esté adjunto a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar el comportamiento de la cámara.
3. Presionar las teclas 'W', 'S', 'A', y 'D' para mover la cámara en diferentes direcciones.
4. Verificar que la cámara se mueva correctamente en cada dirección según las teclas presionadas.

Datos de Prueba:

- Input.GetKeyDown(): Método de Unity para detectar la presión de teclas.
- transform.position: Propiedad de Unity para obtener o establecer la posición de un objeto en el espacio 3D.

Resultado Esperado:

- La cámara debe moverse hacia adelante, hacia atrás, hacia la izquierda y hacia la derecha según las teclas 'W', 'S', 'A', y 'D' presionadas, respectivamente.

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

Fecha de Ejecución: 19/04/2024

Asignado a: -

Revisado por: .

PRUEBA: PE015

Descripción de la prueba: Probando el código “Cubo.cs” Verificar la funcionalidad del script Cubo en el entorno de Unity, específicamente la gestión de las propiedades de un cubo, incluyendo su base, altura, profundidad, y volumen, así como la inicialización del componente Rigidbody2D.

Criterios de aceptación:

- Las propiedades baseC, altura, profundidad, y volumen deben ser accesibles y modificables a través de sus propiedades públicas gsBase, gsAltura, gsProfundidad, y gsVolumen.
- El componente Rigidbody2D debe ser inicializado correctamente en el método Awake().

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script Cubo debe estar adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.

Pasos de la Prueba:

1. Asegurarse de que el script Cubo esté adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.
2. Ejecutar la escena de Unity y observar el comportamiento del objeto.
3. Modificar las propiedades gsBase, gsAltura, gsProfundidad, y gsVolumen para asegurarse de que los cambios se reflejen correctamente en las propiedades correspondientes.
4. Verificar que el componente Rigidbody2D se inicialice correctamente en el método Awake().

Datos de Prueba:

- Rigidbody2D: Componente de Unity para la física 2D.

- GetComponent<Rigidbody2D>(): Método de Unity para obtener el componente Rigidbody2D adjunto al objeto.

Resultado Esperado:

- Las propiedades baseC, altura, profundidad, y volumen deben ser accesibles y modificables a través de sus propiedades públicas gsBase, gsAltura, gsProfundidad, y gsVolumen.
- El componente Rigidbody2D debe ser inicializado correctamente en el método Awake().

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

PRUEBA: PE016

Descripción de la prueba: Probando el código “CuboFacil.cs” Verificar la funcionalidad del script CuboFacil en el entorno de Unity, específicamente la gestión de las fórmulas para calcular el volumen de un cubo en diferentes niveles de dificultad y la inicialización del componente Rigidbody2D

Criterios de aceptación:

- Las fórmulas para calcular el volumen de un cubo en diferentes niveles de dificultad (formulaFacil, formulaMedia, formulaDificil) deben ser accesibles a través de sus propiedades públicas (gFormulaFacil, gFormulaMedia, gFormulaDificil).
- La propiedad pr debe ser accesible y modificable a través de su propiedad pública gsPR.
- El componente Rigidbody2D debe ser inicializado correctamente en el método Awake().

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script CuboFacil debe estar adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.

Pasos de la Prueba:

1. Asegurarse de que el script CuboFacil esté adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.
2. Ejecutar la escena de Unity y observar el comportamiento del objeto.
3. Acceder a las fórmulas de cálculo del volumen de un cubo a través de sus propiedades públicas para verificar que las fórmulas sean correctas.
4. Modificar la propiedad gsPR para asegurarse de que los cambios se reflejen correctamente en la propiedad pr.
5. Verificar que el componente Rigidbody2D se inicialice correctamente en el método Awake().

Datos de Prueba:

- Rigidbody2D: Componente de Unity para la física 2D.
- GetComponent<Rigidbody2D>(): Método de Unity para obtener el componente Rigidbody2D adjunto al objeto.

Resultado Esperado:

- Las fórmulas para calcular el volumen de un cubo en diferentes niveles de dificultad deben ser accesibles a través de sus propiedades públicas.
- La propiedad pr debe ser accesible y modificable a través de su propiedad pública gsPR.
- El componente Rigidbody2D debe ser inicializado correctamente en el método Awake().

Resultado Actual: (A completar después de la ejecución de la prueba)

Observaciones y Comentarios: (A completar después de la ejecución de la prueba)

Estado de la Prueba: Completado

PRUEBA: PE016

Descripción de la prueba: Probando el código “CuboFacil.cs” Verificar la funcionalidad del script CuboFacil en el entorno de Unity, específicamente la gestión de las fórmulas para calcular el volumen de un cubo en diferentes niveles de dificultad y la inicialización del componente Rigidbody2D

Criterios de aceptación:

- Las fórmulas para calcular el volumen de un cubo en diferentes niveles de dificultad (formulaFacil, formulaMedia, formulaDifícil) deben ser accesibles a través de sus propiedades públicas (gFormulaFacil, gFormulaMedia, gFormulaDifícil).
- La propiedad pr debe ser accesible y modificable a través de su propiedad pública gsPR.
- El componente Rigidbody2D debe ser inicializado correctamente en el método Awake().

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script CuboFacil debe estar adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.

Pasos de la Prueba:

1. Asegurarse de que el script CuboFacil esté adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.
2. Ejecutar la escena de Unity y observar el comportamiento del objeto.

3. Acceder a las fórmulas de cálculo del volumen de un cubo a través de sus propiedades públicas para verificar que las fórmulas sean correctas.
4. Modificar la propiedad gsPR para asegurarse de que los cambios se reflejen correctamente en la propiedad pr.
5. Verificar que el componente Rigidbody2D se inicialice correctamente en el método Awake().

Datos de Prueba:

- Rigidbody2D: Componente de Unity para la física 2D.
- GetComponent< Rigidbody2D >(): Método de Unity para obtener el componente Rigidbody2D adjunto al objeto.

Resultado Esperado:

- Las fórmulas para calcular el volumen de un cubo en diferentes niveles de dificultad deben ser accesibles a través de sus propiedades públicas.
- La propiedad pr debe ser accesible y modificable a través de su propiedad pública gsPR.
- El componente Rigidbody2D debe ser inicializado correctamente en el método Awake().

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE017

Descripción de la prueba: Probando el código "DesbloqueaMundo.cs" Verificar la funcionalidad del script DesbloqueaMundo en el entorno de Unity, específicamente el método desbloqueMundo() que desbloquea mundos basado en la escena anterior almacenada en PlayerPrefs.

Criterios de aceptación:

- El método desbloqueMundo() debe desbloquear el mundo "perimetro" si la escena anterior es "figurasL3".
- El método desbloqueMundo() debe desbloquear el mundo "area" si la escena anterior es "perimetroL3".
- El método desbloqueMundo() debe desbloquear el mundo "volumen" si la escena anterior es "areaL3".

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script DesbloqueaMundo debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script DesbloqueaMundo esté adjunto a un objeto en la escena de Unity.
2. Configurar las preferencias del jugador en PlayerPrefs para simular el progreso del jugador a través de las escenas.

3. Ejecutar la escena de Unity y llamar al método desbloqueMundo() para desbloquear los mundos correspondientes.
4. Verificar que los mundos "perimetro", "area", y "volumen" se desbloqueen correctamente según la escena anterior almacenada en PlayerPrefs.

Datos de Prueba:

- PlayerPrefs: Almacenamiento de preferencias del jugador, incluyendo la escena anterior.
- PlayerPrefs.GetString(): Método de Unity para obtener una cadena de texto de las preferencias del jugador.
- PlayerPrefs.SetString(): Método de Unity para establecer una cadena de texto en las preferencias del jugador.

Resultado Esperado:

- Los mundos "perimetro", "area", y "volumen" deben desbloquearse correctamente según la escena anterior almacenada en PlayerPrefs.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE018

Descripción de la prueba: Probando el código “DinoController.cs” Verificar la funcionalidad del script DinoController en el entorno de Unity, específicamente la gestión del movimiento del dinosaurio, la detección de objetos con raycasting, y la modificación de la velocidad basada en la presencia de objetos en la dirección de visión.

Criterios de aceptación:

- El dinosaurio debe moverse hacia adelante a una velocidad constante.
- El método raycasting() debe detectar objetos en la dirección de visión del dinosaurio y retornar el objeto detectado.
- La velocidad del dinosaurio debe ser modificada basada en la presencia de objetos en la dirección de visión.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script DinoController debe estar adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.

Pasos de la Prueba:

1. Asegurarse de que el script DinoController esté adjunto a un objeto en la escena de Unity que tenga un componente Rigidbody2D.
2. Ejecutar la escena de Unity y observar el comportamiento del dinosaurio.
3. Verificar que el dinosaurio se mueva hacia adelante a una velocidad constante.
4. Colocar objetos en la dirección de visión del dinosaurio y verificar que el método raycasting() detecte correctamente estos objetos.

5. Verificar que la velocidad del dinosaurio se modifique basada en la presencia de objetos en la dirección de visión.

Datos de Prueba:

- Rigidbody2D: Componente de Unity para la física 2D.
- Physics2D.Raycast(): Método de Unity para realizar un raycast en la dirección especificada.
- Time.deltaTime: Variable de Unity que representa el tiempo transcurrido desde el último frame.

Resultado Esperado:

- El dinosaurio debe moverse hacia adelante a una velocidad constante.
- El método raycasting() debe detectar correctamente objetos en la dirección de visión del dinosaurio.
- La velocidad del dinosaurio debe ser modificada basada en la presencia de objetos en la dirección de visión.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE019

Descripción de la prueba: Probando el código “EndPoints.cs” Verificar la funcionalidad del script EndPoints en el entorno de Unity, específicamente la comunicación con una API externa para actualizar valores como puntuaciones, intentos, niveles desbloqueados y mundos desbloqueados.

Criterios de aceptación:

- El método APICon() debe realizar una solicitud POST a la API externa y manejar correctamente la respuesta.
- Los métodos updateScoreValue(), updateTries(), updateUnlockedLevels(), y updateUnlockedWorlds() deben llamar al método APICon() con el endpoint correspondiente para actualizar los valores en la API.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script EndPoints debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script EndPoints esté adjunto a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y llamar a los métodos updateScoreValue(), updateTries(), updateUnlockedLevels(), y updateUnlockedWorlds() para simular la actualización de valores en la API.
3. Verificar que las solicitudes POST se realicen correctamente a la API externa y que las respuestas se manejen adecuadamente.

4. Observar los logs de Unity para verificar que no haya errores en las solicitudes y que las respuestas sean las esperadas.

Datos de Prueba:

- `UnityWebRequest.Post()`: Método de Unity para realizar una solicitud POST a una URL.
- `WWWForm`: Clase de Unity para crear formularios web que se pueden enviar con solicitudes web.
- `StartCoroutine()`: Método de Unity para iniciar una corutina, que es una función que puede pausar su ejecución y retomarla en el siguiente frame.

Resultado Esperado:

- Las solicitudes POST a la API externa deben realizarse correctamente y las respuestas deben manejarse adecuadamente.
- Los métodos `updateScoreValue()`, `updateTries()`, `updateUnlockedLevels()`, y `updateUnlockedWorlds()` deben llamar al método `APICon()` con el endpoint correspondiente para actualizar los valores en la API.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE020

Descripción de la prueba: Probando el código “Figures.cs” Verificar la funcionalidad del script Figures en el entorno de Unity, específicamente la creación y posicionamiento de figuras geométricas (cubos, prismas rectangulares y triangulares) en la escena basado en la dificultad seleccionada.

Criterios de aceptación:

- El script debe crear y posicionar correctamente figuras geométricas en la escena basado en la dificultad seleccionada (medio o difícil).
- Las figuras creadas deben tener sus propiedades (altura, base, profundidad, área, volumen) correctamente establecidas y mostradas en la interfaz de usuario.
- La posición de las figuras en la escena debe ser ajustada de manera que no se superpongan y se mantengan dentro de los límites de la escena.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script Figures debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script Figures esté adjunto a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar la creación de figuras geométricas en la escena.
3. Verificar que las figuras creadas tengan sus propiedades correctamente establecidas y mostradas en la interfaz de usuario.
4. Observar la posición de las figuras en la escena para asegurarse de que no se superpongan y se mantengan dentro de los límites de la escena.

Datos de Prueba:

- Instantiate(): Método de Unity para crear una instancia de un objeto.
- Random.Range(): Método de Unity para generar un número aleatorio dentro de un rango especificado.
- Canvas: Componente de Unity para mostrar texto y otros elementos de UI.
- TextMeshProUGUI: Componente de Unity para mostrar texto en la UI.

Resultado Esperado:

- Las figuras geométricas deben crearse y posicionarse correctamente en la escena basado en la dificultad seleccionada.
- Las figuras creadas deben tener sus propiedades correctamente establecidas y mostradas en la interfaz de usuario.
- La posición de las figuras en la escena debe ser ajustada de manera que no se superpongan y se mantengan dentro de los límites de la escena.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE021

Descripción de la prueba: Probando el código “FiguresFacil.cs” Verificar la funcionalidad del script FiguresFacil en el entorno de Unity, específicamente la creación y posicionamiento de figuras geométricas (cubos, prismas rectangulares y triangulares) en la escena con dificultad fácil, y la asignación de preguntas y respuestas a cada figura.

Criterios de aceptación:

- El script debe crear y posicionar correctamente figuras geométricas en la escena con dificultad fácil.
- Cada figura creada debe tener asignadas preguntas y respuestas correctas, y la respuesta correcta debe ser identificada.
- La posición de las figuras en la escena debe ser ajustada de manera que no se superpongan y se mantengan dentro de los límites de la escena.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script FiguresFacil debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script FiguresFacil esté adjunto a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar la creación de figuras geométricas en la escena con dificultad fácil.
3. Verificar que cada figura creada tenga asignadas preguntas y respuestas correctas, y que la respuesta correcta sea identificada.
4. Observar la posición de las figuras en la escena para asegurarse de que no se superpongan y se mantengan dentro de los límites de la escena.

Datos de Prueba:

- Instantiate(): Método de Unity para crear una instancia de un objeto.
- Random.Range(): Método de Unity para generar un número aleatorio dentro de un rango especificado.
- PreguntasRespuestas: Clase personalizada para almacenar preguntas y respuestas asociadas a cada figura.

Resultado Esperado:

- Las figuras geométricas deben crearse y posicionarse correctamente en la escena con dificultad fácil.
- Cada figura creada debe tener asignadas preguntas y respuestas correctas, y la respuesta correcta debe ser identificada.
- La posición de las figuras en la escena debe ser ajustada de manera que no se superpongan y se mantengan dentro de los límites de la escena.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE022

Descripción de la prueba: Probando el código “InputManager1.cs” Verificar la funcionalidad del script InputManager en el entorno de Unity, específicamente la gestión de la entrada del usuario a través de un campo de texto (TMP_InputField), la verificación de la respuesta del usuario contra la respuesta correcta, y la gestión de errores y el fin del juego.

Criterios de aceptación:

- El script debe permitir al usuario ingresar texto en el campo de texto vol.

- Al finalizar la edición del campo de texto, el script debe verificar si la entrada del usuario coincide con la respuesta correcta.
- El script debe llevar un conteo de errores y, al alcanzar tres errores, debe indicar el fin del juego y detener el movimiento del dinosaurio.

Precondiciones: El sistema debe estar en funcionamiento y accesible. El script InputManager debe estar adjunto a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que el script InputManager esté adjunto a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar la interacción con el campo de texto vol.
3. Ingresar texto en el campo de texto y verificar que el script procese correctamente la entrada y verifique si coincide con la respuesta correcta.
4. Observar el conteo de errores y asegurarse de que el juego indique el fin del juego y detenga el movimiento del dinosaurio al alcanzar tres errores.

Datos de Prueba:

- TMP_InputField: Componente de Unity para campos de texto.
- onEndEdit: Evento de Unity que se dispara cuando el usuario termina de editar el texto en un TMP_InputField.
- DinoController: Componente personalizado que controla el movimiento del dinosaurio.

Resultado Esperado:

- El script debe permitir al usuario ingresar texto en el campo de texto vol.
- Al finalizar la edición del campo de texto, el script debe verificar si la entrada del usuario coincide con la respuesta correcta.
- El script debe llevar un conteo de errores y, al alcanzar tres errores, debe indicar el fin del juego y detener el movimiento del dinosaurio.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE023

Descripción de la prueba: Probando el código “Jugador.cs” Verificar la funcionalidad de la clase Jugador en el entorno de desarrollo, específicamente la gestión de atributos como el grupo, el número de lista, los niveles desbloqueados, y las vidas del jugador.

Criterios de aceptación:

- La clase Jugador debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos del jugador, incluyendo el grupo, el número de lista, los niveles desbloqueados, y las vidas.
- Debe ser posible agregar y eliminar niveles desbloqueados de la lista de niveles desbloqueados.

Precondiciones: El entorno de desarrollo debe estar configurado y accesible. La clase Jugador debe estar definida y accesible.

Pasos de la Prueba:

1. Asegurarse de que la clase Jugador esté definida y accesible en el entorno de desarrollo.
2. Crear una instancia de la clase Jugador y verificar que sus atributos se inicialicen correctamente.
3. Modificar los atributos del jugador y verificar que los cambios se reflejen correctamente.
4. Agregar y eliminar niveles desbloqueados de la lista de niveles desbloqueados y verificar que la lista se actualice correctamente.

Datos de Prueba:

- int grupo: Atributo para almacenar el grupo del jugador.
- int numLista: Atributo para almacenar el número de lista del jugador.
- List<Nivel> nivelesDesbloqueados: Lista para almacenar los niveles desbloqueados por el jugador.
- int vidas: Atributo para almacenar el número de vidas del jugador.

Resultado Esperado:

- La clase Jugador debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos del jugador, incluyendo el grupo, el número de lista, los niveles desbloqueados, y las vidas.
- Debe ser posible agregar y eliminar niveles desbloqueados de la lista de niveles desbloqueados.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE024

Descripción de la prueba: Probando el código “Nivel.cs” Verificar la funcionalidad de la clase Nivel en el entorno de desarrollo, específicamente la gestión de atributos como el puntaje, el tema, y el subnivel del nivel.

Criterios de aceptación:

- La clase Nivel debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos del nivel, incluyendo el puntaje, el tema, y el subnivel.

Precondiciones: El entorno de desarrollo debe estar configurado y accesible. La clase Nivel debe estar definida y accesible.

Pasos de la Prueba:

1. Asegurarse de que la clase Nivel esté definida y accesible en el entorno de desarrollo.
2. Crear una instancia de la clase Nivel y verificar que sus atributos se inicialicen correctamente.
3. Modificar los atributos del nivel y verificar que los cambios se reflejen correctamente.

Datos de Prueba:

- int puntaje: Atributo para almacenar el puntaje del nivel.
- string tema: Atributo para almacenar el tema del nivel.
- int subNivel: Atributo para almacenar el subnivel del nivel.

Resultado Esperado:

- La clase Nivel debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos del nivel, incluyendo el puntaje, el tema, y el subnivel.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE025

Descripción de la prueba: Probando el código “PreguntasRespuestas.cs” Verificar la funcionalidad de la clase PreguntasRespuestas en el entorno de desarrollo, específicamente la gestión de atributos como la figura, las respuestas, y la respuesta correcta.

Criterios de aceptación:

- La clase PreguntasRespuestas debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos de la pregunta y respuestas, incluyendo la figura, las respuestas, y la respuesta correcta.

Precondiciones: El entorno de desarrollo debe estar configurado y accesible. La clase PreguntasRespuestas debe estar definida y accesible.

Pasos de la Prueba:

1. Asegurarse de que la clase PreguntasRespuestas esté definida y accesible en el entorno de desarrollo.
2. Crear una instancia de la clase PreguntasRespuestas y verificar que sus atributos se inicialicen correctamente.
3. Modificar los atributos de la pregunta y respuestas y verificar que los cambios se reflejen correctamente.

Datos de Prueba:

- string figura: Atributo para almacenar la figura asociada a la pregunta.
- string[] respuestas: Array para almacenar las respuestas posibles a la pregunta.
- int resCorrecta: Atributo para almacenar el índice de la respuesta correcta en el array de respuestas.

Resultado Esperado:

- La clase PreguntasRespuestas debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos de la pregunta y respuestas, incluyendo la figura, las respuestas, y la respuesta correcta.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE027

Descripción de la prueba: Probando el código “PrismaRectangular.cs” Verificar la funcionalidad de la clase PreguntasRespuestas en el entorno de desarrollo, específicamente la gestión de atributos como la figura, las respuestas, y la respuesta correcta.

Criterios de aceptación:

- La clase PrismaRectangular debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos del prisma rectangular, incluyendo la base, la altura, la profundidad, y el volumen.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

Precondiciones: El sistema debe estar en funcionamiento y accesible. La clase PrismaRectangular debe estar adjunta a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que la clase PrismaRectangular esté adjunta a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar la inicialización de los atributos del prisma rectangular.
3. Modificar los atributos del prisma rectangular y verificar que los cambios se reflejen correctamente.
4. Acceder al componente Rigidbody2D del objeto y realizar operaciones con él para verificar su correcta inicialización y acceso.

Datos de Prueba:

- int baseR: Atributo para almacenar la base del prisma rectangular.
- int altura: Atributo para almacenar la altura del prisma rectangular.
- int profundidad: Atributo para almacenar la profundidad del prisma rectangular.
- int volumen: Atributo para almacenar el volumen del prisma rectangular.
- Rigidbody2D rigidbody2d: Componente de Unity para la física del objeto.

Resultado Esperado:

- La clase PrismaRectangular debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos del prisma rectangular, incluyendo la base, la altura, la profundidad, y el volumen.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE027

Descripción de la prueba: Probando el código “PrismaRectangularFacil.cs” Verificar la funcionalidad de la clase PrismaRectangularFacil en el entorno de Unity, específicamente la gestión de atributos como la fórmula del prisma rectangular, la asignación de preguntas y respuestas, y la inicialización y acceso a su componente Rigidbody2D

Criterios de aceptación:

- La clase PrismaTriangular debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible modificar y acceder a los atributos del prisma triangular, incluyendo la altura, el área del triángulo, y el volumen.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

Precondiciones: El sistema debe estar en funcionamiento y accesible. La clase PrismaTriangular debe estar adjunta a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que la clase PrismaTriangular esté adjunta a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar la inicialización de los atributos del prisma triangular.
3. Modificar los atributos del prisma triangular y verificar que los cambios se reflejen correctamente.
4. Acceder al componente Rigidbody2D del objeto y realizar operaciones con él para verificar su correcta inicialización y acceso.

Datos de Prueba:

- int h: Atributo para almacenar la altura del prisma triangular.
- int aTriangulo: Atributo para almacenar el área del triángulo.
- float volumen: Atributo para almacenar el volumen del prisma triangular.
- Rigidbody2D rigidbody2d: Componente de Unity para la física del objeto.

Resultado Esperado:

- La clase PrismaTriangular debe inicializar correctamente sus atributos al ser instanciada.

- Debe ser posible modificar y acceder a los atributos del prisma triangular, incluyendo la altura, el área del triángulo, y el volumen.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE028

Descripción de la prueba: Probando el código “PrismaTriangular.cs” Verificar la funcionalidad de la clase PrismaTriangular en el entorno de Unity, específicamente la gestión de atributos como la altura del prisma triangular, el área del triángulo, y el volumen del prisma, así como la inicialización y acceso a su componente Rigidbody2D

Criterios de aceptación:

- La clase PrismaRectangularFacil debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible acceder a la fórmula del prisma rectangular y asignar preguntas y respuestas a través de la propiedad gsPR.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

Precondiciones: El sistema debe estar en funcionamiento y accesible. La clase PrismaRectangularFacil debe estar adjunta a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que la clase PrismaRectangularFacil esté adjunta a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar la inicialización de los atributos del prisma rectangular.
3. Acceder a la fórmula del prisma rectangular y verificar que se muestre correctamente.
4. Asignar preguntas y respuestas a través de la propiedad gsPR y verificar que se asignen correctamente.
5. Acceder al componente Rigidbody2D del objeto y realizar operaciones con él para verificar su correcta inicialización y acceso.

Datos de Prueba:

- string formula: Atributo para almacenar la fórmula del prisma rectangular.

- PreguntasRespuestas pr: Atributo para almacenar las preguntas y respuestas asociadas al prisma rectangular.
- Rigidbody2D rigidbody2d: Componente de Unity para la física del objeto.

Resultado Esperado:

- La clase PrismaRectangularFacil debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible acceder a la fórmula del prisma rectangular y asignar preguntas y respuestas a través de la propiedad gsPR.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

PRUEBA: PE029

Descripción de la prueba: Probando el código “PrismaTriangularFacil.cs” Verificar la funcionalidad de la clase PrismaTriangularFacil en el entorno de Unity, específicamente la gestión de atributos como la fórmula del prisma triangular, la asignación de preguntas y respuestas, y la inicialización y acceso a su componente Rigidbody2D

Criterios de aceptación:

- La clase PrismaTriangularFacil debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible acceder a la fórmula del prisma triangular y asignar preguntas y respuestas a través de la propiedad gsPR.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

Precondiciones: El sistema debe estar en funcionamiento y accesible. La clase PrismaTriangularFacil debe estar adjunta a un objeto en la escena de Unity.

Pasos de la Prueba:

1. Asegurarse de que la clase PrismaTriangularFacil esté adjunta a un objeto en la escena de Unity.
2. Ejecutar la escena de Unity y observar la inicialización de los atributos del prisma triangular.
3. Acceder a la fórmula del prisma triangular y verificar que se muestre correctamente.
4. Asignar preguntas y respuestas a través de la propiedad gsPR y verificar que se asigne correctamente.
5. Acceder al componente Rigidbody2D del objeto y realizar operaciones con él para verificar su correcta inicialización y acceso.

Datos de Prueba:

- string formula: Atributo para almacenar la fórmula del prisma triangular.
- PreguntasRespuestas pr: Atributo para almacenar las preguntas y respuestas asociadas al prisma triangular.
- Rigidbody2D rigidbody2d: Componente de Unity para la física del objeto.

Resultado Esperado:

- La clase PrismaTriangularFacil debe inicializar correctamente sus atributos al ser instanciada.
- Debe ser posible acceder a la fórmula del prisma triangular y asignar preguntas y respuestas a través de la propiedad gsPR.
- Debe ser posible acceder al componente Rigidbody2D del objeto y realizar operaciones con él.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE030

Descripción de la prueba: Probando el código “QuizManager.cs” Verificar la funcionalidad de la clase PrismaTriangularFacil en el entorno de Unity, específicamente la gestión de atributos como la fórmula del prisma triangular, la asignación de preguntas y respuestas, y la inicialización y acceso a su componente Rigidbody2D

Criterios de aceptación:

- El sistema debe asignar correctamente las respuestas a los botones de opción basándose en el componente del objeto actual (CuboFacil, PrismaRectangularFacil, PrismaTriangularFacil).
- El sistema debe marcar la respuesta correcta entre las opciones disponibles.
- El sistema debe actualizar las respuestas y la respuesta correcta cuando el objeto actual cambia.

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- El objeto "dino" debe tener un componente "DinoController" con un método "gFF" que devuelve el objeto actual.
- Los objetos actuales (CuboFacil, PrismaRectangularFacil, PrismaTriangularFacil) deben tener componentes con propiedades "gsPR" que contienen las respuestas y la respuesta correcta.

- Los botones "b1", "b2", "b3" deben tener componentes "Respuestas" y "TMP_Text" para mostrar las respuestas y marcarlas como correctas o incorrectas.

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto actual se establezca inicialmente.
2. Verificar que las respuestas y la respuesta correcta se asignen correctamente a los botones "b1", "b2", "b3".
3. Cambiar el objeto actual a través del método "avanza" y esperar a que se actualicen las respuestas y la respuesta correcta.
4. Verificar que las respuestas y la respuesta correcta se actualicen correctamente en los botones "b1", "b2", "b3".

Datos de Prueba:

- Objeto actual inicial: CuboFacil con respuestas ["Respuesta 1", "Respuesta 2", "Respuesta 3"] y respuesta correcta 2.
- Objeto actual después de cambiar: PrismaRectangularFacil con respuestas ["Respuesta A", "Respuesta B", "Respuesta C"] y respuesta correcta 1.

Resultado Esperado:

- Las respuestas y la respuesta correcta se asignen correctamente a los botones "b1", "b2", "b3" inicialmente.
- Después de cambiar el objeto actual, las respuestas y la respuesta correcta se actualicen correctamente en los botones "b1", "b2", "b3".

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE031

Descripción de la prueba: Probando el código “Respuestas.cs” Probando el comportamiento del sistema de gestión de respuestas en Unity, específicamente la lógica de respuesta correcta y la llamada al método "avanza" del QuizManager.

Criterios de aceptación:

- El sistema debe registrar correctamente si una respuesta es correcta o incorrecta.

- El sistema debe llamar al método "avanza" del QuizManager cuando se selecciona una respuesta, independientemente de si es correcta o incorrecta.

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- El componente "Respuestas" debe estar adjunto a un objeto que represente una respuesta en el quiz.
- El objeto debe tener una referencia al QuizManager a través de la propiedad "quizM".

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto "Respuestas" esté listo.
2. Seleccionar una respuesta marcada como correcta y verificar que el sistema registre la respuesta como correcta.
3. Seleccionar una respuesta marcada como incorrecta y verificar que el sistema registre la respuesta como incorrecta.
4. Verificar que, en ambos casos, el sistema llame al método "avanza" del QuizManager.

Datos de Prueba:

- Respuesta correcta: Con la propiedad "correcta" establecida en true.
- Respuesta incorrecta: Con la propiedad "correcta" establecida en false.

Resultado Esperado:

- Cuando se selecciona una respuesta correcta, el sistema debe registrar "Respuesta correcta" y llamar al método "avanza" del QuizManager.
- Cuando se selecciona una respuesta incorrecta, el sistema debe registrar "Respuesta incorrecta" y llamar al método "avanza" del QuizManager.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE032

Descripción de la prueba: Probando el código “boton_direccional.cs” Probando el comportamiento del sistema de navegación entre escenas en Unity, específicamente la lógica de transición entre diferentes escenas basadas en las acciones del usuario.

Criterios de aceptación:

- El sistema debe permitir al usuario navegar a la escena "previewMundos" al seleccionar la opción "Atrás".
- El sistema debe permitir al usuario navegar a la escena "fallaNivel" al seleccionar la opción "No Completa".
- El sistema debe permitir al usuario navegar a la escena "avanzaNivel" al seleccionar la opción "Completa".
- El sistema debe permitir al usuario navegar a la escena "avanzaMundo" al seleccionar la opción "Completa 3".
- El sistema debe permitir al usuario navegar a la escena "finJuego" al seleccionar la opción "Fin".

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- Las escenas "previewMundos", "fallaNivel", "avanzaNivel", "avanzaMundo", y "finJuego" deben estar cargadas y disponibles en el proyecto de Unity.
- El componente "boton_direccional" debe estar adjunto a un objeto que represente un conjunto de botones de navegación en el juego.

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto "boton_direccional" esté listo.
2. Seleccionar la opción "Atrás" y verificar que el sistema navegue a la escena "previewMundos".
3. Seleccionar la opción "No Completa" y verificar que el sistema navegue a la escena "fallaNivel".
4. Seleccionar la opción "Completa" y verificar que el sistema navegue a la escena "avanzaNivel".
5. Seleccionar la opción "Completa 3" y verificar que el sistema navegue a la escena "avanzaMundo".
6. Seleccionar la opción "Fin" y verificar que el sistema navegue a la escena "finJuego".

Datos de Prueba:

- Opción "Atrás": Llama al método "GoToAtras".
- Opción "No Completa": Llama al método "GoToNoCompleta".
- Opción "Completa": Llama al método "GoToCompleta".
- Opción "Completa 3": Llama al método "GoToCompleta3".
- Opción "Fin": Llama al método "GoToFin".

Resultado Esperado:

- Al seleccionar cada opción, el sistema debe navegar a la escena correspondiente sin errores.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE033

Descripción de la prueba: Probando el código “boton_login.cs” Probando el comportamiento del sistema de inicio de sesión en Unity, específicamente la lógica de envío de datos al servidor y la recepción de respuestas.

Criterios de aceptación:

- El sistema debe permitir al usuario ingresar su lista y grupo a través de campos de texto.
- El sistema debe enviar los datos ingresados al servidor utilizando una solicitud POST.
- El sistema debe recibir una respuesta del servidor y procesarla correctamente.
- El sistema debe manejar correctamente los errores de red o de servidor.

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- Los campos de texto "tmp_lista" y "tmp_grupo" deben estar disponibles y configurados para recibir entradas del usuario.
- El servidor en "<https://RicardoDuran.pythonanywhere.com/endp1>" debe estar en funcionamiento y aceptar solicitudes POST con los datos esperados.

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto "Login" esté listo.
2. Ingresar un valor válido en el campo "tmp_lista" y un valor válido en el campo "tmp_grupo".
3. Seleccionar la opción "HacerLogin" para enviar los datos al servidor.
4. Verificar que el sistema envíe correctamente los datos al servidor y reciba una respuesta.
5. Verificar que el sistema procese correctamente la respuesta del servidor.
6. Verificar que el sistema maneje correctamente los errores de red o de servidor.

Datos de Prueba:

- Valor válido para "tmp_lista": "12345".
- Valor válido para "tmp_grupo": "A".

Resultado Esperado:

- Al seleccionar "HacerLogin", el sistema debe enviar los datos ingresados al servidor y recibir una respuesta.
- La respuesta del servidor debe ser procesada correctamente y mostrada en el log del sistema.
- En caso de errores de red o de servidor, el sistema debe registrar el error en el log del sistema.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE034

Descripción de la prueba: Probando el comportamiento del sistema de atracción magnética en Unity, específicamente la lógica de aplicación de fuerza a un objeto basado en la posición de otro objeto.

Criterios de aceptación:

- El sistema debe permitir al usuario configurar un objeto "Magnet" y un factor de fuerza "forceFactor".
- El sistema debe aplicar una fuerza al objeto actual hacia el objeto "Magnet" basada en su posición y el factor de fuerza configurado.
- El sistema debe actualizar continuamente la fuerza aplicada para reflejar el movimiento del objeto "Magnet".

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- El objeto "magnet" debe tener un componente "Rigidbody" para aplicar fuerzas.
- El objeto "Magnet" debe estar configurado y disponible en el escenario.

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto "magnet" esté listo.
2. Configurar el objeto "Magnet" y el factor de fuerza "forceFactor" en el objeto "magnet".
3. Observar el comportamiento del objeto "magnet" en el escenario.
4. Verificar que el objeto "magnet" se mueva hacia el objeto "Magnet" aplicando una fuerza basada en su posición y el factor de fuerza configurado.
5. Verificar que la fuerza aplicada se actualice continuamente para reflejar el movimiento del objeto "Magnet".

Datos de Prueba:

- Objeto "Magnet" configurado en una posición específica en el escenario.
- Factor de fuerza "forceFactor" configurado a un valor específico.

Resultado Esperado:

- El objeto "magnet" debe moverse hacia el objeto "Magnet" aplicando una fuerza basada en su posición y el factor de fuerza configurado.
- La fuerza aplicada debe actualizarse continuamente para reflejar el movimiento del objeto "Magnet".

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE035

Descripción de la prueba: Probando el comportamiento del sistema de interacción con el perímetro de un círculo en Unity, específicamente la lógica de inicialización de dimensiones del círculo, cálculo y visualización del perímetro, y detección de interacción con el perímetro del círculo.

Criterios de aceptación:

- El sistema debe inicializar las dimensiones del círculo con un radio aleatorio entre 1 y 10.
- El sistema debe calcular y visualizar el perímetro del círculo basado en su radio.
- El sistema debe detectar la interacción con el perímetro del círculo y registrar un mensaje en el log del sistema.

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- El objeto "CirclePerimeterInteraction" debe tener componentes "TextMeshProUGUI" configurados para mostrar las dimensiones y el perímetro del círculo.
- El objeto "CirclePerimeterInteraction" debe tener un componente "Rigidbody" para realizar la detección de rayos.
- El objeto que representa el perímetro del círculo debe tener un tag "CirclePerimeter" y estar configurado para interactuar con el objeto "CirclePerimeterInteraction".

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto "CirclePerimeterInteraction" esté listo.
2. Verificar que las dimensiones del círculo se inicialicen correctamente y se muestren en el objeto "circleDimensionsText".
3. Verificar que el perímetro del círculo se calcule correctamente y se muestre en el objeto "circlePerimeterText".
4. Mover el objeto "CirclePerimeterInteraction" hacia el perímetro del círculo y verificar que se detecte la interacción y se registre un mensaje en el log del sistema.

Datos de Prueba:

- Radio del círculo inicial: Aleatorio entre 1 y 10.
- Interacción con el perímetro del círculo: A través de un raycast desde la posición del objeto "CirclePerimeterInteraction" hacia adelante, con una distancia de interacción configurada.

Resultado Esperado:

- Las dimensiones del círculo se inicialicen correctamente y se muestren en el objeto "circleDimensionsText".
- El perímetro del círculo se calcule correctamente y se muestre en el objeto "circlePerimeterText".
- Al mover el objeto "CirclePerimeterInteraction" hacia el perímetro del círculo, se detecte la interacción y se registre un mensaje en el log del sistema.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE036

Descripción de la prueba: Probando el comportamiento del sistema de interacción con el perímetro de un triángulo en Unity, específicamente la lógica de inicialización de dimensiones del triángulo, cálculo y visualización del perímetro, y detección de interacción con el perímetro del triángulo.

Criterios de aceptación:

- El sistema debe inicializar las dimensiones del triángulo con una base y una altura aleatorias entre 1 y 10.
- El sistema debe calcular y visualizar el perímetro del triángulo basado en su base y altura.
- El sistema debe detectar la interacción con el perímetro del triángulo y registrar un mensaje en el log del sistema.

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- El objeto "TrianglePerimeterInteraction" debe tener componentes "TextMeshProUGUI" configurados para mostrar las dimensiones y el perímetro del triángulo.
- El objeto "TrianglePerimeterInteraction" debe tener un componente "Rigidbody" para realizar la detección de rayos.
- El objeto que representa el perímetro del triángulo debe tener un tag "TrianglePerimeter" y estar configurado para interactuar con el objeto "TrianglePerimeterInteraction".

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto "TrianglePerimeterInteraction" esté listo.
2. Verificar que las dimensiones del triángulo se inicialicen correctamente y se muestren en el objeto "triangleDimensionsText".
3. Verificar que el perímetro del triángulo se calcule correctamente y se muestre en el objeto "trianglePerimeterText".
4. Mover el objeto "TrianglePerimeterInteraction" hacia el perímetro del triángulo y verificar que se detecte la interacción y se registre un mensaje en el log del sistema.

Datos de Prueba:

- Base y altura del triángulo iniciales: Aleatorios entre 1 y 10.

- Interacción con el perímetro del triángulo: A través de un raycast desde la posición del objeto "TrianglePerimeterInteraction" hacia adelante, con una distancia de interacción configurada.

Resultado Esperado:

- Las dimensiones del triángulo se inicialicen correctamente y se muestren en el objeto "triangleDimensionsText".
- El perímetro del triángulo se calcule correctamente y se muestre en el objeto "trianglePerimeterText".
- Al mover el objeto "TrianglePerimeterInteraction" hacia el perímetro del triángulo, se detecte la interacción y se registre un mensaje en el log del sistema.

Resultado Actual: Resultado esperado cumplido

Estado de la Prueba: Completado

PRUEBA: PE037

Descripción de la prueba: Probando el comportamiento del sistema de interacción con los perímetros de diferentes figuras geométricas en Unity, específicamente la lógica de detección de interacción con los perímetros de un triángulo, círculo y cuadrado.

Criterios de aceptación:

- El sistema debe realizar un raycast desde la posición de la figura hacia adelante.
- El sistema debe detectar si el objeto golpeado es un perímetro de una de las figuras geométricas especificadas (triángulo, círculo, cuadrado).
- El sistema debe registrar un mensaje en el log del sistema indicando con qué figura geométrica interactuó.

Precondiciones:

- El sistema debe estar en funcionamiento y accesible.
- El objeto "FigureInteraction" debe tener un componente "Rigidbody" para realizar la detección de rayos.
- Los objetos que representan los perímetros de las figuras geométricas (triángulo, círculo, cuadrado) deben tener tags correspondientes ("TrianglePerimeter", "CirclePerimeter", "SquarePerimeter") y estar configurados para interactuar con el objeto "FigureInteraction".

Pasos de la Prueba:

1. Iniciar el sistema y esperar a que el objeto "FigureInteraction" esté listo.
2. Mover el objeto "FigureInteraction" hacia los perímetros de las figuras geométricas.
3. Verificar que el sistema detecte la interacción con los perímetros de las figuras geométricas y registre un mensaje en el log del sistema indicando con qué figura geométrica interactuó.

Datos de Prueba:

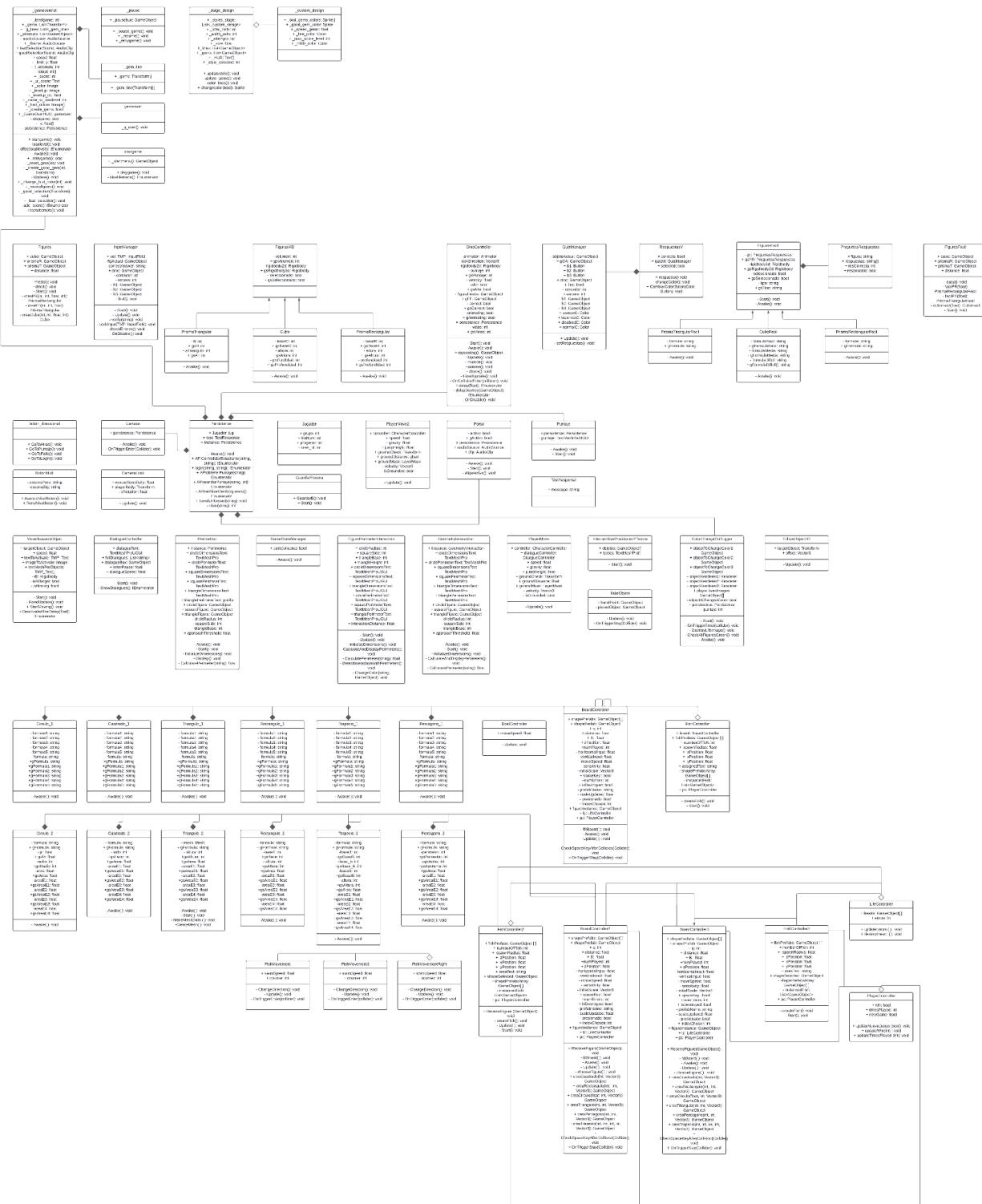
- Distancia de interacción: 2 unidades.
- Objetos de interacción: Perímetros de triángulo, círculo y cuadrado.

Resultado Esperado:

- Al mover el objeto "FigureInteraction" hacia los perímetros de las figuras geométricas, el sistema debe detectar la interacción y registrar un mensaje en el log del sistema indicando con qué figura geométrica interactuó.

Resultado Actual: Resultado esperado cumplido

2.11 Diagrama de clases



https://lucid.app/lucidchart/82c760b4-a9cd-42a6-9add-5746d96b7531/edit?viewport_loc=-13016%2C-4069%2C28263%2C11699%2C0_0&invitationId=inv_5bdf2a99-0b4d-4e9f-9612-52640e8c8ff0

2.12 Licencia

La licenciatante no puede revocar estas libertades en tanto usted siga los términos de la licencia.
Reconocimiento: Debes dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puedes hacerlo de cualquier manera razonable, pero no de ninguna manera que sugiera que tú o tu uso tienen el apoyo del licenciatante.

NoComercial: No puedes usar el material para ningún propósito comercial.

SinDerivadas: Si remixes, transformas o construyes sobre el material, no puedes distribuir el material modificado.

No hay restricciones adicionales: No puedes aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer nada que la licencia permita.

3. MODULO 3

3.1 APIs del videojuego

Endpoint: /estudiante/validar

Descripción

El endpoint `/estudiante/validar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es validar la información de un estudiante específico en el sistema, basándose en el `listNum` y el `grupo` proporcionados en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato de formulario. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Estudiante:** El endpoint extrae los campos `listNum` y `grupo` del cuerpo de la solicitud. Estos campos son necesarios para identificar y validar al estudiante en el sistema.
- **Verificación de Datos:** Antes de proceder con la validación, el endpoint verifica si alguno de los datos (`listNum` o `grupo`) es `None`. Si alguno de los campos está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- **Creación del Objeto Estudiante:** Con los datos obtenidos, se crea un objeto `Estudiante` que se utilizará para intentar validar al estudiante en el sistema. Aunque el código proporcionado no muestra explícitamente la conversión de `listNum` y `grupo` a enteros, es importante asegurarse de que estos valores sean del tipo correcto para la validación.
- **Validación del Estudiante:** Se invoca el método `validar` del controlador `EstudianteController`, pasando el objeto `Estudiante` creado. Este método realiza la

validación del estudiante, verificando si el `listNum` y el `grupo` proporcionados coinciden con los registros existentes en el sistema.

- **Respuesta en JSON:** Finalmente, el endpoint devuelve el resultado de la validación al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```
@app.route("/estudiante/validar", methods=["GET", "POST"]) # UNITY
def estudiante_validar():
    # Acceder a los datos enviados como formulario
    data = request.form
    if not data:
        return jsonify({"message": "Datos faltantes"})

    # Extraer los valores de 'listNum' y 'grupo' del formulario
    listNum = data.get("listNum")
    grupo = data.get("grupo")

    # Verificar si los datos requeridos están presentes
    if listNum is None or grupo is None:
        return jsonify({"message": "Datos faltantes"})

    # Convertir los valores de 'listNum' y 'grupo' a entero

    estudiante = Estudiante(listNum=listNum, grupo=grupo)
    return jsonify(EstudianteController().validar(estudiante))
```

Endpoint: /estudiante/progreso

Descripción

El endpoint `/estudiante/progreso` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es modificar el progreso de un estudiante específico en el sistema, basándose en el `estudiante_id` y el `progreso` proporcionados en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato de formulario. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Estudiante:** El endpoint extrae los campos `estudiante_id` y `progreso` del cuerpo de la solicitud. Estos campos son necesarios para identificar al estudiante específico cuyo progreso se desea modificar y para determinar el nuevo nivel de progreso.
- **Verificación de Datos:** Antes de proceder con la modificación del progreso, el endpoint verifica si alguno de los datos (`estudiante_id` o `progreso`) es `None`. Si alguno de los campos está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- **Creación del Objeto Estudiante:** Con los datos obtenidos, se crea un objeto `Estudiante` que se utilizará para intentar modificar el progreso del estudiante en el sistema.
- **Modificación del Progreso del Estudiante:** Se invoca el método `modificarProgreso` del controlador `EstudianteController`, pasando el objeto `Estudiante` creado. Este método realiza la modificación del progreso del estudiante, actualizando el registro del estudiante con el nuevo nivel de progreso proporcionado.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve el resultado de la modificación del progreso al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```

@app.route("/estudiante/progreso", methods=["GET", "POST"]) # UNITY
def estudiante_modificar_progreso():
    # Acceder a los datos enviados como formulario
    data = request.form
    if not data:
        return jsonify({"message": "Datos faltantes"})

    # Extraer los valores de 'listNum' y 'grupo' del formulario
    estudiante_id = data.get("estudiante_id")
    progreso = data.get("progreso")

    # Verificar si los datos requeridos están presentes
    if estudiante_id is None or progreso is None:
        return jsonify({"message": "Datos faltantes"})

    estudiante = Estudiante(estudiante_id=estudiante_id, progreso=progreso)
    return jsonify(EstudianteController().modificarProgreso(estudiante))

```

Endpoint: /estudiante/modificar

Descripción

El endpoint `/estudiante/modificar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es modificar la información de un estudiante específico en el sistema, permitiendo actualizar el `listNum`, el `grupo`, y posiblemente otros atributos del estudiante basándose en los datos proporcionados en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato JSON. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Estudiante:** El endpoint extrae los campos `estudiante_id`, `listNum`, y `grupo` del cuerpo de la solicitud. Estos campos son necesarios para identificar al estudiante específico cuya información se desea modificar y para determinar los nuevos valores para `listNum` y `grupo`.
- **Verificación de Datos:** Antes de proceder con la modificación de la información del estudiante, el endpoint verifica si alguno de los datos (`estudiante_id`, `listNum`, o `grupo`) es `None`. Si alguno de los campos está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- **Creación del Objeto Estudiante:** Con los datos obtenidos, se crea un objeto `Estudiante` que se utilizará para intentar modificar la información del estudiante en el sistema.
- **Modificación de la Información del Estudiante:** Se invoca el método `modificarEstudiante` del controlador `EstudianteController`, pasando el objeto `Estudiante` creado. Este método realiza la modificación de la información del estudiante, actualizando el registro del estudiante con los nuevos valores proporcionados.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve el resultado de la modificación de la información del estudiante al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```

@app.route("/estudiante/modificar", methods=["GET", "POST"]) # UNITY
def estudiante_modificar():
    # Acceder a los datos enviados como formulario
    data = request.json
    if not data:
        return jsonify({"message": "Datos faltantes"})

    # Extraer los valores de 'listNum' y 'grupo' del formulario
    estudiante_id = data.get("estudiante_id")
    listNum = data.get("listNum")
    grupo = data.get("grupo")

    # Verificar si los datos requeridos están presentes
    if estudiante_id is None or listNum is None or grupo is None:
        return jsonify({"message": "Datos faltantes"})

    estudiante = Estudiante(estudiante_id=estudiante_id, listNum=listNum, grupo=grupo)
    return jsonify(EstudianteController().modificarEstudiante(estudiante))

```

Endpoint: /score/registrar

Descripción

El endpoint `/score/registrar` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es registrar un nuevo puntaje para un estudiante específico en el sistema, permitiendo asociar un valor de puntaje a un nivel de aprendizaje determinado por el `estudiante_id` y el `nivel_id` proporcionados en la solicitud. Aquí se detallan sus funciones y uso:

- Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato de formulario. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- Extracción de Datos del Puntaje:** El endpoint extrae los campos `estudiante_id`, `nivel_id`, y `valor` del cuerpo de la solicitud. Estos campos son necesarios para identificar al estudiante específico al cual se le va a registrar el puntaje y para determinar el nivel de aprendizaje y el valor del puntaje.
- Verificación de Datos:** Antes de proceder con la registro del puntaje, el endpoint verifica si alguno de los datos (`estudiante_id`, `nivel_id`, o `valor`) es `None`. Si alguno de los campos está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- Creación del Objeto Puntaje:** Con los datos obtenidos, se crea un objeto `Puntaje` que se utilizará para intentar registrar el puntaje en el sistema.
- Registro del Puntaje:** Se invoca el método `registrarPuntaje` del controlador `PuntajeController`, pasando el objeto `Puntaje` creado. Este método realiza el registro del puntaje, almacenando el valor del puntaje asociado al estudiante y nivel especificados.

- **Respuesta en JSON:** Finalmente, el endpoint devuelve el resultado del registro del puntaje al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```
# SCORES///////////
@app.route("/score/registrar", methods=["GET", "POST"]) # UNITY
def registrar_score():
    # Acceder a los datos enviados como formulario
    data = request.form
    if not data:
        return jsonify({"message": "Datos faltantes"})

    # Extraer los valores de 'ListNum' y 'grupo' del formulario
    estudiante_id = data.get("estudiante_id")
    nivel_id = data.get("nivel_id")
    valor = data.get("valor")

    # Verificar si los datos requeridos están presentes
    if estudiante_id is None or nivel_id is None or valor is None:
        return jsonify({"message": "Datos faltantes"})

    puntaje = Puntaje(estudiante_id=estudiante_id, nivel_id=nivel_id, valor=valor)
    return jsonify(PuntajeController().registrarPuntaje(puntaje))
```

Endpoint: /puntaje/estudiante/nivel

Descripción

El endpoint `/puntaje/estudiante/nivel` en el código proporcionado es una ruta de Flask diseñada para manejar tanto solicitudes HTTP GET como POST. Su propósito principal es recuperar el último puntaje registrado para un estudiante específico en un nivel de aprendizaje determinado, basándose en los `estudiante_id` y `nivel_id` proporcionados en la solicitud. Aquí se detallan sus funciones y uso:

- **Recepción de Datos:** Al recibir una solicitud, el endpoint primero verifica si se ha enviado algún dato en formato de formulario. Si no se envían datos, se devuelve un mensaje de error indicando que faltan datos.
- **Extracción de Datos del Puntaje:** El endpoint extrae los campos `estudiante_id` y `nivel_id` del cuerpo de la solicitud. Estos campos son necesarios para identificar al estudiante específico y al nivel de aprendizaje para el cual se desea recuperar el último puntaje.

- **Verificación de Datos:** Antes de proceder con la recuperación del último puntaje, el endpoint verifica si alguno de los datos (`estudiante_id` o `nivel_id`) es `None`. Si alguno de los campos está vacío, se devuelve un mensaje de error en formato JSON indicando que faltan datos.
- **Recuperación del Último Puntaje:** Se invoca el método `ultimoPuntaje` del controlador `PuntajeController`, pasando los `estudiante_id` y `nivel_id` obtenidos. Este método realiza la búsqueda del último puntaje registrado para el estudiante en el nivel especificado.
- **Respuesta en JSON:** Finalmente, el endpoint devuelve el último puntaje registrado para el estudiante en el nivel especificado al cliente en formato JSON utilizando `jsonify()`. Esto asegura que la respuesta sea fácilmente interpretable por el cliente, ya que `jsonify()` convierte el objeto de Python en una cadena JSON y establece el tipo de contenido de la respuesta a `application/json`.

```
@app.route("/puntaje/estudiante/nivel", methods=["GET", "POST"]) # UNITY
def ultimo_puntaje_en_nivel_por_estudiante():
    # Imprimir todos los parámetros recibidos para depuración
    data = request.form
    if data is None:
        return jsonify({"message": "Datos faltantes"})
    # Extraer los valores del JSON
    estudiante_id = data.get("estudiante_id")
    nivel_id = data.get("nivel_id")

    if estudiante_id is None or nivel_id is None:
        return jsonify({"message": "Datos faltantes"})

    return jsonify(PuntajeController().ultimoPuntaje(estudiante_id, nivel_id))
```

3.2 Sistema de reportes

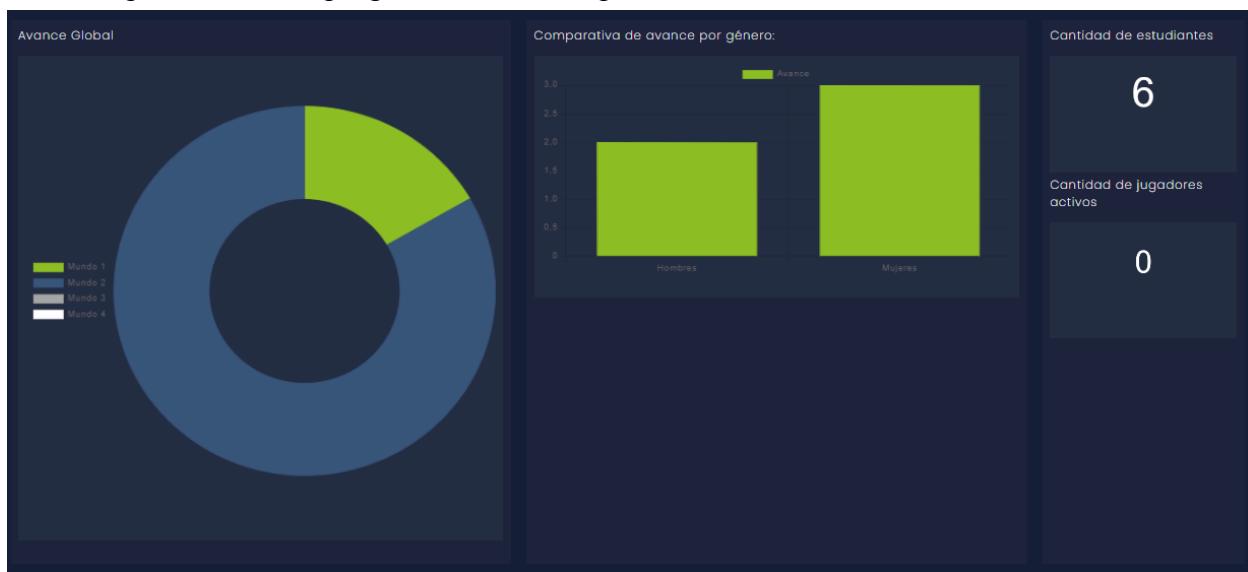
3.2.1 Vista de estudiante:

El estudiante podrá únicamente ver su avance, tomando en cuenta su progreso por puntaje

The screenshot shows a dark-themed dashboard for a student named "Viaje Intermatematico". At the top right are "INICIO" and "INICIAR SESIÓN" buttons. On the left is a logo with letters F and M. Below the header is a section titled "Visualización alumno" containing two tables. The first table has columns: Estudiante ID, Grupo, Número de lista, and Progreso. The second table has columns: Estudiante ID, Fecha, Nivel ID, and Valor.

3.2.2 Vista del profesor:

A diferencia del alumno, el profesor podrá ver un avance global del grupo por mundo, al igual que una gráfica que muestra el avance filtrado por género, una lista de estudiantes y finalmente, una lista que contiene su progreso, con la cuál podrá interactuar con un sistema de filtrado



| Maestro ID | Grupo | |
|------------|-------|-------------------------|
| 6 | | <button>Editar</button> |
| 7 | 331 | <button>Editar</button> |
| 9 | 332 | <button>Editar</button> |
| 15 | 331 | <button>Editar</button> |
| 16 | 332 | <button>Editar</button> |
| 17 | 331 | <button>Editar</button> |

| Lista de Grupos | |
|-----------------|---------------------------|
| Grupo | |
| 331 | <button>Eliminar</button> |
| 332 | <button>Eliminar</button> |
| 333 | <button>Eliminar</button> |
| | <button>Agregar</button> |

Lista de Estudiantes

| Estudiante ID | Número de lista | Grupo | Progreso | |
|---------------|-----------------|-------|----------|-------------------------|
| 1 | 1 | 332 | 1 | <button>Editar</button> |
| 2 | 2 | 331 | 1 | <button>Editar</button> |
| 3 | 3 | 331 | 1 | <button>Editar</button> |
| 4 | 1 | 333 | 1 | <button>Editar</button> |
| 5 | 2 | 333 | 1 | <button>Editar</button> |
| 10 | 7 | 331 | 13 | <button>Editar</button> |

Lista de Puntajes

Selecciona un criterio de filtro:

| Estudiante ID | Nivel ID | Fecha | Puntaje |
|---------------|----------|---------------------|---------|
| 1 | 1 | 2024-05-02 03:29:05 | 7 |
| 1 | 1 | 2024-05-02 03:29:05 | 3 |
| 1 | 2 | 2024-05-02 03:29:05 | 14 |
| 1 | 2 | 2024-05-02 03:29:05 | 8 |
| 1 | 3 | 2024-05-02 03:29:05 | 48 |
| 1 | 3 | 2024-05-02 03:29:05 | 23 |
| 1 | 4 | 2024-05-02 03:29:05 | 12 |
| 1 | 4 | 2024-05-02 03:29:05 | 5 |
| 1 | 5 | 2024-05-02 03:29:05 | 9 |
| 1 | 5 | 2024-05-02 03:29:05 | 1 |
| 7 | 10 | 2024-05-02 04:31:03 | 1 |
| 7 | 10 | 2024-05-02 04:31:33 | 0 |
| 7 | 10 | 2024-05-02 04:31:36 | 3 |

3.3 Pantallas

3.3.1 Inicio de sesión



3.3.2 Registro



3.3.3 Interfaz principal



Nosotros

¡Descubre un mundo de aprendizaje divertido con nuestro juego interactivo de figuras! Desde identificar formas hasta calcular áreas, perímetros y volúmenes, ¡prepárate para sumergirte en una experiencia educativa emocionante!



3.3.4 Sobre el juego



Nivel 1

Identificación de figuras

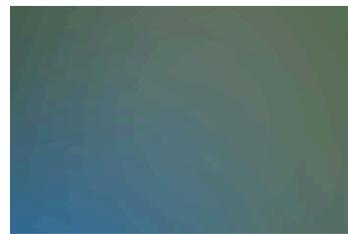
"En este emocionante juego, tu destreza para reconocer y atrapar las figuras requeridas será puesta a prueba. ¡Sumérgete en un desafío lleno de adrenalina donde deberás demostrar tu ojo visual."



Nivel 2

Perímetro

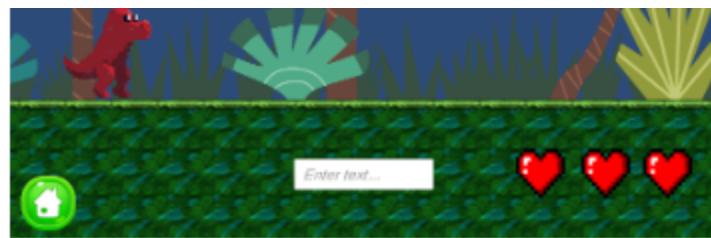
¡Únete a Castorín Fraccionelli en su emocionante misión de construir su propia casita! Tu tarea es llevar las diferentes figuras geométricas a las respuestas correctas de perímetro. ¡Demuestra tus habilidades en este



Nivel 3

Área

Sumérgete en las profundidades marinas y enfócate al desafío de pescar el pez que lleva incrustada la fórmula correcta del área en su



Nivel 4

Volumen

Embárcate en una emocionante aventura prehistórica y ayuda al simpático dinosaurio a seleccionar o escribir la respuesta correcta para que pueda disfrutar de un festín lleno de figuras y aprendizaje de volumen. ¡Únete a esta fascinante experiencia educativa y diviértete mientras exploras el mundo de los volúmenes junto a nuestro amigo jurásico!

3.3.5 Descarga del juego y manual de usuario

¡Descarga gratis NumNest!

¡Bienvenido a NumNest, el juego interactivo que transforma el aprendizaje de matemáticas en una aventura emocionante! Sumérgete en un mundo lleno de formas coloridas y desafíos matemáticos mientras exploras cuatro niveles emocionantes: identificación de figuras, cálculo de perímetros, áreas y volúmenes. ¿Estás listo para ayudar a Castorín Fraccionelli a construir su casita, pescar al pez con la fórmula correcta del área y alimentar al dinosaurio hambriento? Descarga NumNest ahora y descubre cómo el aprendizaje puede ser divertido, interactivo y gratificante. ¡Únete a la diversión matemática hoy mismo!

[¡Descarga ya!](#)

[Descarga el manual de usuario](#)

3.4 Código

<https://github.com/tc2005b-202411-442/team3-team3Web>

3.5 URL

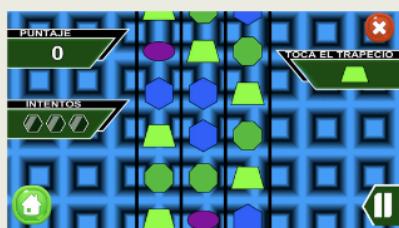
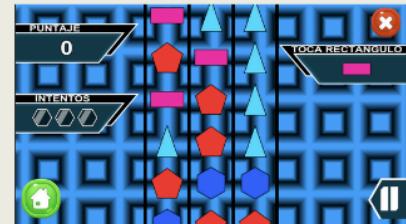
<https://rd-sudo.github.io/viajeintermatematico/index.html>

4. MODULO 4

4.1 Storyboard

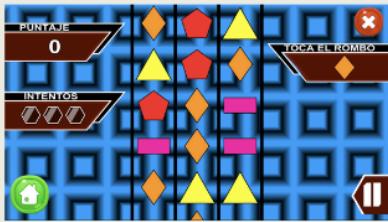
STORYBOARD

Nicole Kapellmann Lepine A01664563
Arely Yael Villatoro Amador A01663303
Dana Mendoza Palacios Roji A01658253
Pablo Ricardo Durán Sánchez A01663878



STORYBOARD

Nicole Kapellmann Lepine A01664563
Arely Yael Villatoro Amador A01663303
Dana Mendoza Palacios Roji A01658253
Pablo Ricardo Durán Sánchez A01663878



STORYBOARD

Nicole Kapellmann Lepine A01664563
Arely Yael Villatoro Amador A01663303
Dana Mendoza Palacios Roji A01658253
Pablo Ricardo Durán Sánchez A01663878



STORYBOARD

Nicole Kapellmann Lepine A01664563
Arely Yael Villatoro Amador A01663303
Dana Mendoza Palacios Roji A01658253
Pablo Ricardo Durán Sánchez A01663878



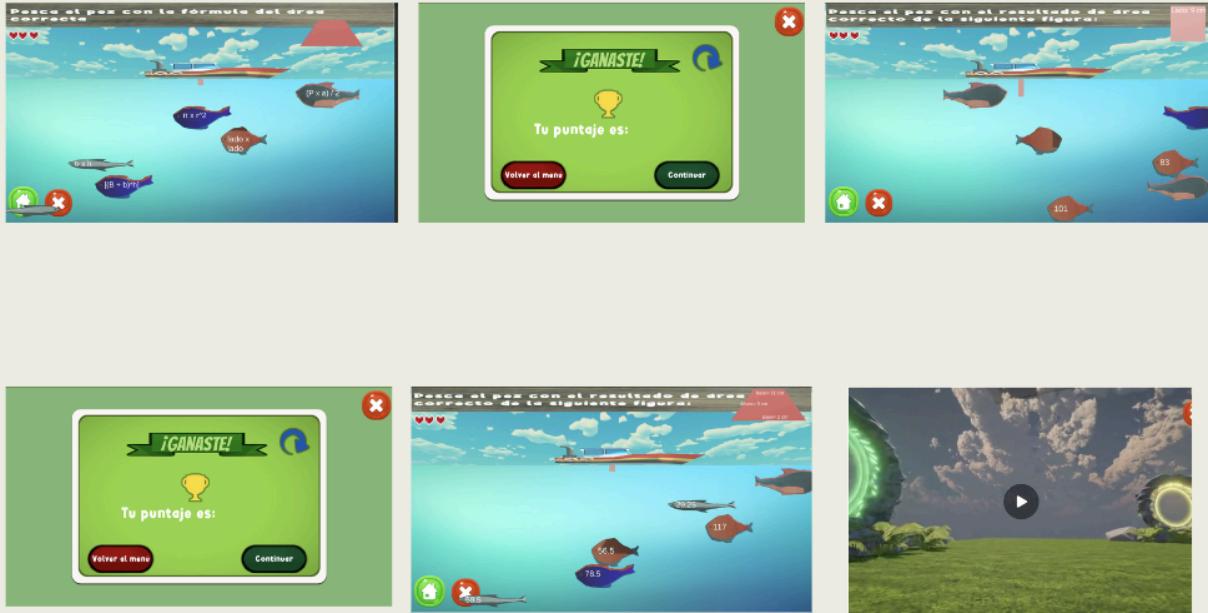
STORYBOARD

Nicole Kapellmann Lepine A01664563
Arely Yael Villatoro Amador A01663303
Dana Mendoza Palacios Roji A01658253
Pablo Ricardo Durán Sánchez A01663878



STORYBOARD

Nicole Kapellmann Lepine A01664563
Arely Yael Villatoro Amador A01663303
Dana Mendoza Palacios Roji A01658253
Pablo Ricardo Durán Sánchez A01663878



4.2 Código

<https://github.com/tc2005b-202411-442/team3-team3Unity>

4.3 Assets usados

- Alstra Infinite. (2021). *Fish - PolyPack* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/account/assets>
- amusedArt. (2018). *Fantasy Bee* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/packages/3d/characters/animals/fantasy-bee-135487>
- Avionx. (2022). *Skybox Series Free* [Asset 2D]. Unity Asset Store. Licencia: Extension Asset. Recuperado de: https://assetstore.unity.com/packages/2d/textures-materials/sky/skybox-series-free-10363_3#description
- BearStudios. (2022). *BearStudios Low Poly Pine Trees* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/packages/3d/environments/bearsstudios-low-poly-pine-trees-215102>

- B.G.M. (2022). *Arcade Game BGM #17* [Audio]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/audio/music/arcade-game-bgm-17-210775>
- B.G.M. (2019). *Casual Game BGM #5* Audio]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/audio/music/casual-game-bgm-5-135943>
- Dimension Dreams. (2017). *Touch The Color!* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/templates/touch-the-color-63404>
- Flixberry Entertainment. (2017). *Life Bar* [Asset 2D]. OpenGameAer.Org. Licencia: CC-BY 4.0 & CC-BY-3.0. Recuperado de:
<https://opengameart.org/content/heart-pixel-art>
- LaFinca. (2024). *Midgard Skybox* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/2d/textures-materials/sky/midgard-skybox-273733>
- Lee, S. (2017). *Pixel Art Dinosaur (With Animations)* [Asset 2D]. GDM. Licencia: Standard Pro Licence. Recuperado de:
<https://www.gamedevmarket.net/asset/pixel-art-dinosaur-with-animations>
- Lord Enot. (2023). *Pack Free Textures* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/2d/textures-materials/pack-free-textures-1-264695>
- Marya_Belevich. (2023). *Hyper casual mobile GUI* [Asset 2D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/2d/gui/hyper-casual-mobile-gui-268659>
- Mayo Games. (2020). *Pine forest set* [Asset 3D]. Unity Asset Store. Licencia: Extension Asset. Recuperado de:
<https://assetstore.unity.com/packages/3d/environments/pine-forest-set-free-sample-177774>
- Meshtint Studio. (2020). *Meshtint Free Burrow Cute Series* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/3d/characters/creatures/meshtint-free-burrow-cute-series-184837>
- MiMU Studio. (2017). *2D Casual UI HD* [Asset 3D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/packages/2d/gui/icons/2d-casual-ui-hd-82080>
- OArielG. (2019). *Simple Heart Health System* [Asset 2D]. Unity Asset Store. Licencia: Standard Unity Asset Store EULA. Recuperado de:
<https://assetstore.unity.com/account/assets>
- OnixGames. (2017). *Construct 2 Tileset* [Asset 2D]. OpenGameAer.Org. Licencia: Public Domain CC0. Recuperado de:

<https://opengameart.org/content/construct-2-tileset-4x-tilemaps-4x-backgrounds-4x-objects>

- Polygrunt Studios. (2020). *POLYGRUNT - LowPolyBoat* [Asset 3D]. Unity Asset Store. Licencia:Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/account/assets>
- RawPixel.com. (s.f.). *Diseño de fondo con textura de suelo de madera* [Imagen]. FreePik. Recuperado de: https://www.freepik.es/foto-gratis/diseno-fondo-textura-suelo-madera_16014620.htm#query=wood%20texture&position=4&from_view=keyword&track=ais&uuid=20137144-d060-46e2-a302-5dfb18f56635
- Sagital3D. (2017). *Desert Kits 64 Sample* [Asset 3D]. Unity Asset Store. Licencia:Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/account/assets>
- Simon Pasi. (2021). *Mini First Person Controller* [Asset 3D]. Unity Asset Store. Licencia:Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/packages/tools/input-management/mini-first-person-controller-174710>
- Tom Feldmann. (2024). *Old Cartoon Music Pack Free* [Audio]. Unity Asset Store. Licencia:Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/packages/audio/music/orchestral/old-cartoon-music-pack-free-277325>
- Werniech. (2024). *Stone Portal 1- Base* [Asset 3D]. Turbosquid. Licencia: Standard 3D Model License. Recuperado de: <https://www.gamedevmarket.net/asset/pixel-art-dinosaur-with-animations>
- yann. (2017). *Intergalactic Portal* [Asset 3D]. Unity Asset Store. Licencia:Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/packages/audio/sound-fx/intergalactic-portal-89506>
- Yuki2022. (2022). *Free Stylized Skybox* [Asset 3D]. Unity Asset Store. Licencia:Standard Unity Asset Store EULA. Recuperado de: <https://assetstore.unity.com/account/assets>