Informe POO-Desarrollo de Videojuego-“Monsters”

**Integrantes: Espejo Pizarro Nicolás**

**Vergara Castillo Gohan**

**García Felipe Brayan**

**Profesor: Miranda Escobar Esteban**

**Carrera: ICCI**

**índice**

1. **Introducción** ...........................................................................................................................
2. **Objetivos** …............................................................................................................................

Objetivos generales ….................................................................................................................

Objetivos específicos ..................................................................................................................

1. **Marco teórico** …....................................................................................................................

Definición de “clase”….......................................................................................................

4 Pilares de POO…...................................................................................................................

Lenguaje javascript…...................................................................................................................

Lenguaje Ruby…...................................................................................................................

1. **Equipo y materiales** ..............................................................................................................
2. **Desarrollo experimental** …...................................................................................................
3. Procedimiento :Juego en Javascript y ruby ….............................................
4. **Conclusión** ….........................................................................................................................
5. **Referencia Bibliográfica** …...................................................................................................

**1. Introducción**

En este informe se dará a conocer el “juego” creado con las directrices respectivas de POO (programación orientada a objetos), el cual contendrá todas sus características tanto en estructuración, fuentes y códigos de programación, el cómo se hizo “MONSTER”, las fuentes de información que se usaron y cómo funciona el código respectivo.

La Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener. Sigue el principio de desarrollo de software utilizado por muchos programadores **y** para [evitar duplicar](https://profile.es/blog/que-es-el-testing-de-software/) el código y crear de esta manera programas eficientes. Además, evita el acceso no deseado a los datos o la exposición de código propietario mediante la encapsulación y la abstracción.

Los pilares de POO son: Abstracción, polimorfismo, encapsulamiento y herencia, entre otros.

**2. Objetivos:**

**-Objetivos generales:**

-Explicar el “juego” diseñado en base a sus principios y códigos enseñados en clases de POO(programación orientada a objetos)

**Objetivos específicos:**

-Demostrar la funcionalidad del “juego-MONSTERS” en Javascript y Ruby

-Realizar una demostración Gráfica y visual de forma teórica y práctica; en el PC.

**3.-Marco teórico**

**Pilar 1** Encapsulación: **La encapsulación** es la característica de un lenguaje POO que **permite que todo lo referente a un objeto quede aislado dentro de éste**.

**Pilar 2** Abstracción: Como la propia palabra indica, el principio de abstracción lo que implica es que la clase debe **representar las características de la entidad hacia el mundo exterior, pero ocultando la complejidad** que llevan aparejada.

**Pilar 3** Herencia: Desde el punto de vista de la genética, cuando una persona obtiene de sus padres ciertos rasgos (el color de los ojos o de la piel, una enfermedad genética, etc...) se dice que los hereda. Del mismo modo **en POO cuando una clase hereda de otra obtiene todos los rasgos que tuviese la primera**.

**Pilar 4** Polimorfismo: El concepto de polimorfismo se refiere al hecho de que **varios objetos de diferentes clases, pero con una base común, se pueden usar de manera indistinta**, sin tener que saber de qué clase exacta son para poder hacerlo.

**lenguaje javascript:** Es el lenguaje de programación que debes usar para añadir características interactivas a tu sitio web, (por ejemplo, juegos, eventos que ocurren cuando los botones son presionados o los datos son introducidos en los formularios, efectos de estilo dinámicos, animación, y mucho más).Fue inventado por Brendan Eich, cofundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla.

Puedes hacer casi cualquier cosa con JavaScript. Puedes empezar con pequeñas cosas como carruseles, galerías de imágenes, diseños fluctuantes, y respuestas a las pulsaciones de botones. Con más experiencia, serás capaz de crear juegos, animaciones 2D y gráficos 3D, aplicaciones integradas basadas en bases de datos ¡y mucho más!. JavaScript por sí solo es bastante compacto aunque muy flexible, y los desarrolladores han escrito gran cantidad de herramientas encima del núcleo del lenguaje JavaScript, desbloqueando una gran cantidad de funcionalidad adicional con un mínimo esfuerzo.

El mundo de posibilidades que nos abre Javascript es muy extenso y hoy en día, es casi imprescindible para el desarrollo de páginas webs. Como todo, tiene sus inconvenientes y sus limitaciones, pero gracias a la gran cantidad de librerías que existen se pueden hacer grandes cosas.

Javascript es uno de los lenguajes de programación más sencillos que existen, y cómo has podido ver en los ejemplos anteriores, su rapidez permite ejecutar funciones de manera inmediata.

Con la revolución tecnológica, los [**datos**](https://www.tecnologias-informacion.com/mineria-de-datos.html) se han convertido en el activo más valioso. La validación de los datos de un formulario es fundamental, ya no solo para asegurarnos de que los datos que estamos recogiendo son reales, sino también para evitar inyecciones a base de datos, o bots metiendo información falsa. Javascript es una herramienta que nos permite realizar validación de datos a un primer nivel.Aunque la validación de datos siempre se debe hacer del lado del servidor, no está de más añadir una doble validación con Javascript, porque así el usuario puede ver al momento los fallos, sin tener que esperar a enviar el formulario. Las validaciones que se suelen hacer son: tipo de datos (por ejemplo, en un campo email, verificar que efectivamente es tiene un formato de correo electrónico, es decir un @), extensión de documentos, tamaño de archivos subidos o longitud de caracteres, entre otras muchas opciones.

**Lenguaje Ruby:** El lenguaje de programación Ruby se considera uno de los más simples dentro del universo de programación. También se considera reciente, fue creado por el programador japonés Yukihiro Matsumoto en 1995 y permite un desarrollo ágil y rápido de las aplicaciones.Ruby ha ganado una gran comunidad de usuarios debido a su combinación de legibilidad fácil de usar con gran complejidad técnica. Esto atrajo a muchos desarrolladores, que buscaban un lenguaje capaz de desarrollar muchas aplicaciones pero que fuera simple y preciso de entender.

El lenguaje Ruby se utiliza principalmente en el desarrollo de aplicaciones web, pero también se puede utilizar para desarrollar otro tipo de aplicaciones de software, como veremos más adelante. Este lenguaje está disponible en plataformas como Windows, Linux y muchas otras, considerándose multiplataforma.

Está orientado a objetos y funciona con datos que el usuario creará en función de sus propias necesidades. Los objetos son formas de representar objetos de la vida real dentro del software.Muchas aplicaciones famosas utilizan el lenguaje Ruby, como los sitios web de Airbnb y SoundCloud, la red social Twitch y el sitio web de streaming de series y películas Hulu.

El lenguaje Ruby puede ser muy útil, ya que este lenguaje se puede utilizar para crear varios programas para desktop. Hoy en día, es uno de los lenguajes más utilizados para el desarrollo de aplicaciones back-end en el mundo.**Ruby on rails** fue creado para desarrollar aplicaciones web para startups y se inspiró totalmente en el desarrollo ágil utilizado en este tipo de empresas. Hoy en día, este framework también se usa ampliamente para desarrollar aplicaciones como aplicaciones comerciales y corporativas, sistemas de ventas y otras aplicaciones utilizadas por Internet.El usuario que usa el lenguaje Ruby para crear aplicaciones web puede acceder a RubyGems, que es el sistema de empaquetado de Ruby. RubyGems facilita la distribución, instalación y actualización de aplicaciones en su sistema.

A diferencia de otros lenguajes de programación, programar en Ruby no requiere, por ejemplo, caracteres de terminación para una declaración o que declare variables. La variable solo necesita tener un nombre y un valor asignado. Algunos tipos de variables en el lenguaje Ruby: Constantes - mayúscula, Locales - minúscula Instanciado - @, classes - @@, Globales - $.

**4. Equipo y materiales**

Materiales:

1. PC
2. Programación en lenguaje Javascript.
3. Programación en lenguaje Ruby.

**5. Desarrollo experimental**

Procedimiento:

Utilizamos VS CODE (Es mas comodo y definitivamente más conveniente) para dar lugar a los códigos

Empezamos con Javascript:

1. Primero, ve a tu sitio de pruebas y crea una carpeta llamada scripts. Luego, dentro de la nueva carpeta de scripts, crea un nuevo archivo llamado main.js y guárdalo.
2. A continuación, abre tu archivo index.html y se introduce en el las líneas de código, justo antes de la etiqueta de cierre </body>:
3. Esto hace básicamente el mismo trabajo que el elemento <link> para CSS: aplica el código JavaScript a la página, para que pueda actuar sobre el HTML (y CSS, o cualquier cosa en la página).
4. Ahora añade el código al archivo main.js:
5. Finalmente, asegúrate de que has guardado los archivos HTML y JavaScript, y abre index.html en el navegador.

Por otro lado tenemos en Ruby:

1. Una vez hayamos instalado Ruby lo abrimos en la máquina y ejecutamos.

2. Se puede editar un fichero con extensión .rb para programar.

3. Con el touch creamos el fichero y con el nano lo editamos y lo guardamos.

4. Se puede utilizar la función nativa puts, que imprime una línea de texto por la pantalla y se le pasa una cadena de texto, es decir, un objeto de tipo string. Ejecutamos el fichero con la extensión .rb

6. Se pueden crear símbolos (symbol), un tipo de dato que tiene Ruby muy parecido a los string.

7. También están en Ruby los Arrays. En ellos se puede añadir elementos de diferentes tipos, varios elementos a la vez, se puede acceder a un elemento, entre otras acciones.

8. En Ruby en ningún momento se define el tipo del objeto sobre el que estes trabajando, en el momento de la definición es cuando el intérprete de Ruby hace la asignación.

Apartado - **classes.mjs**

1.Se empieza a escojer elementos aleatorios de una lista, y mediante el uso de “name generator” genera nombre; mediante letras y consonantes, genera sílabas para asegurar que los nombres sean pronunciables, esto mediante el comando “for”y el aleatorio “randomInt” y “choose”. El cual retorna el nombre respectivo. Con esto se crea el Nombre del Individuo.

2. Se observa La creación de otra “clase”; la de los monstruos, con características como:agua, planta, fuego, el cual contiene “ataque, defensa, estamina, vida,” entre otros.

Se le da atributos de color a estos monstruos de agua, planta y fuego.

En “strongAttack” También observamos las funciones de ataque y las características respectivas con el comando “weakAttack”, tanto de 5 en las habilidades de combate como también toca el área de la “estamina”

En funciones como: En “damage” se juega con los atributos de “hp”, “defensa”, y mediante la función “if” pone en tabla la “estamina”, donde mediante la función “rest” juega con el “hp”, estamina y “max5tamina”

También observamos la función “status” el cual mediante el condicional “if” y “elseif” toca los atributos de “color” para influir en el “daño”, ataque, defensa del “Animal”. Con todas estas características se da mayor Identidad y función, el cómo se va a comportar el Animal.

Apartado -**main.mjs**

**Cosas a Resaltar.**

1. Primero se empieza importando “Monster, nameGenerator, choose, randomInt” de la “**classes.mjs”**

2. En donde tomamos condicionales como “for” y randomizar el ramdom.

3. Mediante la función “function enemyMove” usamos datos como “myself, enemy, choice”

En donde esta función toma como inputs a un monstruo tuyo y uno salvaje = 4 acciones y hace que el enemigo las realice.

Tenemos 4 casos = “usa ATAQUE FUERTE”, " intenta BLOQUEAR" y " se pone a DESCANSAR".

Luego Procedimos a generar una lista de cosas importantes, tipo biomas

["DESIERTO", "PANTANO", "BOSQUE"];["TEMPLO", "CASTILLO"];["FUEGO", "AGUA", "PLANTA"];

5.(esta función toma uno de tus monstruos y un monstruo salvaje y te deja pelear), Si se ganó o no Nos retorna un resultado.

Aquí empezamos Con acciones como: Mostrar la “info de los participantes, el cual Te pone el tabla cuales son tus acciones

Las opciones tienen que estar dentro del rango-opción tiene que ser un número entre 1 y 5

por lo tanto mediante el condicional “if” controlamos diversos datos, unos que nos pueden servir y otros que no.

5. casos como : “Usa ataque Debil” " usa ATAQUE FUERTE", " intenta BLOQUEAR" y " se pone a DESCANSAR" ,

También mensajes certeros como:

console.log("\x1b[0m\nMONSTERS! es un juego de peleas por turnos donde controlas");

console.log("un equipo de 4 monstruos contra una serie de monstruos");

console.log("salvajes camino a tu destino.\n");

console.log("Tu objetivo como su guía es darle ordenes a tus monstruos");

console.log("para poder derrotar a cada monstruo salvaje.\n");

6. En donde ya Podemos Ubicarnos en qué Consiste el juego y EMPEZAR con esa breve y EFICAZ descripción a Jugar el juego o bien Entender en qué Consiste./Datos de la partida.

diferentes escenarios como:

“ ("1. Modo historia\n2. Modo arena\n3. Sobre el juego\n4. Salir\n > ")); prompt("") “

7. A este punto vamos a tomar un “ switch (option) “ para desplegar opciones como variar la dificultad que el jugador quiere. Esta dificultad afortunadamente es Amplia y se puede variar de 5 en 5

Ahora podemos observar mensajes de instrucciones como:

console.log("\nEstás en un " + biome + ", tienes un largo camino por delante.");

console.log("Tu destino es el " + target + ", donde se rumorea está escondido un gran tesoro.");

Con esto ya podemos ver claramente la finalidad y lo emocionante que será todo.

También se ve el Bioma y cómo este influye en la generación de más elementos, el bioma determina qué elementos predominan sobre otros.

también se puede observar mediante mensajes de alguna “baja” de algún personaje. Obviamente todo va de la mano con sus “if” entre otras características, mensajes como:

“ console.log("\nSin tus monstruos, acabas sucumbiendo a los monstruos salvajes."); “

(mensaje de “baja” de tu “compañero”)

También se puede observar si un Monstruo se une a tu batalla.

En esencia Aquí Se desarrolla en parte El código para saber el comportamiento de los Monstruos.

8. Por último tenemos los créditos respectivos y la finalización del Juego.

case 3:

console.log("\x1b[2J\x1b[1;1H--- CREDITOS ---\n");

console.log("Codigo: Nicolas Espejo");

console.log("Diseño: Nicolas Espejo, Gohan Vergara");

wait();

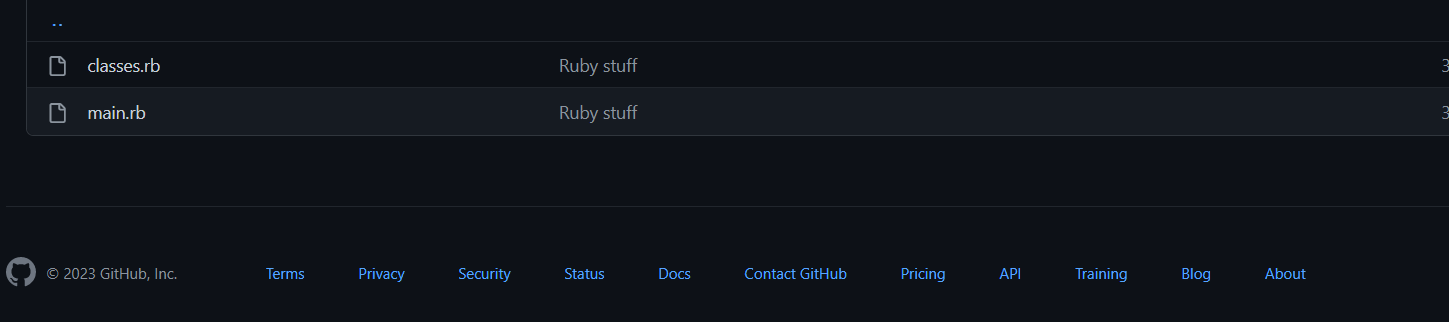
break;

case 4:

console.log("\x1b[2J\x1b[1;1HFIN\n");

exit();

En caso del lenguaje Ruby el Código afortunadamente ya está hecho y funcional, lo cual se traspasará al lenguaje Ruby



MONSTERS! es un juego de peleas por turnos donde controlas un equipo de 4 monstruos contra una serie de monstruos salvajes camino a tu destino Tu objetivo como su guía es darle órdenes a tus monstruos para poder derrotar a cada monstruo salvaje, tus monstruos pueden sufrir altercados entre otras cosas donde Te expones a una ardua y emocionante batalla para asegurar la Sobrevivencia.

* Se utilizó los pilares de Abstracción y encapsulamiento

Conclusión: El trabajo fue Satisfactorio, así como la funcionalidad del juego, cada proceso en la creación fue satisfactorio, fue un reto el que se haya hecho todo esto, así como una excelente experiencia para entender La programación orientada a objetos.

**8. Referencias Bibliográficas**

**Webgrafía:**

[**https://ruby-doc.org/**](https://ruby-doc.org/) **y** [**https://www.geeksforgeeks.org/javascript/**](https://www.geeksforgeeks.org/javascript/)

[**https://www.campusmvp.es/recursos/post/los-conceptos-fundamentales-sobre-programacion-orientada-objetos-explicados-de-manera-simple.aspx**](https://www.campusmvp.es/recursos/post/los-conceptos-fundamentales-sobre-programacion-orientada-objetos-explicados-de-manera-simple.aspx)

[**https://aws.amazon.com/es/what-is/javascript/**](https://aws.amazon.com/es/what-is/javascript/)

[**https://desarrolloweb.com/articulos/poo-fundamentos-luis-fernandez.html**](https://desarrolloweb.com/articulos/poo-fundamentos-luis-fernandez.html)

[**https://developer.mozilla.org/es/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics**](https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics)