Tabla de contenido

Press Start

¿Qué es Phaser ?

Phaser es un framework Open Source para la creación de juegos de escritorio y móviles basado en HTML5.

Utiliza WebGL y canvas para el renderizado, y alterna de forma automática entre ellos según la compatibilidad del navegador. La librería base JavaScript que utiliza para esta acción es **Pixi.js**, *otro framework similar, tambien para la creación de juegos.*

¿ Qué ofrece ?

Herramientas	Descripción
Físicas	Tres motores físicos separados (ARCADE, NINJA, P2)
Cámara	Sistema de cámara de seguimiento o total
Inputs	Sistema de entrada ratón, teclado, tactil y gamepad
Sonido	Administrador de sonido (volumen, bucle, velocidad)
Preloader	Carga de datos, antes del renderizado
Datos de mapa de bits y tilemaps	Mapeado de imágenes y sprites
Video	Soporte de video
Interpolación	Soporte de webGL y Canvas
Partículas	Sistema de generación de particulas
Animación	Varios sistemas de animación de imágenes sprites y texto

---

Ventajas

- **Open Source.**
- Gran comunidad.
- muchos y buenos tutoriales.
- Actualización **diaria** de ejemplos y nuevos juegos.

- **Responsive.**
- Facilidad de crear un juego en muy poco tiempo.
- Sistema de plugins.
- Todas las herramientas disponibles.

Desventajas

- 3D no implementado.
- Alguna herramienta tiene algún pequeño error.
- Se puede quedar pequeño, dependiendo de la ambición del proyecto.

¿ Por Qué Phaser ?

Si de verdad tuviera que decir algo malo de Phaser, seria que tiene un potencial muy alto y no esta aprovechado. La gran comunidad presente y el hecho de que sea open source , acerca mucho al desarrollador frente a otros HTML engines.

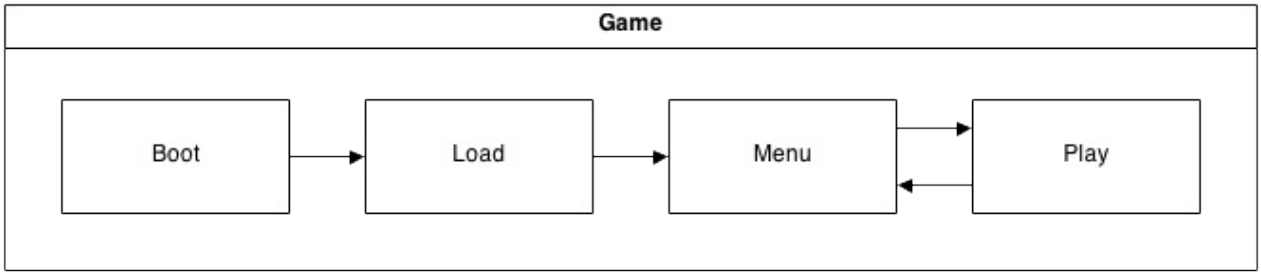
Muchos engines como puede ser Construct2 o Game Maker simplemente son editores (sin código), otros como Impact ni siquiera son Open Source y/o simplemente no tienen una comunidad tan activa y grande.

Game:

Aquí es donde la magia ocurre. El objeto Game es el corazón de tu juego, que provee acceso rápido a las funciones comunes y maneja el proceso boot.

- El objeto **Game es el controlador principal de todo el juego** . Es responsable de manejar el proceso de arranque, analizar los valores de configuración, crear el renderizador y configurar todos los sistemas de Phaser, como **las físicas, el sonido y los inputs**. Una vez que se haya completado, se iniciará el estado predeterminado y se iniciará el bucle del juego principal.
- Dondequiera que se pueda tener acceso a las característicsa del juego, se podrá tener acceso a todos estos sistemas de la base.

Una estructura de juego seria similiar a esto:



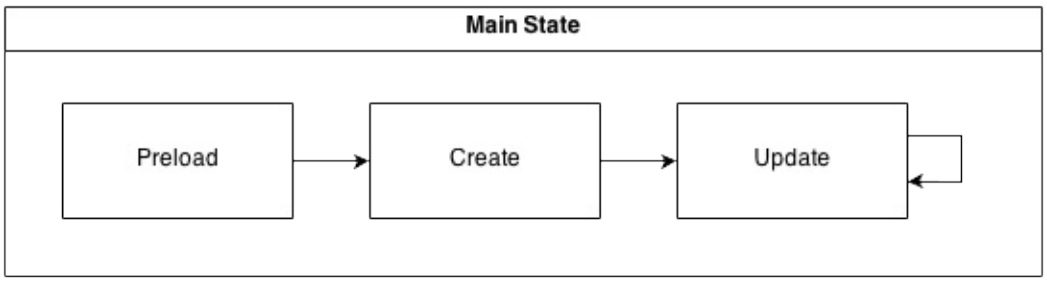
States:

La clase State es una parte/s fundamental/es de tu juego. Te da acceso rápido a las funciones comunes tales como: la cámara, cache, input, sonido, etc.

Cada State se compone de una serie de métodos básicos para su manejo. Lo mas importantes son:

- **Init:** Es la primerísima función llamada cuando el State comienza. Es decir que se le llama antes que a preload, create o cualquier otro. Si necesitas enrutar Game hacia otro State puedes hacerlo aquí, o si necesitas preparar el set de variables o de objetos antes de que preloading empiece.
- **Create:** Es llamado una vez que preload se ha completado, esto incluye el tiempo de cargado de cualquier asset del loader. Si no tienes el método preload, entonces create es el primer método al que se llama en tu State.
- **Preload:** Es llamado primero. Normalmente usarías esto para cargar los assets del juego (o los necesarios para el actual State). No deberías crear ningún objeto en este método que requiera assets que estés cargando en este, porque puede que no estén disponibles.
- **Render:** Casi todos los objetos de visualización en Phaser se renderizan automáticamente, no necesitas decirles que se rendericen. De todas formas el método render es llamado después de que el juego y los plugins se hallan renderizado.
- **Update:** Es un método que está vacío para tu propio uso. Este se ejecuta 60 veces por segundo, en este método es donde se añade toda la lógica de State, vease interactuar con un menu o personaje.

Una estructura similar a un State sería esto:



Text:

Crea un nuevo objeto del juego para visualizar texto.

Usa un objeto Canvas oculto y lo renderiza en él. Luego hace una textura para el renderizado de la vista. Por esto mismo sólo puedes visualizar fuentes que están actualmente disponibles en el navegador.

Image:

Crea una nuevo objeto del juego para visualizar una Imagen.

Una Image es un objeto ligero que se puede utilizar para mostrar cualquier cosa que no necesite físicas o animación. Todavía puede se girar, escalar, recortar y recibir eventos. Esto lo hace perfecto para logos, fondos, botones simples y otros gráficos que no sean Sprite.

Sprite:

Los sprites son el alma del juego, usados para casi todo lo relacionado con la parte visual.

Consiste de un set de coordenadas y una textura que se renderizan hacia el canvas. También contienen propiedades adicionales permitiendo las físicas de este o la animación.

Herramientas

Físicas :

una de las mejores características de Phaser son sus físicas. En total hay tres tipos diferentes. las *physics engines* sin aquellas que gestionan las colisiones y el movimiento de todos los objetos del juego.

- P2 es un sistema completo de físicas que nos permite hacer juegos con colisiones complejas como Angry Birds.
- Ninja, es menos potente que P2, pero tiene algunas características interesantes para manejar tilemaps e inclinaciones.
- Arcade es el sistema mas básico de físicas que solo trata con colisiones con rectángulos (llamados AABB), aun así, son las que tienen mejor rendimiento.

Cámara:

La cámara de Phaser es básicamente lo que ve el jugador:

- Cámara fija, la cual se utiliza cuando quieres mostrar la totalidad del juego.
- Cámara con seguimiento. Sigue el movimiento de un objecto determinado.

También permite tener elementos fijos. Esto viene muy bien cuando quieres implementar una interfaz de usuario.

Además cuenta con una serie de efectos como flash o apagado.

Inputs:

existen varias formas de interactuar con el juego.

- mediante un ratón, lanzar acciones mediante clicks.
- mediante un teclado. Asignas a cada tecla una acción.
- Táctil. Básicamente es el control para dispositivos móviles.
- mediante un gamepad, al igual que el teclado, le asignas a cada botón una acción.

Sonido:

Permite el manejo de archivos de sonido, pudiendo reproducir, poner en bucle bajar volumen, etc..

las principales extensiones de los archivos de sonido son wav, mp3 y ogg. Básicamente depende de la compatibilidad con el navegador.

Datos de mapa de bits y tilemaps:

Una herramienta esencial a la hora de crear animaciones y generación de mapas.

los mapas de bits o atlas son lo que, a partir de una imagen formada por varias, unidas en una, recoge la los puntos exactos para mostrar, permitiendo así realizar animaciones.

los tilemaps son los que a partir de una imagen, pinta un mapa o *mundo.*

Vídeo:

Al igual que el sonido, permite el manejo de archivos de vídeo.

Las extensiones son, al igual que el sonido, aquellas que permita el navegador.

Partículas:

se generan a partir de *Phaser Particle Storm* : Un avanzado sistema de partículas que permite crear efectos, como fuego o explosiones.

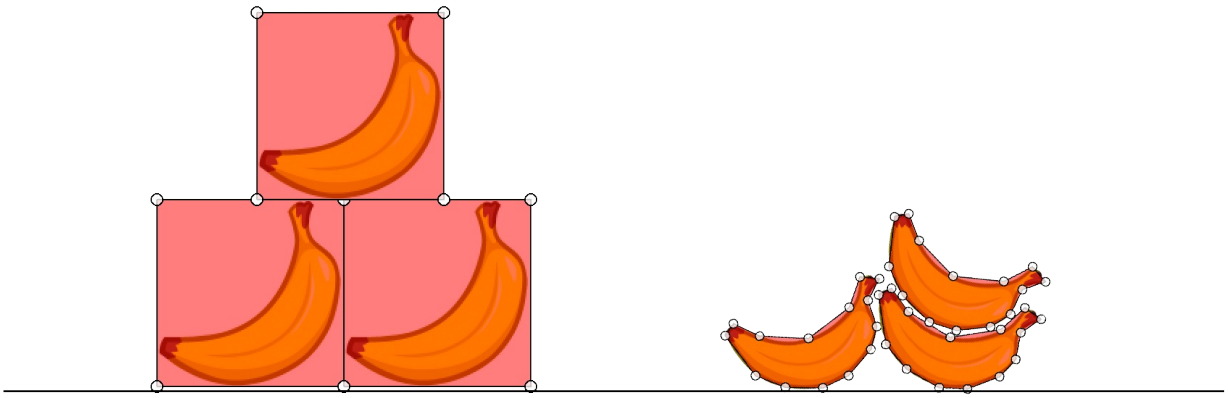
Animaciones:

Otra gran herramienta de Phaser son las animaciones, básicamente es las que le dan "vida" al juego. Permite, a partir de una imagen o imágenes, recrear movimiento.

Las animaciones pueden ser aquellas que dan movimiento a una imagen o hacer, como en HTML, una transición como fade in o transparencia por ejemplo.

Físicas p2:

la diferencia principal con el resto de físicas, es que permite crear interacciones físicas mucho más complejas y colisiones también, más complejas. Cuando se utiliza el sistema Arcade Physics, la hitbox de un sprite siempre será un simple rectángulo delimitador:



Toda esa funcionalidad adicional tiene un costo de rendimiento. Dado que las físicas son mucho más complejas, se necesita mucho más poder de procesamiento, y esto puede ser especialmente notable cuando se ejecuta el juego en dispositivos móviles.

No solo nos encontramos con esto, también está el echo de que ya no existe la funcionabilidad que proporcionaba Phaser para la física ARCADE.

- Ya no hay collisiones con objetos de forma natural para realizar ciertas acciones, ejemplo : saltar si tocas el suelo. Para arreglar hay que recoger los datos del objeto colisionador con el que colisiona para para determinar si esta en contacto o no.
- tampoco hay detección natural de o*verlapping, por lo tanto hay que comprobar los dos cuerpos en colision y determinar si entre ambos ha de haber overlap o no.*

Un dato muy interesante es el echo de que un objeto atraviese a otro si llega a cierta velocidad. Este proceso no se debe a un error sino a como está construida la física P2, dando el efecto llamado "tunneling".

En P2, hay una funcionalidad de CCD (Continuous Collision Detection) implementada, pero parece que está señalado como experimental (https://github.com/schteppe/p2.js/releases/tag/v0.6.1).Hay un plugin disponible que se trata

de una nueva física llamada Box2d que no es gratuita y se ha implementado por defecto.

BIBLIOGRAFÍA:

- **Documentación : https://phaser.io/docs/2.6.2/index**
- **Ejemplos: https://phaser.io/examples**
- **Latest News: https://phaser.io/**
- **Libro de Phaser : https://phaser.io/shop/books/discover-phaser**
- **Foro preguntas y respuestas : http://www.html5gamedevs.com/**
- **mapa de bits : https://www.leshylabs.com/apps/sstool/**
- **boilerplate: https://github.com/lean/phaser-es6-webpack**