Intro a Phaser 3

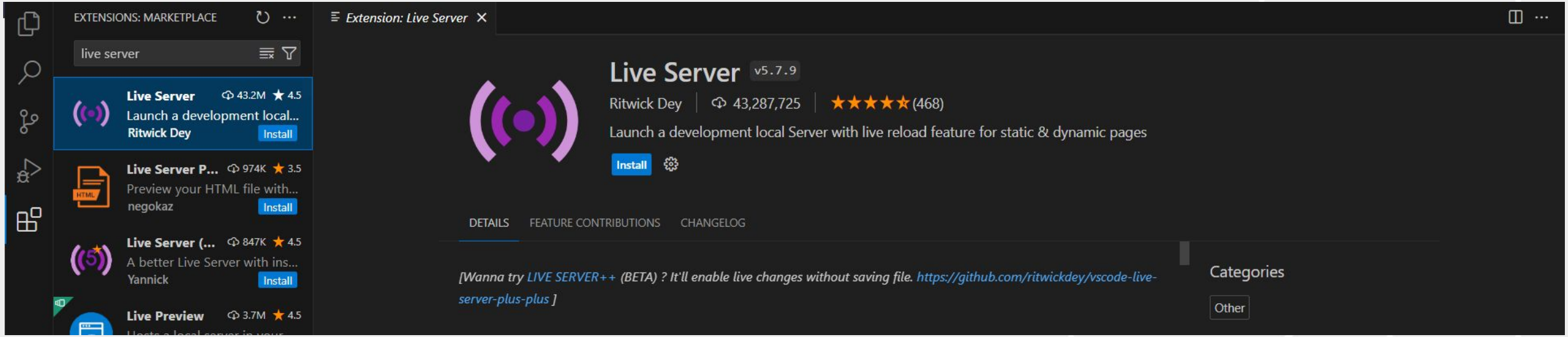
Update: 1/2/2024

Mst. Erick Brenes.

Que haremos

- Crearemos el proyecto pero usando NPM.
- Haremos el manejo de escenas.
  - Forma manual
  - Uso de módulos

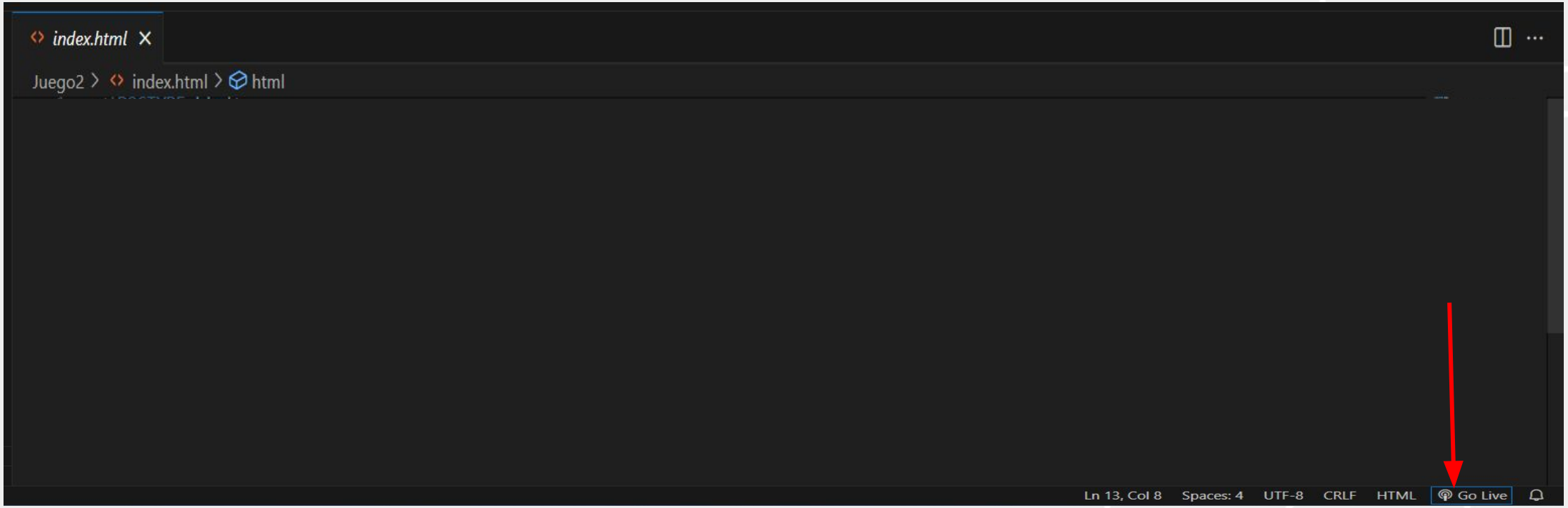
Pre: instalación de servidor



Nuestros juegos necesitan un servidor para poder funcionar. Los juegos no se pueden ejecutar con un doble click en el index.html para esto usamos NodeJS que es descargar e instalar.

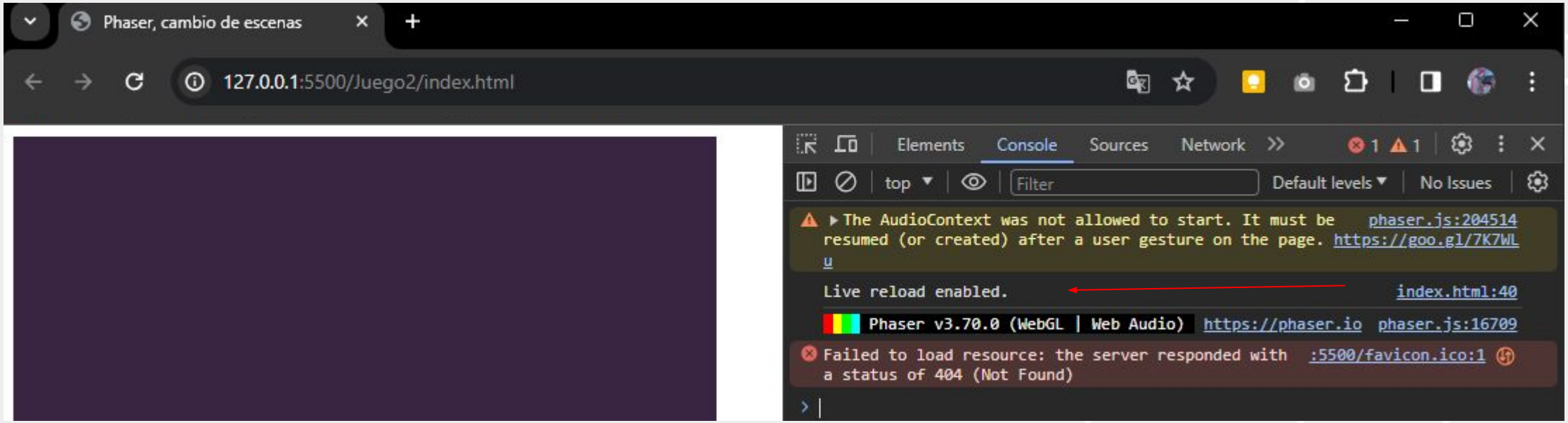
Live Server es un plugin de Visual Studio Code que permite levantar un servidor de forma rápida.

Pre: probar el live server



Tras instalar y reiniciar el visual studio (no siempre requerido). Al abrir un html podrán ver en la parte inferior un *go live*, si le dan click abrirá su browser predeterminado con el código ejecutado en un servidor local.

Pre: probar el live server



Si en esa pestaña abren el *developer console* observar la impresion que dice live reload enabled. Ahora cada vez que le den save al proyecto este tab se recargará.

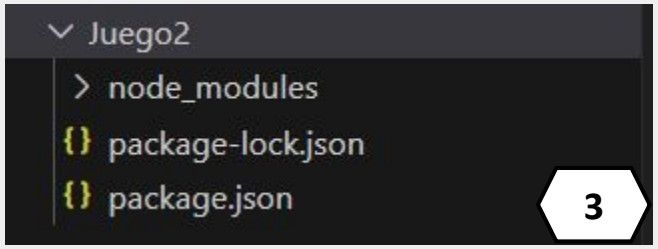
Manos a la obra

1. Instalación por medio de consola

```
PS D:\Desarrollo\HTML\Phaser\Juego2> npm init -y
Wrote to D:\Desarrollo\HTML\Phaser\Juego2\package.json:

{
  "name": "juego2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
```

```
PS D:\Desarrollo\HTML\Phaser\Juego2> npm install --save phaser
[###############....] | reify:eventemitter3: http fetch GET 200 https://registry.npmjs.org/eventemitter3/-/eventemitter3-5.0.1.tgz 414ms (cache miss)
```



En la carpeta del proyecto, en la consola colocamos:

- npm init -y
- npm install --save phaser

Al finalizar la carpeta tendrá los archivos de la tercer imagen

Carga de escenas manual

2.1 - El Index

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Phaser, cambio de escenas</title>
    </head>
    <body>

        <script src="./node_modules/phaser/dist/phaser.js"></script>
        <!-- Region => Seccion para las escenas -->
        <script src="./Scripts/Scenes/escena1.js"></script>
        <script src="./Scripts/Scenes/escena2.js"></script>
        <script src="./Scripts/Scenes/escena3.js"></script>
        <!-- Seccion para las escenas <= Region-->
        <script src="./Scripts/init.js"></script>
    </body>
</html>
```

Nuestro index debe de contener lo básico de un html y aparte los scripts que vamos a ocupar que serían el init de configuración y las escenas.

2.2 - Archivo de configuración básica

Juego2 > Scripts > JS init.js > ...

```
1   const config = {
2       width:  500,
3       height: 400,
4       parent: "container",
5       type:   Phaser.AUTO,
6       backgroundColor:    '#392542',
7
8   }
9
10  new Phaser.Game(config);
```

Este archivo por el momento tiene lo básico para funcionar que es tamaño y tipo. Las otras opciones se usan para cambiar el color y decir donde va a estar localizado el juego.

2.3 - Clase base para las escenas

Juego2 > Scripts > Scenes > JS escena1.js > ...

```
1   class escena1 extends Phaser.Scene{
2       constructor(){
3           super({key:"escena1"});
4       }
5
6       preload(){}
7
8       create(){}
9
10      update(time, delta){}
11  }
```

Estas cuatro funciones son la definición de una clase para que una escena funcione.

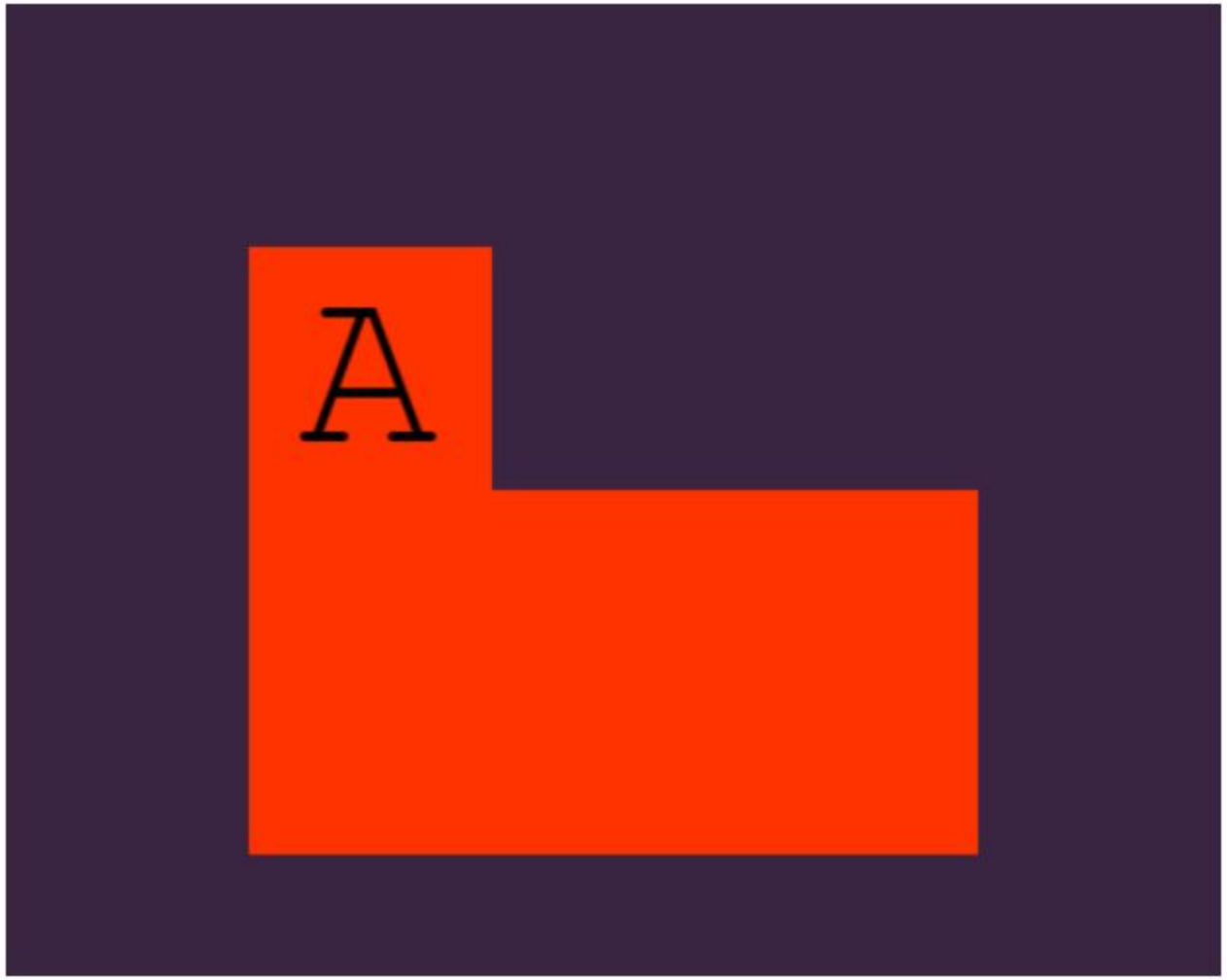
- *Preload* carga los elementos antes de iniciar la ejecución.
- *Create* permite crear los elementos y hacer el llamado a funciones.
- *Update* es la función que se ejecuta después de creado y que revisa observadores en tiempo de ejecución

2.4 - Escena 1

Juego2 > Scripts > Scenes > JS escena1.js > ...

```
1   class escena1 extends Phaser.Scene{
2       constructor(){
3           super({key:"escena1"});
4       }
5
6       preload(){}
7
8       create(){
9           let graphics = this.add.graphics();
10          graphics.fillStyle(0xff3300,1);
11          graphics.fillRect(100,200,300,150);
12          graphics.fillRect(100,100,100,100);
13
14          this.add.text(120,110,"A", {font: "96px Courier", fill: "#000"});
15      }
16
17      update(time, delta){}
18  }
```

En esta escena mostramos una caja superior con una letra y una caja inferior más grande que sirve de contenedor.



Este es el resultado de lo que acabamos de hacer.

2.5 - Escena 2 y Escena 3

Juego2 > Scripts > Scenes > escena2.js > ...

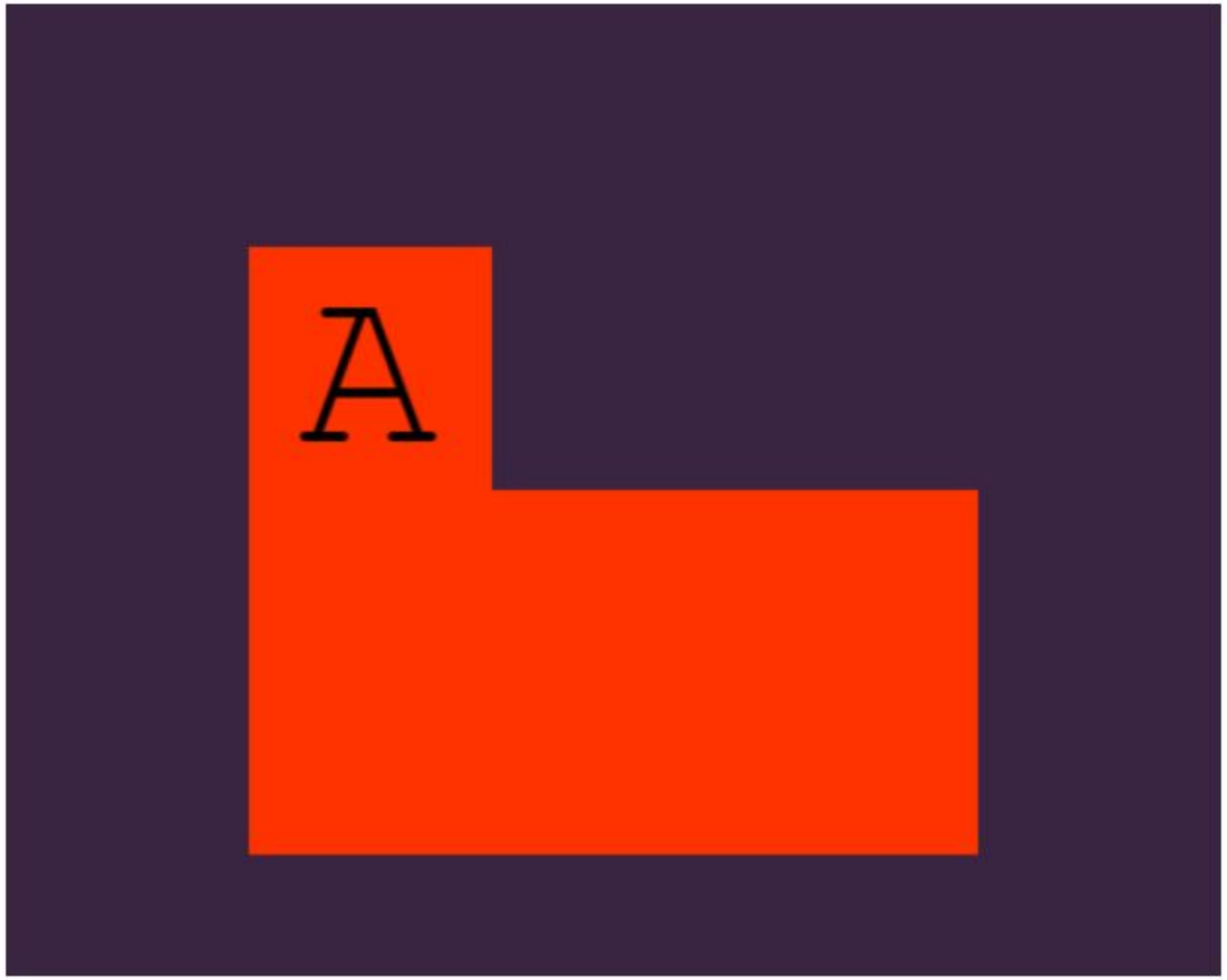
```
1  class escena2 extends Phaser.Scene{
2      constructor(){
3          super({key:"escena2", active: true});
4      }
5
6      preload(){}
7
8      create(){
9          let graphics = this.add.graphics();
10         graphics.fillStyle(0xff3399,1);
11         graphics.fillRect(100,200,300,150);
12         graphics.fillRect(200,100,100,100);
13
14         this.add.text(220,110,"B", {font: "96px Courier", fill: "#000"});
15     }
16
17     update(time, delta){}
18 }
```

Juego2 > Scripts > Scenes > escena3.js > ...

```
1  class escena3 extends Phaser.Scene{
2      constructor(){
3          super({key:"escena3"});
4      }
5
6      preload(){}
7
8      create(){
9          let graphics = this.add.graphics();
10         graphics.fillStyle(0xff9999,1);
11         graphics.fillRect(100,200,300,150);
12         graphics.fillRect(300,100,100,100);
13
14         this.add.text(320,110,"C", {font: "96px Courier", fill: "#000"});
15     }
16
17     update(time, delta){}
18 }
```

Hacemos una copia del código de escena 1 y lo pegamos en escena 2 y escena 3
Cambiamos los valores:

- Constructor
- Línea 10 (color de caja)
- Línea 12 (donde inicia la caja superior)
- Línea 34 (Letra y donde va a estar colocada)



Si salvamos y vamos al browser se verá igual. Esto por que no agregamos las escenas en el config y tampoco le dijimos que cada escena está activa.

2.6 - Escena 2 y Config file

escena2.js

Juego2 > Scripts > Scenes > escena2.js > ...

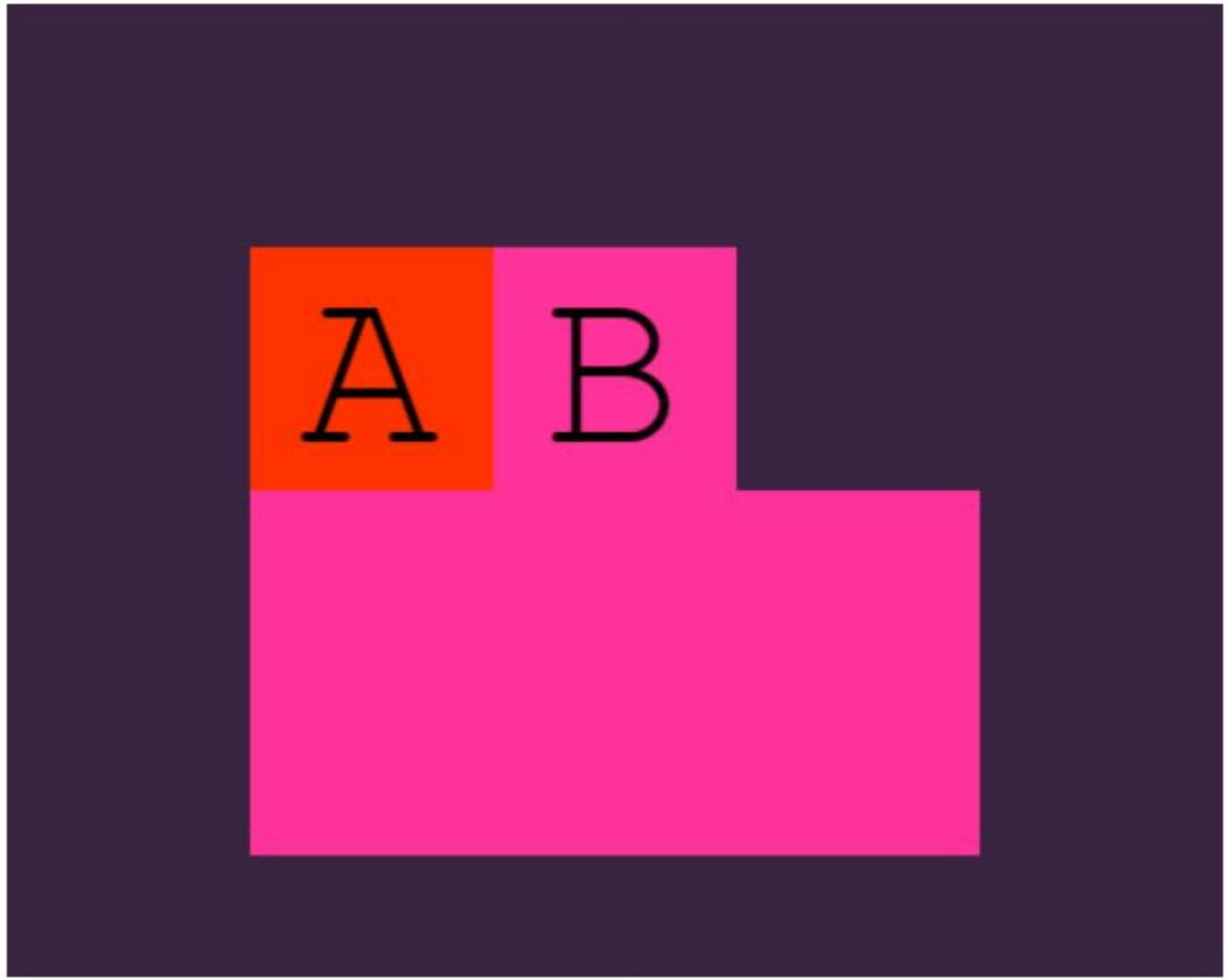
```
1   class escena2 extends Phaser.Scene{
2       constructor(){
3           super({key:"escena2", active: true});
4       }
5
```

Juego2 > Scripts > init.js > ...

```
1   const config = {
2       width:  500,
3       height: 400,
4       parent: "container",
5       type:   Phaser.AUTO,
6       backgroundColor:    '#392542',
7       scene: [escena1, escena2]
8   }
9
10  new Phaser.Game(config);
```

Hay que hacer dos cambios para la escena 2 aparezca en pantalla.

- En *escena2.js* en el constructor agregar *active=true*
- En el *init.js* vamos a agregar la escena2 en el array de scenes



Si salvamos y vamos al browser mostrará ahora dos letras con sus espacios. Para el tab C podemos hacer lo mismo que con B o llamarlo desde alguno de los dos escenas visibles. Es lo que haremos a continuación.

2.7 - Escena 3 desde Escena 2

Juego2 > Scripts > Scenes > JS escena2.js > ...

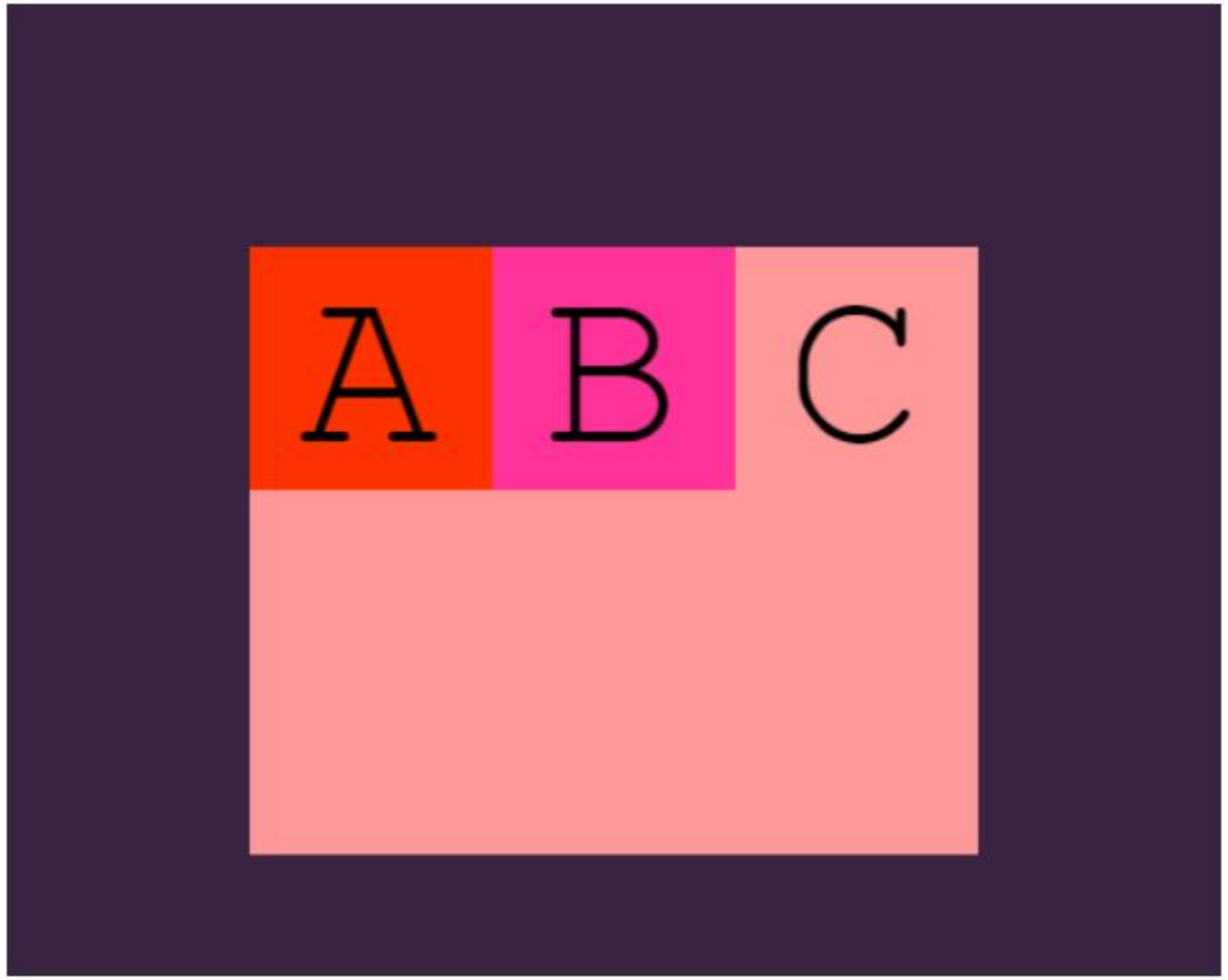
```
1   class escena2 extends Phaser.Scene{
2       constructor(){
3           super({key:"escena2", active: true});
4       }
5
6       preload(){}
7
8       create(){
9           let graphics = this.add.graphics();
10          graphics.fillStyle(0xff3399,1);
11          graphics.fillRect(100,200,300,150);
12          graphics.fillRect(200,100,100,100);
13          this.add.text(220,110,"B", {font: "96px Courier", fill: "#000"});
14
15          //Hacemos la carga de escena 3 al manejador de escena fuera del init.js
16          this.scene.add("escena3", new escena3);
17      }
18
19      update(time, delta){}
20  }
```

Lo único que debemos hacer es agregar esta línea donde deseamos hacer el llamado de la siguiente escena.

```
this.scene.add("escena3", new escena3);
```

Opciones para jugar con el llamado de escenas

```
this.scene.bringToTop(); *Encima de todo*
this.scene.sendToBack(); *Atrás de todo*
this.scene.moveUp(); *Un paso arriba*
this.scene.moveDown(); *Un paso atrás*
this.scene.moveAbove(); *Encima de una escen
this.scene.moveBelow(); *Abajo de una escena
```



Tras salvar vemos como queda nuestro ejemplo.

Tarea:

- Crear un nuevo tab, usar una de las opciones para llamarlo ex. *MoveUp*

Carga de escenas por módulo

3.1 - El Index

Juego3 > index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Phaser, cambio de escenas</title>
    </head>
    <body>
        <script src="./node_modules/phaser/dist/phaser.js"></script>

        <script src="./Scripts/init.js" type="module"></script>
    </body>
</html>
```

Quitando el llamado de las escenas desde el index, luego agregamos el type dentro del llamado del init.

3.2 - Hacer clases exportables

escena1.js

Juego3 > Scripts > Scenes > escena1.js > escena1

```
1  export default class escena1 extends Phaser.Scene{
2      constructor(){
3          super({key:"escena1"});
4      }
5
6      preload(){}
7
8      create(){
9          let graphics = this.add.graphics();
10         graphics.fillStyle(0xff3300,1);
11         graphics.fillRect(100,200,300,150);
12         graphics.fillRect(100,100,100,100);
13
14         this.add.text(120,110,"A", {font: "96px Courier", fill
15     }
16
17     update(time, delta){}
18 }
```

escena2.js

Juego3 > Scripts > Scenes > escena2.js > escena2 > constructor

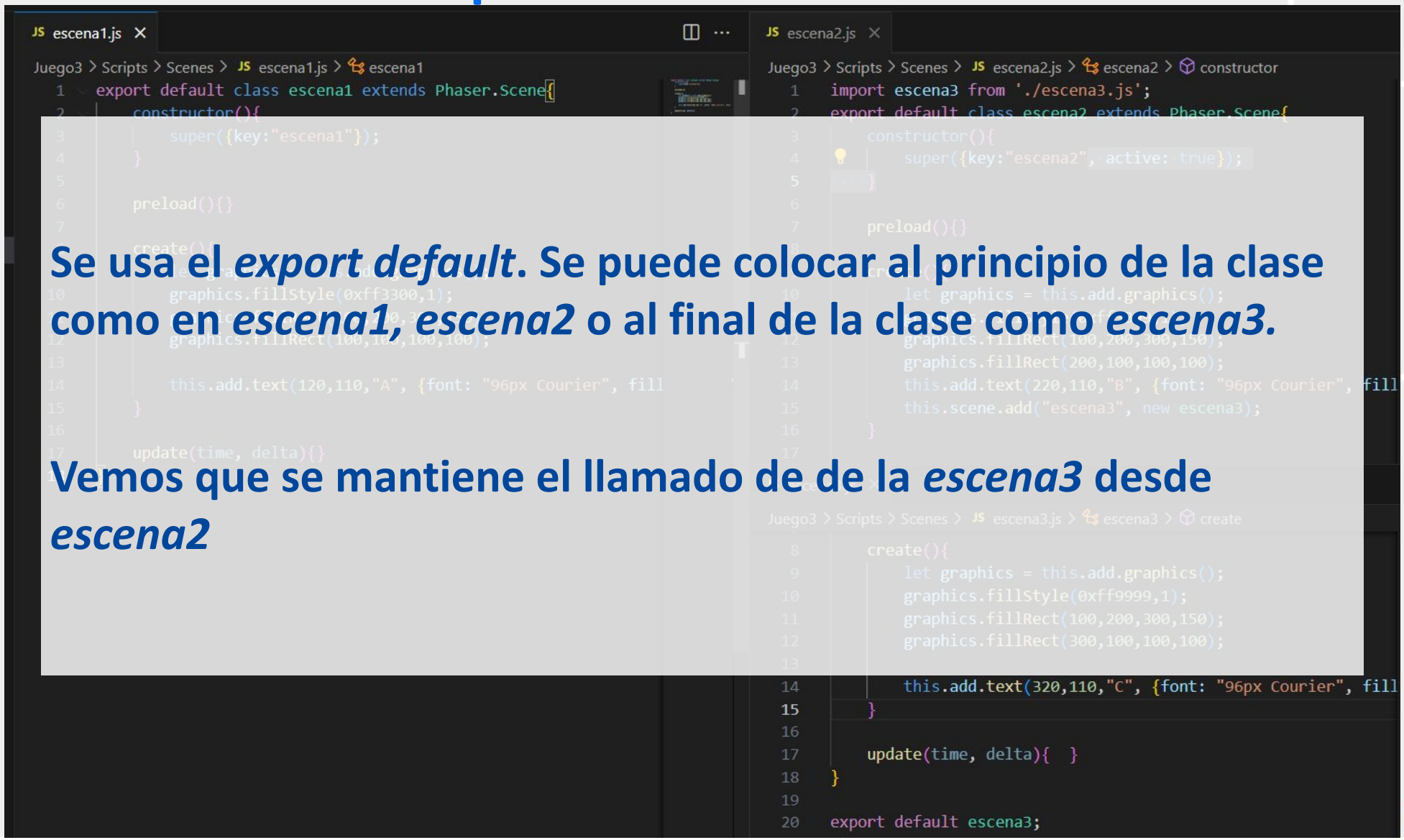
```
1  import escena3 from './escena3.js';
2  export default class escena2 extends Phaser.Scene{
3      constructor(){
4          super({key:"escena2", active: true});
5      }
6
7      preload(){}
8
9      create(){
10         let graphics = this.add.graphics();
11         graphics.fillStyle(0xff3399,1);
12         graphics.fillRect(100,200,300,150);
13         graphics.fillRect(200,100,100,100);
14         this.add.text(220,110,"B", {font: "96px Courier", fill
15         this.scene.add("escena3", new escena3);
16     }
17
```

escena3.js

Juego3 > Scripts > Scenes > escena3.js > escena3 > create

```
8      create(){
9          let graphics = this.add.graphics();
10         graphics.fillStyle(0xff9999,1);
11         graphics.fillRect(100,200,300,150);
12         graphics.fillRect(300,100,100,100);
13
14         this.add.text(320,110,"C", {font: "96px Courier", fill
15     }
16
17     update(time, delta){  }
18 }
19
20 export default escena3;
```

3.2 - Hacer clases exportables



Se usa el *export default*. Se puede colocar al principio de la clase como en *escena1, escena2* o al final de la clase como *escena3.*

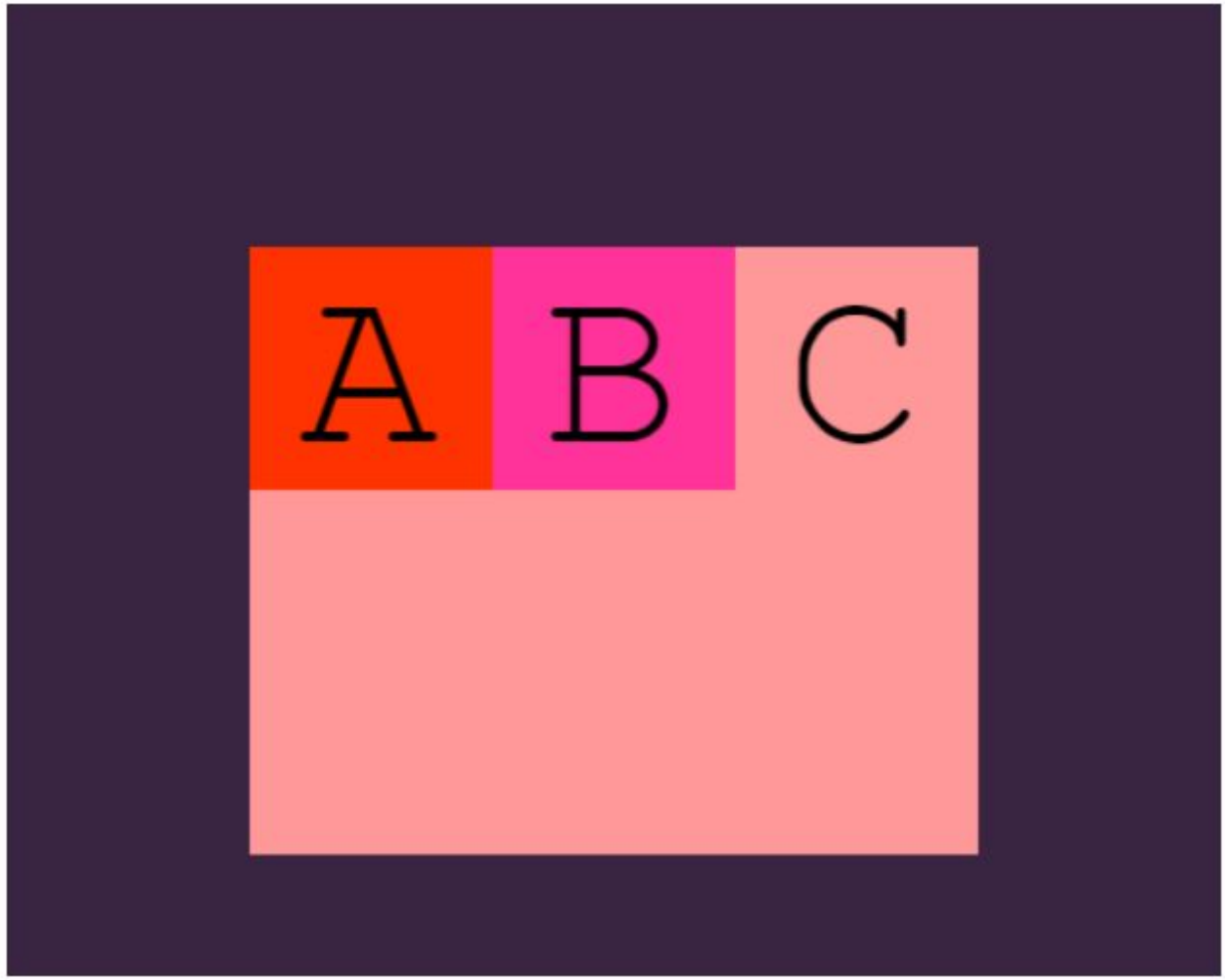
Vemos que se mantiene el llamado de de la *escena3* desde *escena2*

3.3 - Importar Escenas en el init

Juego3 > Scripts > JS init.js > ...

```
1   import escena1 from './Scenes/escena1.js';
2   import escena2 from './Scenes/escena2.js';
3
4   const config = {
5       width:  500,
6       height: 400,
7       parent: "container",
8       type:   Phaser.AUTO,
9       backgroundColor:    '#392542',
10      scene: [escena1, escena2]
11  }
12
13  new Phaser.Game(config);
```

Hacemos el llamado de las escenas en la parte superior con el *import*



Tras salvar vemos como queda nuestro ejemplo.

Tarea:

- Crear un nuevo tab, usar una de las opciones para llamarlo ex. *MoveDown*