



Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

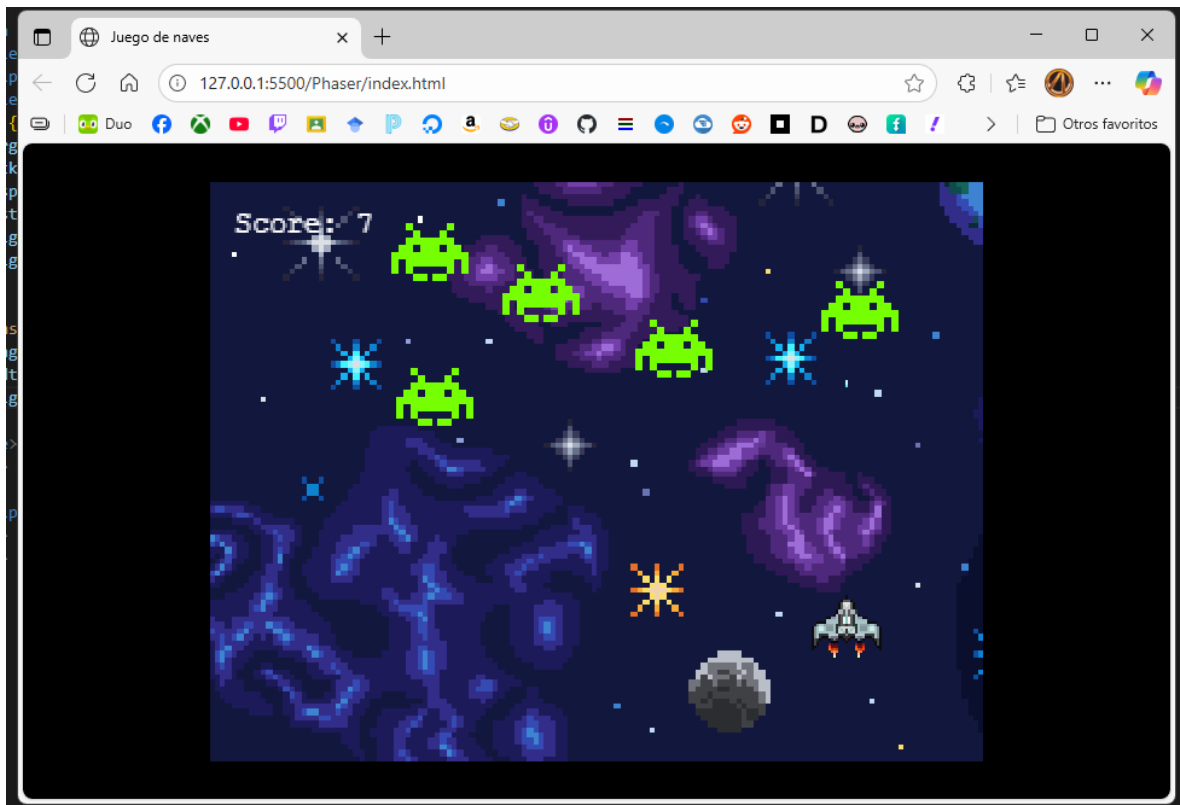
Programación para internet – Ciclo 2025 B

Reporte de Tarea 2. Juego con phaser

Profesor: López Franco Michel Emanuel

Bautista Chavira Alan Gabriel – 218539152

26/agosto/2025



La funcionalidad del juego es muy sencilla, simplemente el cursor mueve una nave y esta dispara con click izquierdo. Al disparar se crea un proyectil que avanza hacia arriba hasta dar con un enemigo, si el enemigo y el proyectil se solapan el enemigo se reposiciona para dar la ilusión de que fue destruido y apareció otro y se añade un punto al contador.

Para poder probarlo en local sin subirlo a un servidor web real tuve que bajar una extensión de VS Code llamada Live server, porque mi navegador por defecto no me deja consultar recursos almacenados de forma local por seguridad pero debería funcionar perfectamente desplegado en una página real siempre y cuando los assets vengan incluidos de la misma forma que están en el repositorio

En cuanto al código la lógica del juego es muy sencilla

```
for (let i = 0; i < 5; i++) {  
  const enemy = enemies.create(  
    Phaser.Math.Between(50, 270),  
    Phaser.Math.Between(20, 100),  
    'enemy'  
  );  
  enemy.setScale(0.03);  
}  
  
this.physics.add.overlap(bullets, enemies, (bullet, enemy) => {  
  bullet.destroy();  
  enemy.setPosition(  
    Phaser.Math.Between(50, 270),  
    Phaser.Math.Between(20, 100)  
  );  
  
  score += 1;  
  scoreText.setText('Score: ' + score);  
});  
}
```

Estas son las funciones que ponen los enemigos en pantalla, detectan cuando son impactados y actualiza el contador de puntuación (esto está en el create)

Suena contraintuitivo actualizar la puntuación en el create en vez del update, pero score esta declarado de forma que add overlap añade un registro de un objeto que maneja las físicas y al hacer esto dicho objeto se actualiza constantemente. No es necesario hacer esas comprobaciones en el Update, el phaser las maneja solito.

```

function update(time) {
    player.setVelocity(0);

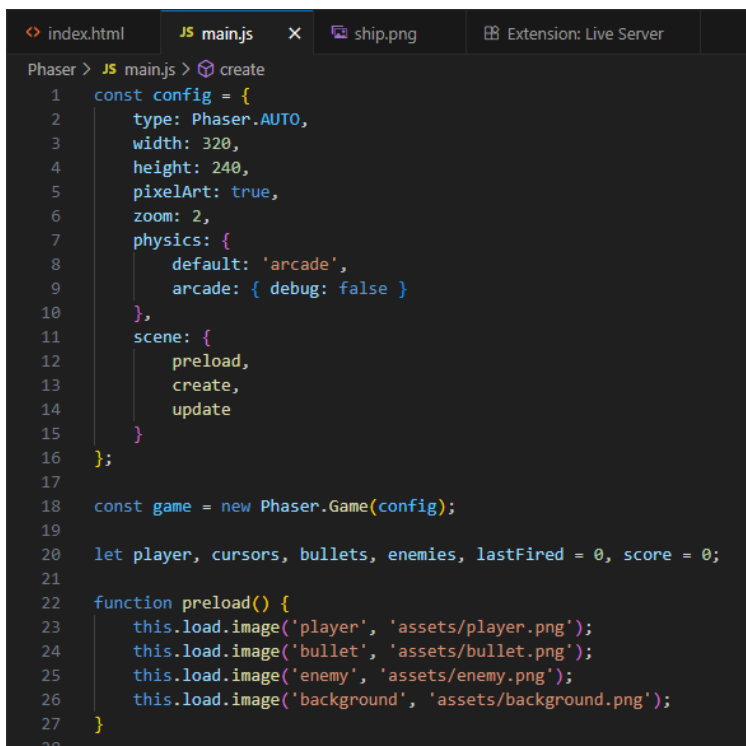
    player.x = this.input.activePointer.x;
    player.y = this.input.activePointer.y;

    if (this.input.activePointer.isDown && time > lastFired) {
        const bullet = bullets.get(player.x, player.y - 10, 'bullet');
        if (bullet) {
            bullet.setActive(true).setVisible(true);
            bullet.setScale(0.02);
            bullet.body.velocity.y = -200;
            lastFired = time + 300;
        }
    }
}

```

Usamos el Update para estar leyendo constantemente la posición del mouse y mover la nave así como saber cuando se puede disparar y cuando no (también me dio flojera ajustar el tamaño en pixeles de los sprites así que solo los reescale de forma bruta)

Lo demás es meramente para inicializar cosas o cargar recursos



```

index.html  JS main.js  ship.png  Extension: Live Server
Phaser > JS main.js > create
1  const config = {
2      type: Phaser.AUTO,
3      width: 320,
4      height: 240,
5      pixelArt: true,
6      zoom: 2,
7      physics: {
8          default: 'arcade',
9          arcade: { debug: false }
10     },
11     scene: {
12         preload,
13         create,
14         update
15     }
16 };
17
18 const game = new Phaser.Game(config);
19
20 let player, cursors, bullets, enemies, lastFired = 0, score = 0;
21
22 function preload() {
23     this.load.image('player', 'assets/player.png');
24     this.load.image('bullet', 'assets/bullet.png');
25     this.load.image('enemy', 'assets/enemy.png');
26     this.load.image('background', 'assets/background.png');
27 }
28

```