https://youtu.be/pbjVYCFbl0U

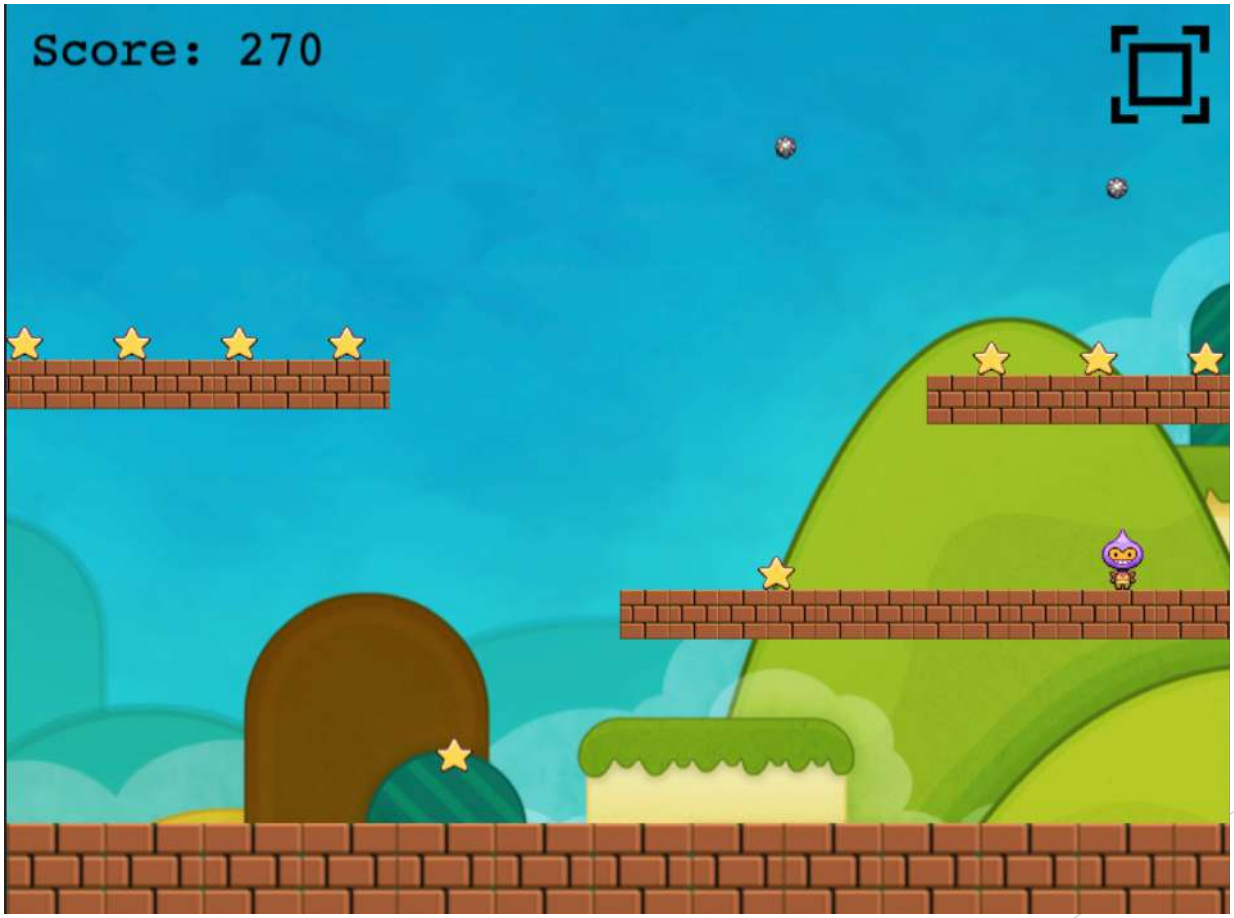
MAJ : 09/24

Introduction : Framework

Phaser c’est quoi ?

- framework JS de jeu HTML5 multi-navigateurs
- Documentation : https://photonstorm.github.io/phaser3-docs/
- La seule exigence du navigateur est la prise en charge de la balise canvas
- Nouveauté 2024 -> Editeur payant : https://phaser.io/editor
- Utiliser un serveur web pour faire fonctionner votre code même si cela ne vous semble pas obligatoire. Pourquoi ?
  - Votre jeu va devoir charger des ressources : images, fichiers audio, données JSON, peut-être d'autres fichiers JavaScript. Et pour ce faire, il doit fonctionner sans être gêné par les chaînes de sécurité du navigateur. Il a donc besoin d’un serveur web.

Démo



Etape 1 : corps du code

- Fichier HTML passe-partout avec un script JS :

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8" />
    <title>Demo</title>
    <script src="//cdn.jsdelivr.net/npm/phaser@3.80.1/dist/phaser.js"></script>
    <style type="text/css">
        body {
            margin: 0;
        }
    </style>
</head>
<body>
```

Etape 1 : config et jeu

```
var config = {
    type: Phaser.AUTO,
    width: 800,
    height: 600,
    scene: {
        preload: preload,
        create: create,
        update: update
    }
};

var game = new Phaser.Game(config);

function preload ()
{
}

function create ()
{
}

function update ()
{
}
```

- **config :**

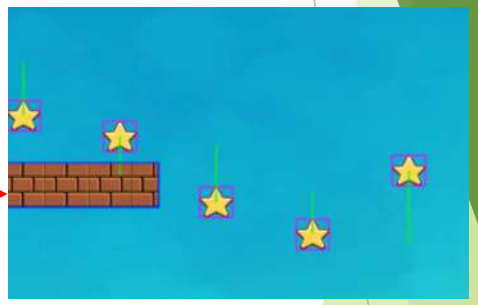
PHASER.AUTO choisit automatiquement le moteur de rendu

Width et height : largeur, hauteur de la fenetre

D'autres options sont possibles comme scale ou physics

```
physics: {
    default: 'arcade',
    arcade: {
        gravity: { y: 300 },
        debug: false
    }
},
```

true



Permet d'instancier un nouveau jeu.

On pourra alors utiliser preload(), create() et update()

Surcharge update : update(time,delta)

time : temps écoulé depuis le lancement

delta : temps écoulé entre 2 frames (16,6 ms par défaut)

Etape 2 : chargement des ressources

- On va charger les ressources dont on a besoin pour notre jeu
- Rq : il est possible d'utiliser setOrigin pour modifier l'origine d'une image (centrée par défaut)

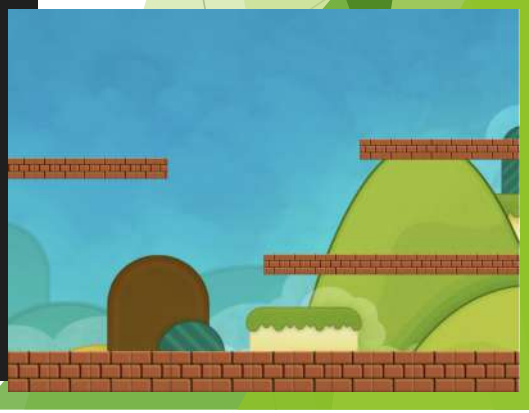
```
function preload ()
{
    this.load.image('sky', 'assets/sky2.png');
    this.load.image('ground', 'assets/platform.png');
    this.load.image('star', 'assets/star.png');
    this.load.image('bomb', 'assets/bomb.png');
    this.load.spritesheet('dude', 'assets/dude.png', { frameWidth: 32, frameHeight: 48 });
}
```

Etape 3 : mise en place du monde

- On ajoute le fond “sky”
- On crée les plateformes. Pour anticiper les collisions on applique le système physique le plus simple : Arcade physic (a définir dans le config). Les plateformes font partie d’un groupe statique (car elles ne bougent pas)

```
var platforms;

function create ()
{
    this.add.image(400, 300, 'sky');

    platforms = this.physics.add.staticGroup();

    platforms.create(400, 568, 'ground').setScale(2).refreshBody();

    platforms.create(600, 400, 'ground');
    platforms.create(50, 250, 'ground');
    platforms.create(750, 220, 'ground');
}
```

```
physics: {
    default: 'arcade',
    arcade: {
        gravity: { y: 300 },
        debug: false
    }
},
```



Etape 3 : mise en place du monde - physique

- Arcade Physics :
  - Dans Arcade Physics, il existe deux types de corps physiques : Dynamique et Statique. Un corps dynamique est un corps qui peut se déplacer via des forces telles que la vitesse ou l'accélération. Il peut rebondir et entrer en collision avec d'autres objets et cette collision est influencée par la masse du corps et d'autres éléments.
  - À l'opposé, un corps statique a simplement une position et une taille. Il n'est pas touché par la gravité, on ne pas régler la vitesse et quand quelque chose entre en collision avec lui, il ne bouge jamais. Statique de nom, statique de nature. Et parfait pour le sol et les plates-formes.

Etape 4 : création du joueur

- On précise sa physique dynamique (ici valeur de rebond et collision sur les bordures) :

```
player = this.physics.add.sprite(100, 450, 'dude');

player.setBounce(0.2);
player.setCollideWorldBounds(true);
```

- On précise son animation (SpriteSheet) :

  (repeat à -1 : boucle à l'infini)

```
this.anims.create({
    key: 'left',
    frames: this.anims.generateFrameNumbers('dude', { start: 0, end: 3 }),
    frameRate: 10,
    repeat: -1
});
```

Rq : le gestionnaire d'animations est un système global. Les animations qui y sont créées sont globalement disponibles pour tous les objets de jeu

Etape 4 : création du joueur

- On peut appliquer une gravité au joueur (propriété body).

Plus elle est élevée plus le sprite est lourd

- Nous devons gérer les collisions à l'aide d'un collider.

```
player.body.setGravityY(300);
this.physics.add.collider(player, platforms);
```

Etape 5 : gestion du clavier

- Phaser a un gestionnaire de clavier intégré

```
cursors = this.input.keyboard.createCursorKeys();
```

On peut alors accéder à des propriétés pour gérer le déplacement du joueur.

Attention, pour gérer le saut il faut aussi tester que le joueur est bien au sol.

```
if (cursors.up.isDown && player.body.touching.down)
{
    player.setVelocityY(-330);
}
```

```
if (cursors.left.isDown)
{
    player.setVelocityX(-160);

    player.anims.play('left', true);
}
else if (cursors.right.isDown)
{
    player.setVelocityX(160);

    player.anims.play('right', true);
}
else
{
    player.setVelocityX(0);

    player.anims.play('turn');
}
```

Il faut placer ces tests dans la boucle update qui est appelée 60 fois par seconde.

Etape 6 : mise en place du jeu

- On va maintenant mettre en place notre gameplay :
  1. Ajouter des étoiles à récupérer
  2. Afficher le score
  3. Ajouter des bombes à éviter
  4. Ajouter des bruitages
  5. …

Etape 6 : mise en place du jeu

- On va maintenant mettre en place notre gameplay :
  1. **Ajouter des étoiles à récupérer**

```
stars = this.physics.add.group({
    key: 'star',
    repeat: 11,
    setXY: { x: 12, y: 0, stepX: 70 }
});

stars.children.iterate(function (child) {

    child.setBounceY(Phaser.Math.FloatBetween(0.4, 0.8));

});
```

Ajout de 12 étoiles espacées de 70 pixels

Modification du rebond de chaque étoile entre 0,4 et 0,8

```
this.physics.add.overlap(player, stars, collectStar, null, this);
```

Détection de collision :

```
function collectStar (player, star)
{
    star.disableBody(true, true);
}
```

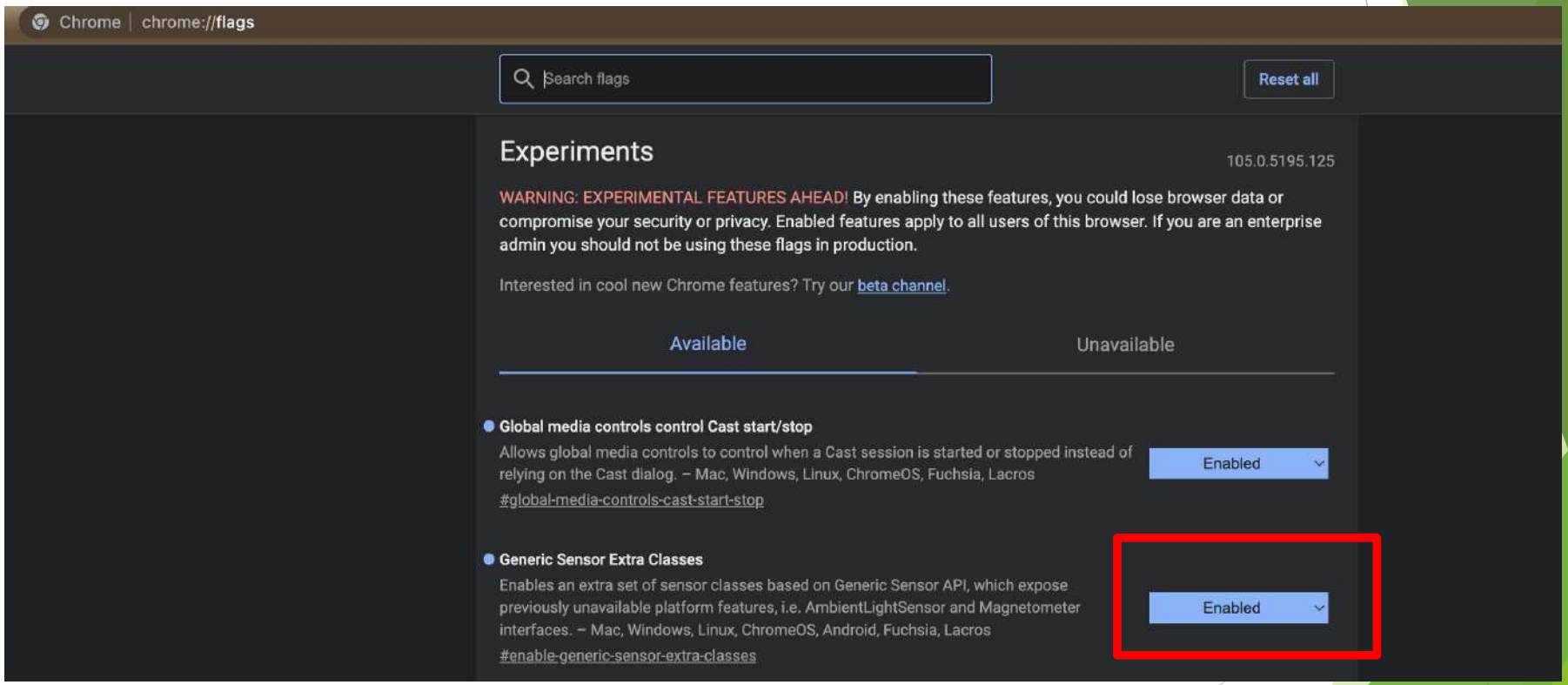
Final

Etape 6 : mise en place du jeu

- On va maintenant mettre en place notre gameplay :
  1. Ajouter des étoiles à récupérer
  2. Afficher le score
  3. Ajouter des bombes à éviter
  4. Ajouter des bruitages

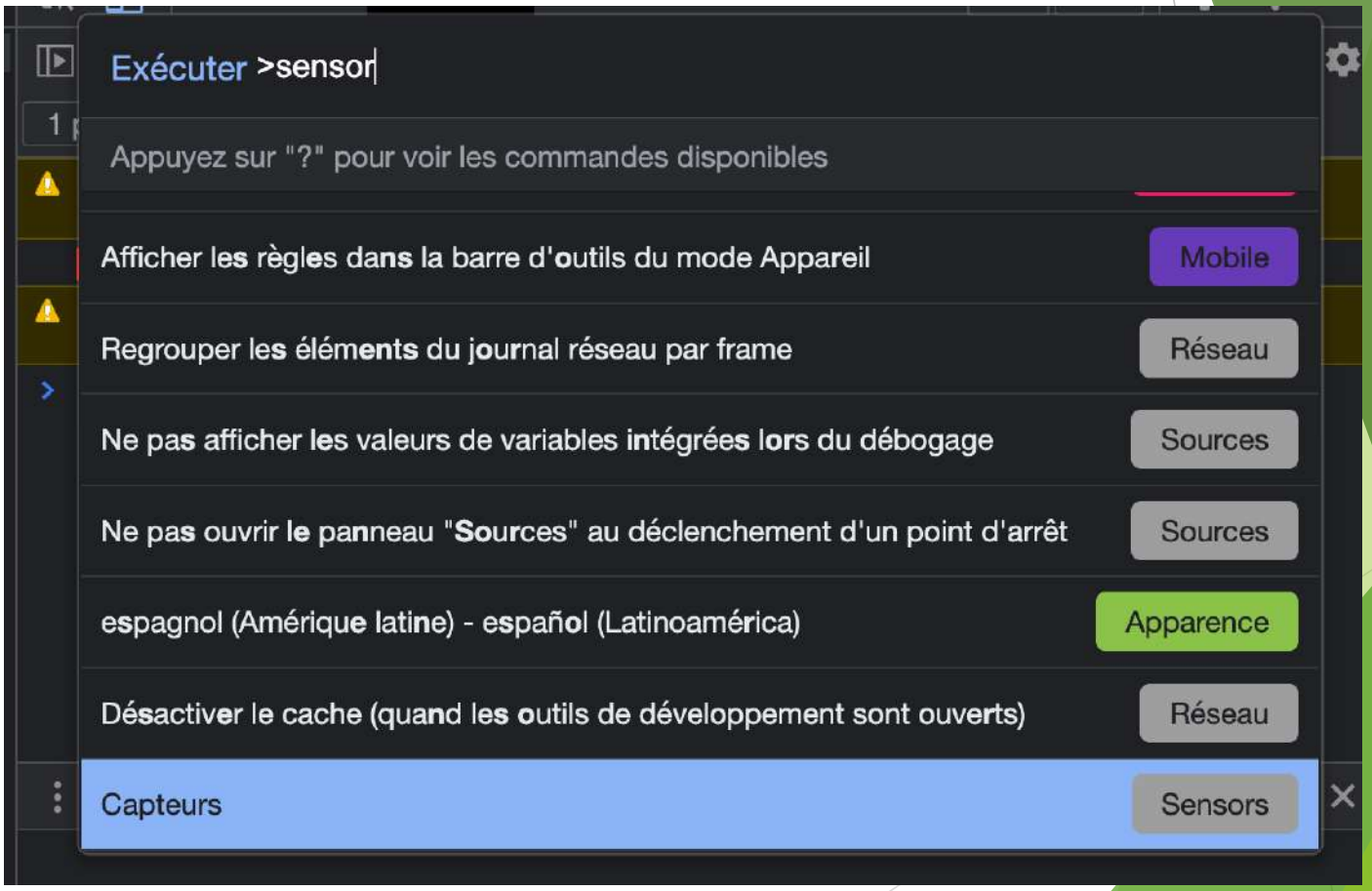
  … La suite en TD

Tests : Sensor Smartphone Chrome

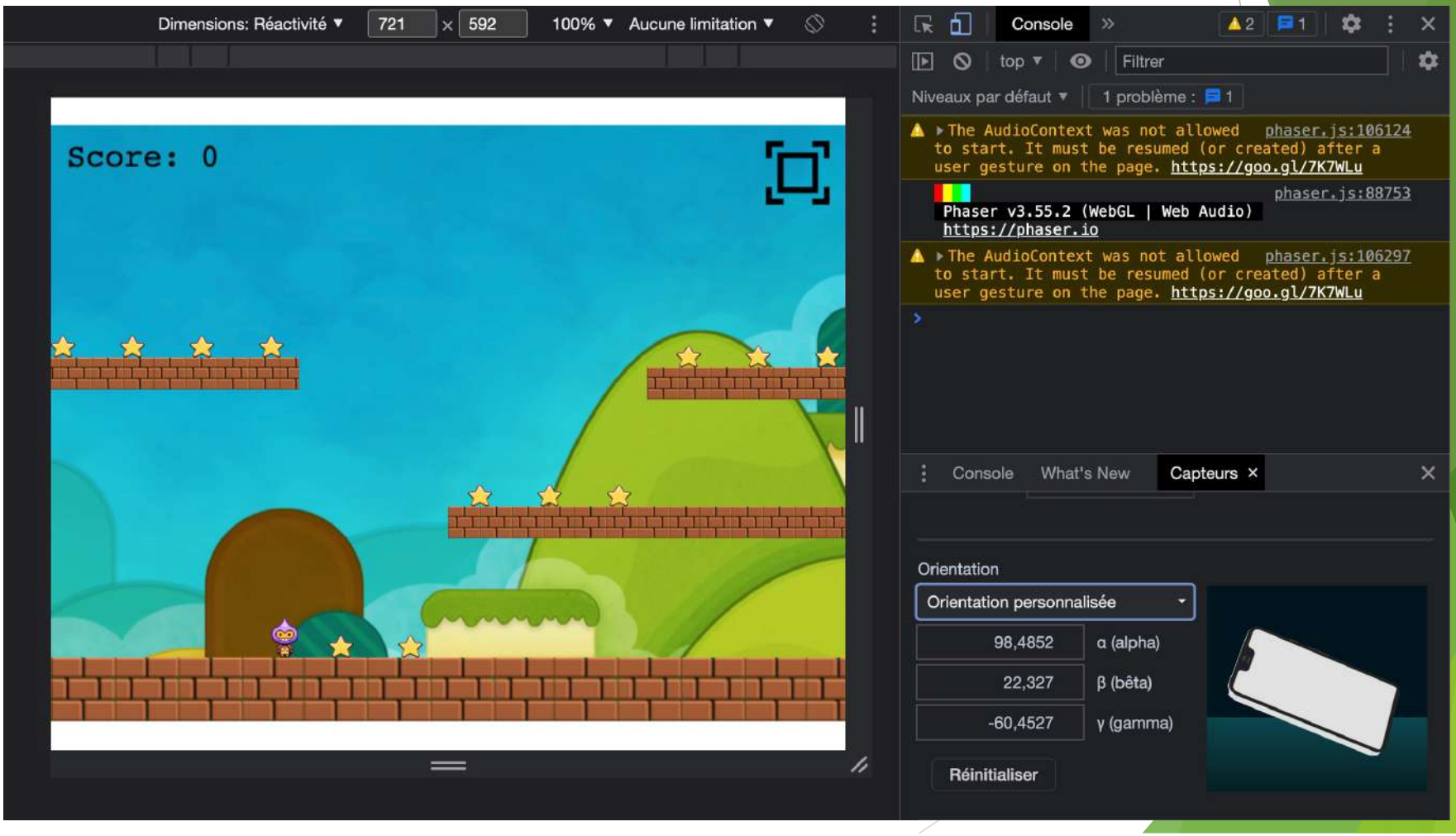


Tests : Sensor Smartphone Chrome

CTRL+SHIFT+P
Ou
CMD+SHIFT+P



Tests : Sensor Smartphone Chrome



Déploiement sur Smartphone

- Certificat SSL (HTTPS) obligatoire pour la gestion des capteurs

https://w3c.github.io/deviceorientation/#security-and-privacy