LPB2025 : Projet GameJam 2025

Cours : LPB2025
Projet : GameJam 2025
Date : 6 mai 2025
Objectif : Créer un jeu vidéo 2D en JavaScript

Introduction

Dans le cadre du cours **LPB2025**, le projet **GameJam 2025** vous invite à créer un jeu vidéo 2D en JavaScript. Vous contrôlerez un carré bleu à l'aide des flèches du clavier pour attraper des carrés rouges qui tombent. Chaque carré attrapé ajoute 10 points à votre score. Si un carré rouge atteint le bas de l'écran, le jeu s'arrête avec un écran de *Game Over*. Vous pouvez alors recommencer en appuyant sur la touche `R`.

Ce projet est une occasion amusante d'apprendre des concepts clés de programmation tout en créant un jeu que vous pouvez personnaliser !

Objectifs pédagogiques

- Comprendre comment utiliser le `<canvas>` HTML5 pour dessiner des graphismes.
- Gérer les entrées clavier pour contrôler un personnage.
- Implémenter une boucle de jeu pour une animation fluide.
- Détecter des collisions entre objets.
- Afficher un score et gérer l'état du jeu (*Game Over*, redémarrage).

Technologies utilisées

- **HTML** : Structure de la page avec un `<canvas>` pour le jeu.
- **CSS** : Centrage du canvas et du titre avec Flexbox.
- **JavaScript** : Logique du jeu, animations, et interactions.

Concepts techniques

Le canvas HTML5

Le `<canvas>` est un élément HTML qui permet de dessiner des graphismes 2D avec JavaScript. Dans **GameJam 2025**, nous utilisons le canvas pour :

- Dessiner le joueur (un carré bleu).
- Afficher les obstacles (carrés rouges).
- Montrer le score et l'écran de *Game Over*.

Le contexte `2d` du canvas (accessible via `ctx`) fournit des méthodes comme `fillRect()` pour dessiner des rectangles et `fillText()` pour afficher du texte.

La boucle de jeu

Une boucle de jeu est une fonction qui s'exécute environ 60 fois par seconde pour mettre à jour et dessiner le jeu. Nous utilisons `requestAnimationFrame` pour une animation fluide. La boucle :

1. Efface le canvas.
2. Met à jour la position du joueur et des obstacles.
3. Vérifie les collisions.
4. Dessine tous les éléments.

Gestion des entrées clavier

Les événements `keydown` et `keyup` détectent les touches pressées ou relâchées. Dans ce jeu :

- Les flèches gauche et droite déplacent le joueur.
- La touche `R` relance le jeu après un *Game Over*.

Un objet `keys` stocke l'état des touches pour des mouvements fluides.

Détection de collisions

Une collision se produit lorsque deux rectangles (le joueur et un obstacle) se chevauchent. Nous utilisons une condition mathématique pour comparer leurs positions et tailles. Dans **GameJam 2025** :

- Une collision avec un carré rouge augmente le score et supprime l'obstacle.
- Si un carré rouge sort de l'écran, le jeu passe en mode *Game Over*.

Gestion de l'état du jeu

Une variable `gameOver` indique si le jeu est terminé. Quand `gameOver` est `true` :

- Le joueur ne peut plus bouger.
- Aucun nouvel obstacle n'est créé.
- Un écran *Game Over* s'affiche avec le score final.

La fonction `resetGame()` réinitialise le jeu pour une nouvelle partie.

Structure du code

Le fichier HTML (`index.html`)

Le fichier HTML définit la structure de la page, avec un titre et un canvas centrés à l'aide de Flexbox.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>GameJam 2025</title>
    <style>
        body {
            margin: 0;
```

```
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            background-color: #f0f0f0;
        }
        .game-container {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        h1 {
            font-family: Arial, sans-serif;
            font-size: 36px;
            color: #333;
            margin-bottom: 20px;
            text-align: center;
        }
        canvas {
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <div class="game-container">
        <h1>GameJam 2025</h1>
        <canvas id="gameCanvas" width="800" height="600"></canvas>
    </div>
    <script src="game.js"></script>
</body>
</html>
```

Le fichier JavaScript (`game.js`)

Le fichier JavaScript contient la logique du jeu. Voici le code complet, avec des commentaires pour chaque section.

```
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');

// Joueur
const player = {
    x: 50,
    y: canvas.height - 50,
    width: 50,
    height: 50,
    speed: 5
};

// Variables du jeu
let obstacles = [];
let score = 0;
```

```
let keys = {};
let gameOver = false;

// Gestion des événements clavier
document.addEventListener('keydown', (e) => {
    e.preventDefault();
    keys[e.key] = true;
    // Recommencer avec la touche 'r' ou 'R'
    if ((e.key.toLowerCase() === 'r') && gameOver) {
        resetGame();
    }
});

document.addEventListener('keyup', (e) => {
    keys[e.key] = false;
});

// Réinitialiser le jeu
function resetGame() {
    player.x = 50;
    obstacles = [];
    score = 0;
    gameOver = false;
    ctx.textAlign = 'start';
}

// Déplacer le joueur
function updatePlayer() {
    if (!gameOver) {
        if (keys['ArrowLeft'] && player.x > 0) {
            player.x -= player.speed;
        }
        if (keys['ArrowRight'] && player.x < canvas.width - player.width) {
            player.x += player.speed;
        }
    }
}

// Dessiner le joueur
function drawPlayer() {
    ctx.fillStyle = 'blue';
    ctx.fillRect(player.x, player.y, player.width, player.height);
}

// Créer un obstacle (carré rouge)
function createObstacle() {
    if (!gameOver) {
        const obstacle = {
            x: Math.random() * (canvas.width - 50),
            y: 0,
            width: 50,
            height: 50,
            speed: 3
```

```
        };
        obstacles.push(obstacle);
    }
}

// Mettre à jour les obstacles
function updateObstacles() {
    for (let i = 0; i < obstacles.length; i++) {
        obstacles[i].y += obstacles[i].speed;
        if (obstacles[i].y > canvas.height) {
            gameOver = true;
            obstacles.splice(i, 1);
            i--;
        }
    }
}

// Dessiner les obstacles
function drawObstacles() {
    ctx.fillStyle = 'red';
    for (let obstacle of obstacles) {
        ctx.fillRect(obstacle.x, obstacle.y, obstacle.width, obstacle.height);
    }
}

// Vérifier les collisions
function checkCollisions() {
    for (let i = 0; i < obstacles.length; i++) {
        if (
            player.x < obstacles[i].x + obstacles[i].width &&
            player.x + player.width > obstacles[i].x &&
            player.y < obstacles[i].y + obstacles[i].height &&
            player.y + player.height > obstacles[i].y
        ) {
            score += 10;
            obstacles.splice(i, 1);
            i--;
        }
    }
}

// Afficher le score et l'écran de game over
function drawUI() {
    ctx.fillStyle = 'black';
    ctx.font = '20px Arial';
    ctx.textAlign = 'start';
    ctx.fillText('Score: ' + score, 10, 20);

    if (gameOver) {
        ctx.fillStyle = 'rgba(0, 0, 0, 0.7)';
        ctx.fillRect(0, 0, canvas.width, canvas.height);
        ctx.fillStyle = 'white';
        ctx.font = '40px Arial';
```

```
        ctx.textAlign = 'center';
        ctx.fillText('Game Over!', canvas.width / 2, canvas.height / 2);
        ctx.font = '20px Arial';
        ctx.fillText('Score final: ' + score, canvas.width / 2, canvas.height / 2 + 40);
        ctx.fillText('Appuyez sur R pour recommencer', canvas.width / 2, canvas.height / 2 +
80);
    }
}

// Créer un nouvel obstacle toutes les 2 secondes
setInterval(createObstacle, 2000);

// Boucle de jeu
function gameLoop() {
    if (!gameOver) {
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        updatePlayer();
        drawPlayer();
        updateObstacles();
        drawObstacles();
        checkCollisions();
    }
    drawUI();
    requestAnimationFrame(gameLoop);
}

// Lancer le jeu
gameLoop();
```

Comment jouer

1. Ouvrez `index.html` dans un navigateur.
2. Utilisez les flèches gauche et droite pour déplacer le carré bleu.
3. Attrapez les carrés rouges pour augmenter votre score.
4. Si un carré rouge atteint le bas de l'écran, le jeu s'arrête.
5. Appuyez sur `R` pour recommencer une nouvelle partie.

Personnalisations possibles

Voici quelques idées pour améliorer **GameJam 2025** :

- **Changer les graphismes** : Utilisez `ctx.drawImage()` pour remplacer les carrés par des images (par exemple, un personnage ou des objets).
- **Ajouter des sons** : Intégrez des effets sonores avec `new Audio('son.mp3').play()` lors des collisions.
- **Augmenter la difficulté** : Accélérez les obstacles au fil du temps ou réduisez l'intervalle entre leur création.
- **Ajouter une pause** : Implémentez une touche `P` pour mettre le jeu en pause.
- **Modifier l'interface** : Changez la couleur du score ou ajoutez une animation pour les collisions.

Ressources pour aller plus loin

- **MDN Web Docs** : Documentation sur `<canvas>` et JavaScript (https://developer.mozilla.org).
- **Phaser.js** : Bibliothèque pour créer des jeux 2D plus complexes (https://phaser.io).
- **Tutoriels vidéo** : Cherchez "JavaScript game development" sur YouTube pour des guides pratiques.
- **CodePen** : Testez des idées de jeux en ligne (https://codepen.io).

Conclusion

Avec **GameJam 2025**, vous avez créé un jeu 2D fonctionnel en JavaScript dans le cadre du cours **LPB2025**. Vous avez appris à utiliser le canvas, gérer les entrées clavier, créer une boucle de jeu, détecter des collisions, et afficher une interface utilisateur. Continuez à expérimenter avec ce code pour ajouter vos propres idées et créer des jeux encore plus impressionnants ! Bravo pour votre travail !