Overview

This is a simple JavaScript-based space game where the player controls a spaceship to avoid obstacles while trying to stay airborne. The game increases in difficulty as the player’s score increases.

Features

- **Gameplay**: The player uses the arrow keys to control the spaceship's vertical movement.
- **Obstacles**: Randomly generated obstacles appear on the screen, and the player must avoid them.
- **Collision Detection**: The game detects collisions between the player’s spaceship and obstacles, triggering a sound and ending the game.
- **Score Tracking**: The score increases over time based on survival duration.
- **Speed Increase**: As the score increases, the speed of the obstacles gradually increases, enhancing the game’s challenge.

JavaScript Functionality

- **Game Area Setup**: Creates the game canvas and initializes game variables.
- **Game Piece and Obstacles**: Defines the player's spaceship and the obstacles that appear.
- **Event Listeners**: Listens for button clicks to start and pause the game, and key presses for player controls.
- **Game Loop**: Continuously updates the game state, including the player's position, obstacle positions, and score.
- **Collision Detection**: Checks if the player collides with any obstacles, plays a sound, and ends the game if a collision occurs.

Key Functions

- **startGame()**: Initializes game elements and starts the game loop.
- **updateGameArea()**: Main game loop that updates positions and checks for collisions.
- **accelerate(n)**: Controls the spaceship's movement based on key presses.
- **everyinterval(n)**: Utility function to check if a specific number of frames has passed.

Reflection

It was a lot of fun working on this game and learning about the technical applications of functions I had heard of but never used. I will continue to workshop this game as a basic entry point into game development because its simplicity makes it the perfect place to start. I had some trouble adding a few enhancements, things like the game speeding up as it progressed ended up breaking the game over and over again until I gave in.

Game Area

```
const myGameArea = {

    canvas: document.createElement("canvas"),

    start: function() {

        this.canvas.width = 480;

        this.canvas.height = 270;

        document.body.appendChild(this.canvas);

        this.context = this.canvas.getContext("2d");

        this.frameNo = 0;

    },

    clear: function() {

        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);

    }

};
```

Purpose: Initializes the game canvas where the game will be drawn.

Key Functions:

- `start()`: Sets the canvas dimensions, appends it to the document, and initializes the drawing context.
- `clear()`: Clears the canvas for redrawing in each game loop.

}

Game Pieces and obstacles

```
function gameObject(width, height, color, x, y, type) {
    this.type = type;
    this.width = width;
    this.height = height;
    this.x = x;
    this.y = y;
    // ...
    this.update = function() {
        const ctx = myGameArea.context;
        if (this.type == "text") {
            ctx.font = this.width + " " + this.height;
            ctx.fillStyle = color;
            ctx.fillText(this.text, this.x, this.y);
        } else if (this.image) {
            ctx.drawImage(this.image, this.x, this.y, this.width, this.height);
        } else {
            ctx.fillStyle = color;
            ctx.fillRect(this.x, this.y, this.width, this.height);
        }
    };
    // ...
}
```

Purpose: Defines the properties and behaviors of game objects (the spaceship and obstacles).

Key Features:

- `update()`: Draws the object on the canvas, handling different types (text, image, or rectangle).

Game Loop and Updates

```
function updateGameArea() {

    myGameArea.clear();

    myGameArea.frameNo += 1;

    // ...

    for (let i = 0; i < myObstacles.length; i++) {

        if (myGamePiece.crashWith(myObstacles[i])) {

            collideSound.play();

            alert("You lose! Your score: " + myGameArea.frameNo);

            clearInterval(myGameArea.interval);

            return;

        }

    }

    // ...

}
```

Purpose: The core of the game, continuously updates game states (positions, score, collisions).

Key Features:

- Clears the canvas for redrawing.

- Increases the frame count for scoring.
- Checks for collisions and handles game over conditions.

Collision Detection

```
this.crashWith = function(otherobj) {

    const myleft = this.x;

    const myright = this.x + this.width;

    const mytop = this.y;

    const mybottom = this.y + this.height;

    const otherleft = otherobj.x;

    const otherright = otherobj.x + otherobj.width;

    const othertop = otherobj.y;

    const otherbottom = otherobj.y + otherobj.height;

    return !(mybottom < othertop || mytop > otherbottom || myright < otherleft || myleft > otherright);

};
```

Purpose: Checks if two objects (the spaceship and an obstacle) collide.

Key Features:

- Uses bounding box collision detection, which checks the positions of the edges of the objects to determine overlap.

User Input and Control

```
window.addEventListener('keydown', function(e) {
    switch (e.key) {
        case 'ArrowUp':
            accelerate(-0.05);
            break;
        case 'ArrowDown':
            accelerate(0.05);
            break;
    }
});
```

Purpose: Listens for user key presses to control the spaceship.

Key Features:

- Responds to the up and down arrow keys to adjust the spaceship's vertical speed.

Sounds

```
const jumpSound = new Audio('jump.mp3');
const collideSound = new Audio('Collide.mp3');
collideSound.load();
```

Purpose: Prepares audio files for sound effects in the game.

Key Features:

- Initializes sound objects for actions like jumping and colliding, enhancing player feedback.