Coper – Javascript

Save as `copter.html`
serve with 'python3 -m http.server' → http://localhost:8000/copter.html
Caution: backtick is different from quote!

Base

(could be given a template file as a starting point – url? - copy/paste?)

```
<canvas id="canvas_element" width="320" height="240" style="image-rendering: pixelated;
object-fit: contain; width:100%; max-width: 100%; height: 100%; max-height: 100%;">
</canvas>
<script>
    const canvas = document.getElementById('canvas_element');
    const context = canvas.getContext('2d');

    let paused;
    function start() {paused = false; requestAnimationFrame(timerEvent);}
    function pause() {paused = true;}

    const keys = {
        27: 'ESCAPE',
        32: 'SPACE'
    };
    const keys_pressed = {};
    function init_keys() {for (let key in keys) {keys_pressed[keys[key]]=false;}}
    window.addEventListener('keydown', eventKeyDown, true);
    window.addEventListener('keyup'  , eventKeyUp  , true);
    function eventKeyDown(event) {
        if (event.keyCode in keys) {
            keys_pressed[keys[event.keyCode]]=true;  event.preventDefault();
        }
    }
    function eventKeyUp(event) {
        if (event.keyCode in keys) {
            keys_pressed[keys[event.keyCode]]=false; event.preventDefault();
        }
    }

    const v = {
        "color_background": `rgba(0,0,0,255)`,
    };


    function reset() {
        init_keys();
    }

    function timerEvent() {
        context.fillStyle = v.color_background;
        context.fillRect(0, 0, canvas.width, canvas.height);

        if (keys_pressed.ESCAPE) {pause(); reset();}

        if (!paused) {requestAnimationFrame(timerEvent);}
    }

    reset();
    start();
</script>
```

Background

```
        "color_background": 'rgba(0,0,0,255)',
    };
    v.background_image = new Image();
    v.background_image.src = "images/CopterLevel1.gif";

    function reset() {
        init_keys();
        v.view_x_pos   =  0;
    }
```

```
        context.fillStyle = v.color_background;
        context.fillRect(0, 0, canvas.width, canvas.height);

        v.view_x_pos += 1
        context.drawImage(v.background_image, -v.view_x_pos, 0);
```

Copter

```
        "color_background": 'rgba(0,0,0,255)',
    };
    v.copter_image = new Image();
    v.copter_image.src = "images/ship.gif";
    v.background_image = new Image();
```

```
    function reset() {
        init_keys();
        v.view_x_pos   =  0;
        v.copter_x_pos = 50;
        v.copter_y_pos = canvas.height / 2;
    }
```

```
        if (keys_pressed.ESCAPE) {pause(); reset();}
        if (keys_pressed.SPACE) {v.copter_y_pos+=-2;}
        else                    {v.copter_y_pos+= 1;}

        context.drawImage(
            v.copter_image, Math.round(v.copter_x_pos), Math.round(v.copter_y_pos)
        );
     }
```

Collision (Single Point)

```
</canvas>
<canvas id="canvas_colisions" width="320" height="240" style="display: none;"></canvas>
<script>
```

```
   const colisions_context=document.getElementById('canvas_colisions').getContext('2d');
    const v = {
```

```
        else                    {v.copter_y_pos+= 1;}

        colisions_context.fillStyle = v.color_background;
        colisions_context.fillRect(0, 0, canvas.width, canvas.height);
        colisions_context.drawImage(v.background_image, -v.view_x_pos, 0);

        const point = [v.copter_image.width/2, v.copter_image.height/2];
            const [r,g,b,a] = colisions_context.getImageData(
                v.copter_x_pos + point[0],
                v.copter_y_pos + point[1], 1, 1).data
            const pixel_color = `rgba(${r},${g},${b},${a})`;
            if (pixel_color != v.color_background) {
                reset();
            }

        context.drawImage(
            variables.copter_image,
```

Level

```
    const v = {
        "color_exit"      : 'rgba(255,255,0,255)',
        "level_number"    : 1,
```

```
    v.copter_image = new Image();
    v.copter_image.src = "images/ship.gif";
    ~~v.background_image = new Image();~~
    ~~v.background_image.src = "images/CopterLevel1.gif";~~

    function load_level(level_number) {
        v.background_image = new Image();
        v.background_image.src = `images/CopterLevel${level_number}.gif`;
        reset();
    }
```

```
            if (pixel_color == v.color_exit) {
                alert("Next Level");
                load_level(++v.level_number);
            }
            if (pixel_color != v.color_background) {
                reset();
```

```
    load_level(v.level_number);
    ~~reset();~~
    start();
```

Physics

```
    const keys = {
        27: 'ESCAPE',
        38: 'UP',
        40: 'DOWN',
        37: 'LEFT',
        39: 'RIGHT',
```

```
    const v = {
        v.copter_x_vel =  0;
        v.copter_y_vel =  0;
```

```
        if (keys_pressed.ESCAPE) {pause(); reset();}
        ~~if (keys_pressed.SPACE) {v.copter_y_pos+=-2;}~~
        ~~else                    {v.copter_y_pos+= 1;}~~
        if (keys_pressed.UP    ) {v.copter_y_vel += -0.1;}
        if (keys_pressed.DOWN  ) {v.copter_y_vel +=  0.1;}
        if (keys_pressed.LEFT  ) {v.copter_x_vel += -0.1;}
        if (keys_pressed.RIGHT ) {v.copter_x_vel +=  0.1;}
        v.copter_x_vel  = v.copter_x_vel * 0.99;
        v.copter_y_vel  = v.copter_y_vel * 0.99;
        v.copter_y_vel += 0.025;
        v.copter_x_pos += v.copter_x_vel;
        v.copter_y_pos += v.copter_y_vel;
```

Collision (Multipoint)

```
    const colisions_context
=document.getElementById('canvas_colisions').getContext('2d');
    const copter_colision_points = [[0,0],[32,9],[17,2],[22,12],[2,12]];
```

```
        ~~const point = [v.copter_image.width/2, v.copter_image.height/2];~~
        for (let point of copter_colision_points) {
            const [r,g,b,a] = colisions_context.getImageData(
```

```
        }

        context.drawImage(
            v.copter_image,
```

Parallax

```
    function load_level(level_number) {
        ~~v.background_image = new Image();~~
        ~~v.background_image.src = `images/CopterLevel${level_number}.gif`;~~
        const images = [new Image(), new Image(), new Image()];
        v.background_images = images;
        function load_background_image(layer, filename) {
            images[layer].onload = function() {images[layer] = this;}
            images[layer].src = filename;
        }
        load_background_image(2, `images/CopterLevel${level_number}_layer2.gif`);
        load_background_image(1, `images/CopterLevel${level_number}_layer1.gif`);
        load_background_image(0, `images/CopterLevel${level_number}.gif`);
        reset();
    }
```

```
        v.view_x_pos += 1
        ~~context.drawImage(v.background_image, -v.view_x_pos, 0);~~
        for (let i=2 ; i>=0 ; i--) {
            const offset_x = Math.round(v.view_x_pos / Math.pow(2,i));
            context.drawImage(v.background_images[i], -offset_x, 0);
        }
```

```
        colisions_context.drawImage(
            ~~v.background_image,~~
            v.background_images[0],
            -v.view_x_pos,
            0
        );
```