```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>Bullets shooter</title>

<style>
canvas {
    background: black;
}
</style>

<!-- http://taira-komori.jpn.org/sound/ -->
<audio id="shotAudio">
  <source src="laser2.mp3">
  Your WEB browser does not support audio tag.
</audio>
<audio id="explosionAudio">
  <source src="explosion1.mp3">
  Your WEB browser does not support audio tag.
</audio>
<audio id="enemyShotAudio">
  <source src="pyo1.mp3">
  Your WEB browser does not support audio tag.
</audio>
<audio id="deadAudio">
  <source src="destruction1.mp3">
  Your WEB browser does not support audio tag.
</audio>

<!-- http://dova-s.jp/bgm/play5046.html -->
<audio id="bgmAudio">
  <source src="5046.mp3">
  Your WEB browser does not support audio tag.
</audio>

<canvas width="500" height="500"></canvas>

<script src="explode.js"></script>

<script>

"use strict";

var shotAudio = document.getElementById('shotAudio');
var explosionAudio = document.getElementById('explosionAudio');
var enemyShotAudio = document.getElementById('enemyShotAudio');
var deadAudio = document.getElementById('deadAudio');
var bgmAudio = document.getElementById('bgmAudio');
var muteSE = true;

function playSE(audio) {
    if (muteSE) return;
    audio.currentTime = 0;
    audio.play();
}

var bgReady = false;
var bgImg = new Image();
bgImg.onload = function() {
    bgReady = true;
};
```

```
// https://www.nasa.gov/mission_pages/hubble/multimedia/index.html
bgImg.src = 'hubble_friday_05062016.jpg';

bgmAudio.loop = 'true';
bgmAudio.play();

function Enemy(canvas) {

    var x, y;
    var vx, vy;

    var life = 3;
    var dying = false;
    var explode = new Explode();

    var bullets = [];

    var er = 10; // enemy radius
    var br = 3; // bullet radius

    function spawn() {
        x = canvas.width*Math.random();
        y = -er;
        vx = 0;
        vy = 100*Math.random() + 50;
    };
    spawn();

    var lifeProp = [];
    lifeProp[3] = { color: 'white',  fireP: 0.001, numB:  4, daze: 0 };
    lifeProp[2] = { color: 'yellow', fireP: 0.005, numB: 10, daze: 2 };
    lifeProp[1] = { color: 'red',    fireP: 0.01,  numB: 20, daze: 5 };

    this.draw = function(ctx) {

        if (dying) {
            explode.draw(ctx, x, y);
        } else if (1 <= life) {
            ctx.beginPath();
            ctx.fillStyle = lifeProp[life].color;
            ctx.arc(x, y, er, 0, Math.PI*2, false);
            ctx.fill();
        }

        bullets.forEach(function(bullet) {
            var x = bullet.x;
            var y = bullet.y;
            ctx.beginPath();
            ctx.fillStyle = 'hsl(' + 360*bullet.life + ', 100%, 50%)';
            ctx.arc(x, y, br, 0, Math.PI*2, false);
            ctx.fill();
        });
    };

    this.fire = function() {
        var n = lifeProp[life].numB;
        for (var i = 0; i < n; i++) {
            var th = i*2*Math.PI/n;
            var vx = 300*Math.cos(th);
            var vy = 300*Math.sin(th);
```

```
            var bullet = { x: x, y: y, vx: vx, vy: vy, life: 1 };
            bullets.push(bullet);
        }
        playSE(enemyShotAudio);
    };

    this.fireAtRandom = function() {
        if (Math.random() < lifeProp[life].fireP) {
            this.fire();
            return true;
        }
        return false;
    };

    this.hit = function(tx, ty, r) {
        var dx = tx - x;
        var dy = ty - y;
        if (dx*dx + dy*dy < (er + r)*(er + r)) {
            return true;
        }
        return false;
    };

    this.bulletsHit = function(tx, ty, r) {
        for (var i = 0; i < bullets.length; i++) {
            var bullet = bullets[i];
            var dx = tx - bullet.x;
            var dy = ty - bullet.y;
            if (dx*dx + dy*dy < (br + r)*(br + r)) {
                return true;
            }
        }
        return false;
    };

    var lastTick;
    var th = Math.PI*Math.random();

    this.update = function() {

        if (dying && !explode.update()) {
            dying = false;
        }

        if (lastTick == undefined) {
            lastTick = Date.now();
        }

        var tick = Date.now();
        var dt = tick - lastTick;
        if (0 < life) {
            x += dt*vx/1000;
            x += lifeProp[life].daze*Math.cos(th);
            th += 1.5*dt*2*Math.PI/1000;
            y += dt*vy/1000;
            if (x < -er
            || canvas.width + er < x
            || canvas.height + er < y) {
                spawn();
            }
```

```
        }
        bullets.forEach(function(bullet) {
            bullet.x += dt*bullet.vx/1000;
            bullet.y += dt*bullet.vy/1000;
            bullet.life -= dt/1000/1.0;
        });

        bullets = bullets.filter(function(bullet) {
            return 0 < bullet.life;
        });
        lastTick = tick;
    };

    this.die = function() {
        dying = true;
        playSE(explosionAudio);
        explode.reset();
    };

    this.damage = function() {
        if (--life <= 0) {
            this.die();
        } else {
            this.fire();
        }
    };

    this.isAlive = function() {
        return 1 <= life;
    };
}

var enemies;

function init(canvas, n) {
    enemies = [];
    for (var i = 0; i < n; i++) {
        enemies.push(new Enemy(canvas));
    }
}

window.onload = function() {

    var canvas = document.getElementsByTagName('canvas')[0];
    var ctx = canvas.getContext('2d');

    var w = canvas.width;
    var h = canvas.height;

    var Status = { title:0, toStart:1, countDown:2,
                   inGame:3, gameClear:4, dying:5, gameOver:6 };
    var status = Status.title;

    var count;
    var myX, myY;
    var bullets, lastShoot;
    var explode = new Explode();
    var numEnemies;

    var keys = [];
```

```
document.onkeydown = function(e) {
    if (!keys[e.keyCode]) keys[e.keyCode] = 0;
    keys[e.keyCode]++;
};
document.onkeyup = function(e) {
    keys[e.keyCode] = 0;
};

var startTick = Date.now();

(function loop() {

    var tick = Date.now();

    ctx.save();
    ctx.clearRect(0, 0, w, h);

    if (bgReady) {
        var sx = (bgImg.width - canvas.width)/2;
        var dy = (tick - startTick)/100 % bgImg.height;

        ctx.drawImage(bgImg,
            sx, 0, canvas.width, canvas.height - dy,
            0, dy, canvas.width, canvas.height - dy);

        ctx.drawImage(bgImg,
            sx, bgImg.height - dy, canvas.width, dy,
            0, 0, canvas.width, dy);
    }

    if (keys[32 /*space*/] == 1) {
        if (status == Status.title) {
            status = Status.toStart;
            muteSE = false;
        }
        if (status == Status.gameOver) {
            status = Status.title;
        }
        if (status == Status.gameClear) {
            status = Status.toStart;
            muteSE = false;
        }
        keys[32]++; // pretend key repeat
    }

    var msg;
    var drawEnemies = false;
    var drawPlayer = false;
    var drawStatus = false;

    switch (status) {
    case Status.title:
        msg = 'Press space to start';
        numEnemies = 3;
        break;
    case Status.toStart:
        myX = canvas.width/2;
        myY = 4*canvas.height/5;
        bullets = [];
        init(canvas, numEnemies);
```

```
            status = Status.countDown;
            count = 3;
            (function() {
                setTimeout(function loop() {
                    if (--count < 0) {
                        status = Status.inGame;
                    } else {
                        setTimeout(loop, 1000);
                    }
                }, 1000);
            })();
            //break; // intentionally fall through
        case Status.countDown:
            msg = "" + count;
            drawPlayer = true;
            drawStatus = true;
            break;
        case Status.inGame:
            drawEnemies = true;
            drawPlayer = true;
            drawStatus = true;
            break;
        case Status.gameClear:
            msg = 'GAME CLEAR!!!';
            drawEnemies = true;
            drawPlayer = true;
            drawStatus = true;
            break;
        case Status.dying:
            explode.draw(ctx, myX, myY);
            if (!explode.update()) {
                status = Status.gameOver;
                muteSE = true;
            }
            drawEnemies = true;
            drawStatus = true;
            break;
        case Status.gameOver:
            msg = 'GAME OVER';
            drawEnemies = true;
            drawStatus = true;
            break;
        default:
            console.log('unknown status: ' + status);
        }

        if (drawPlayer) {
            if (keys[39 /*Right*/]) { myX += 2; }
            if (keys[37 /*Left */]) { myX -= 2; }
            if (keys[38 /*Up   */]) { myY -= 2; }
            if (keys[40 /*Down */]) { myY += 2; }
            if (keys[90 /* z */]) {
                if (lastShoot == undefined || 200 < tick - lastShoot) {
                    var bulletNotInUse = bullets.filter(function(bullet) {
                        return !bullet.inUse;
                    });
                    var bullet;
                    if (0 < bulletNotInUse.length) {
                        bullet = bulletNotInUse[0];
                    } else {
```

```
                bullet = {};
                bullets.push(bullet);
            }
            bullet.x = myX;
            bullet.y = myY;
            bullet.v = -2;
            bullet.inUse = true;
            playSE(shotAudio);
            lastShoot = tick;
        }
    }
}

var myR = 3;

drawEnemies && enemies.forEach(function(enemy) {
    enemy.update();
    if (enemy.isAlive()) {
        enemy.fireAtRandom();
    }
    enemy.draw(ctx);
    if (status != Status.inGame) return;
    if ((enemy.isAlive() && enemy.hit(myX, myY, myR))
    || enemy.bulletsHit(myX, myY, myR)) {
        explode.reset();
        status = Status.dying;
        playSE(deadAudio);
    }
});

if (drawPlayer) {
    ctx.beginPath();
    ctx.fillStyle = 'green';
    ctx.moveTo(myX, myY - 10);
    ctx.lineTo(myX + 10, myY + 10);
    ctx.lineTo(myX - 10, myY + 10);
    ctx.fill();
    ctx.beginPath();
    ctx.fillStyle = 'skyBlue';
    ctx.arc(myX, myY, myR, 0, Math.PI*2, false);
    ctx.fill();
}

bullets && bullets.forEach(function(bullet) {
    if (!bullet.inUse) return;

    ctx.beginPath();
    ctx.fillStyle = 'yellow';
    var x = bullet.x;
    var y = bullet.y;
    ctx.arc(x, y, 2, 0, Math.PI*2, false);
    ctx.fill();
    drawEnemies && enemies.forEach(function(enemy) {
        if (enemy.isAlive() && enemy.hit(x, y, 2)) {
            enemy.damage();
            bullet.inUse = false;
        }
    });
    bullet.y += bullet.v;
    if (bullet.y < 0) {
```

```
                bullet.inUse = false;
            }
        });

        if (drawStatus) {
            var remainingEnemies = enemies.filter(function(enemy) {
                return enemy.isAlive();
            }).length;
            if (status == Status.inGame && remainingEnemies <= 0) {
                status = Status.gameClear;
                muteSE = true;
                numEnemies++;
            }
            ctx.font = '20pt Calibri';
            ctx.fillStyle = 'white';
            ctx.textAlign = 'right';
            ctx.textBaseline = 'top';
            var s = "enemie(s): " + remainingEnemies;
            ctx.fillText(s, canvas.width - 10, 0);
        }

        if (msg) {
            ctx.font = '40pt Calibri';
            ctx.fillStyle = 'white';
            ctx.textAlign = 'center';
            ctx.textBaseline = 'middle';
            ctx.fillText(msg, canvas.width/2, canvas.height/2);
        }

        ctx.restore();

        requestAnimationFrame(loop);
    })();
};

</script>
```