3D Tetris

three.js

Mario Jugurčić

UVOD

- Inicijalno u WebGL → three.js
- Olakšan rad, sloj apstrakcije
- three.js, javascript, jquery, html i css

Inicijalni problemi

- three.js je relativno nova tehnologija
- Prva verzija izašla je 2010. godine
- Danas je aktualna verzija r70
- Mnogo izmjena iz verzije u verziju
- Nepotpuna i štura dokumentacija

Izrada projekta

- Projekt je razdjeljen u 3 skripte
  - tetris.js
  - tetris.block.js
  - tetris.board.js

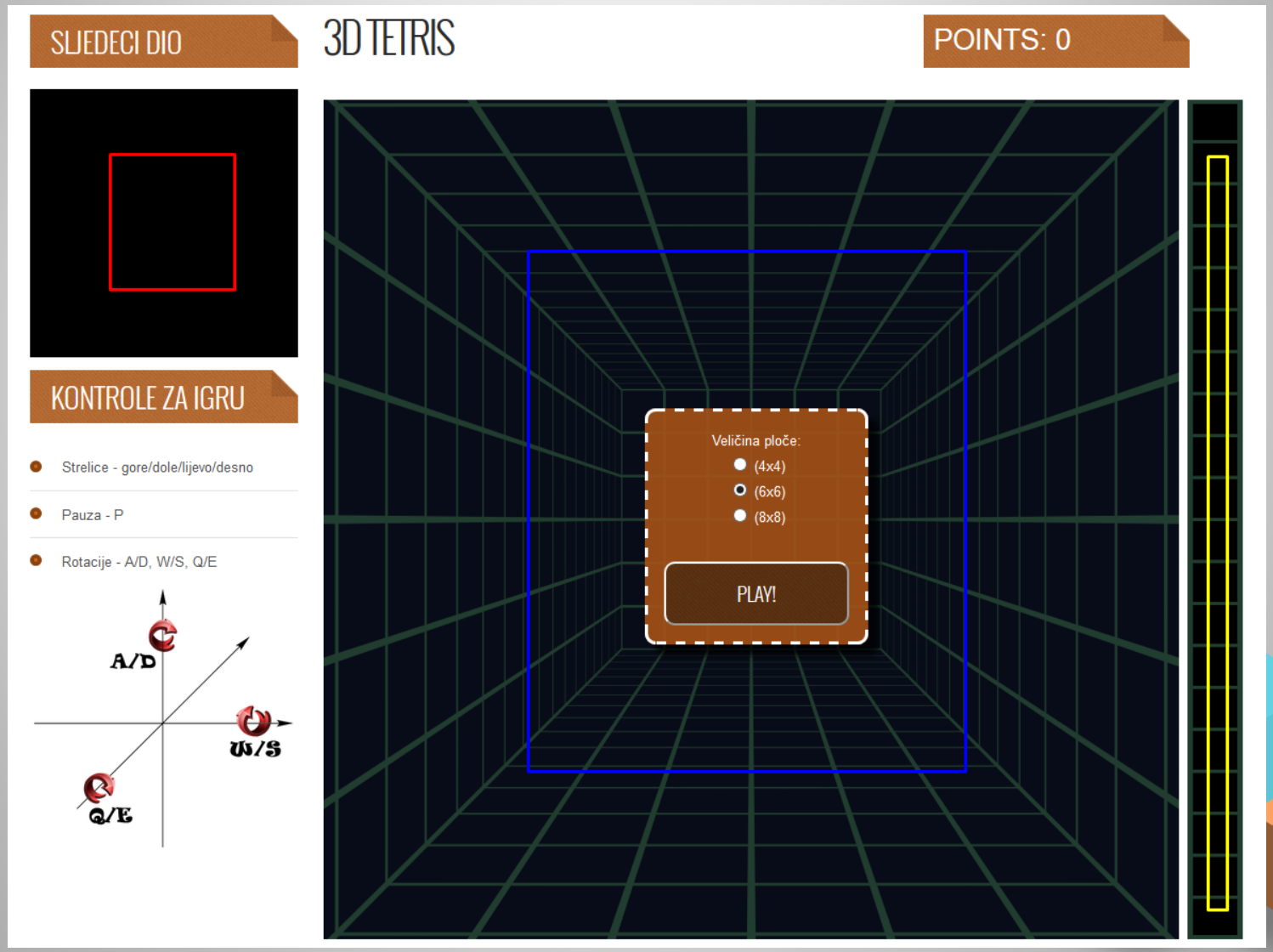
- tetris.js – glavna logika programa, prima korisničke unose
- tetris.block.js – sve operacije vezane uz jedan tetris dio (pomicanje, rotacije, generiranje...)
- tetris.board.js – informacije o ploči za igru (zauzetost polja, popunjenost redaka, najveća visina...)

Izrada projekta

- Korisnik unosi veličinu terena i pokreće igru
- Stvara se teren (velika kocka razdjeljena na segmente)
- Generira se blok od manjih kockica
- Blok se svake sekunde spušta za jedan red niže
- Nakon što dotakne dno blok se ponovo razdvaja na manje kockice

- Postoje provjere kolizije (blok sa zidovima, blok sa ostalim nepomičnim blokovima)
- Postoji provjera popunjenosti redova
- Postoji provjera najveće dostignute visine

Izrada projekta



tetris.js

- Postoje globalne varijable u koje se spremaju svi podaci (window.Tetris i window.Tetris2)
- Inicijalno postavljanje početnih vrijednosti

```
Tetris.renderer = new THREE.WebGLRenderer();
Tetris.camera = new THREE.PerspectiveCamera(VIEW_ANGLE, ASPECT, NEAR, FAR);
Tetris.scene = new THREE.Scene();


Tetris.camera.position.z = 800;
Tetris.scene.add(Tetris.camera);


Tetris.renderer.setSize(WIDTH, HEIGHT);

 threeContainer.append(Tetris.renderer.domElement);
```

tetris.js

- Postavljanje terena

```
var texture1 = THREE.ImageUtils.loadTexture('images/plocica.png', {}, function () {
    Tetris.renderer.render(Tetris.scene, Tetris.camera);
});


texture1.wrapS = texture1.wrapT = THREE.RepeatWrapping;
texture1.repeat.set(BC.splitX, BC.splitX);


texture1.anisotropy = Tetris.renderer.getMaxAnisotropy();


var materials = [
    new THREE.MeshBasicMaterial({map: texture3, side: THREE.BackSide}),
    new THREE.MeshBasicMaterial({map: texture3, side: THREE.BackSide}),
    new THREE.MeshBasicMaterial({map: texture2, side: THREE.BackSide}),
    new THREE.MeshBasicMaterial({map: texture2, side: THREE.BackSide}),
    new THREE.MeshBasicMaterial({map: texture1, side: THREE.BackSide}),
    new THREE.MeshBasicMaterial({map: texture1, side: THREE.BackSide})
];


var boundingBox = new THREE.Mesh(
    new THREE.BoxGeometry(BC.width, BC.height, BC.depth, BC.splitX, BC.splitY, BC.splitZ),
    new THREE.MeshFaceMaterial(materials)
);


Tetris.scene.add(boundingBox);
Tetris.boundingBox = boundingBox;
}
```

tetris.js

- Game loop

```
function animate() {
    if (!Tetris.pause) {
        var time = Date.now();
        Tetris.frameTime = time - Tetris._lastFrameTime;
        Tetris._lastFrameTime = time;
        Tetris.cumulatedFrameTime += Tetris.frameTime;


        while (Tetris.cumulatedFrameTime > Tetris.gameStepTime) {
            Tetris.cumulatedFrameTime = 0;
            Tetris.Block.move(0, 0, -1);
        }


        Tetris.renderer.render(Tetris.scene, Tetris.camera);
    }
    if (!Tetris.gameOver) window.requestAnimationFrame(animate);
};
```

tetris.block.js

- Oblici blokova

```
Tetris.Block.shapes = [
    [
        {x:0, y:0, z:0},
        {x:1, y:0, z:0},
        {x:1, y:1, z:0},
        {x:1, y:2, z:0}
    ],
    [
        {x:0, y:0, z:0},
        {x:0, y:1, z:0},
        {x:0, y:2, z:0},
    ],
    [
        {x:0, y:0, z:0},
        {x:0, y:1, z:0},
        {x:1, y:0, z:0},
        {x:1, y:1, z:0}
    ],
    [
        {x:0, y:0, z:0},
        {x:0, y:1, z:0},
        {x:0, y:2, z:0},
        {x:1, y:1, z:0}
    ],
    [
        {x:0, y:0, z:0},
        {x:0, y:1, z:0},
        {x:1, y:1, z:0},
        {x:1, y:2, z:0}
    ]
];
```

tetris.block.js

- Generiranje bloka

```
var piece = new THREE.Geometry();
geometry = new THREE.BoxGeometry(Tetris.blockSize, Tetris.blockSize, Tetris.blockSize);
piece.merge(geometry, geometry.matrix);

for (i = 1; i < Tetris.Block.shape.length; i++) {
    tmpGeometry = new THREE.Mesh(new THREE.BoxGeometry(Tetris.blockSize, Tetris.blockSize,
                    Tetris.blockSize));
    tmpGeometry.position.x = Tetris.blockSize * Tetris.Block.shape[i].x;
    tmpGeometry.position.y = Tetris.blockSize * Tetris.Block.shape[i].y;
    tmpGeometry.updateMatrix();
    piece.merge(tmpGeometry.geometry, tmpGeometry.matrix);
}


Tetris.Block.mesh = new THREE.Mesh(piece,
    new THREE.MeshBasicMaterial({color:0xffffff, transparent:true, opacity:0.2, side: THREE.DoubleSide})
);


Tetris.scene.add(Tetris.Block.mesh);
```

tetris.board.js

- Inicijalizacija ploče

```
Tetris.Board.init = function (_x, _y, _z) {
    Tetris.Board.fields = [];
    for (var x = 0; x < _x; x++) {
        Tetris.Board.fields[x] = [];
        for (var y = 0; y < _y; y++) {
            Tetris.Board.fields[x][y] = [];
            for (var z = 0; z < _z; z++) {
                Tetris.Board.fields[x][y][z] = Tetris.Board.FIELD.EMPTY;
            }
        }
    }
};
```

tetris.board.js

- Provjera kolizija

```
for (i = 0; i < shape.length; i++) {
    if ((shape[i].x + posx) < 0 || (shape[i].y + posy) < 0 || (shape[i].x + posx) >= fields.length ||
                    (shape[i].y + posy) >= fields[0].length) {
        return Tetris.Board.COLLISION.WALL;
    }


    if (fields[shape[i].x + posx][shape[i].y + posy][shape[i].z + posz - 1] ===
                    Tetris.Board.FIELD.PETRIFIED) {
        return ground_check ? Tetris.Board.COLLISION.GROUND : Tetris.Board.COLLISION.WALL;
    }


    if((shape[i].z + posz) <= 0) {
        return Tetris.Board.COLLISION.GROUND;
    }
}
```

tetris.board.js

- Metoda „checkCompleted”
  - Provjerava ispunjenost reda odozgo prema gore
  - Ako je red ispunjen miče se te se ostali redovi spuštaju

- Metoda „checkHeight”
  - Provjerava najvišu dosegnutu točku unutar ploče
  - Ažurira pokazivač sa strane

tetris.block.js

- Transformacija bloka u manje dijelove prilikom pada na dno

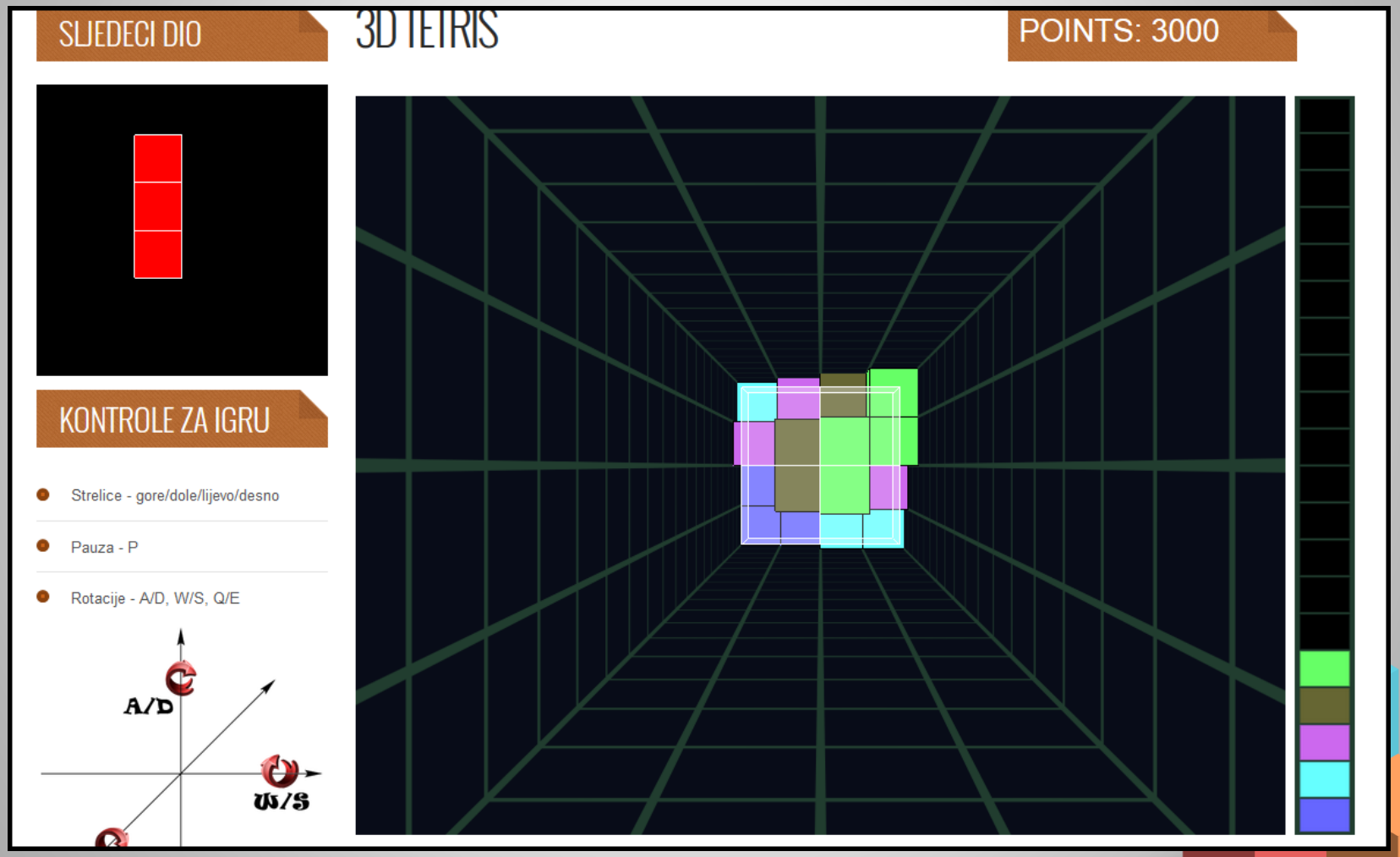
```
Tetris.Block.petrify = function () {
    var shape = Tetris.Block.shape;
    for (var i = 0; i < shape.length; i++) {
        Tetris.addStaticBlock(Tetris.Block.position.x + shape[i].x, Tetris.Block.position.y + shape[i].y, Tetris.Block.position.z +
                shape[i].z);
        Tetris.Board.fields[Tetris.Block.position.x + shape[i].x][Tetris.Block.position.y + shape[i].y][Tetris.Block.position.z +
                shape[i].z] = Tetris.Board.FIELD.PETRIFIED;
    }
};
```

tetris.js

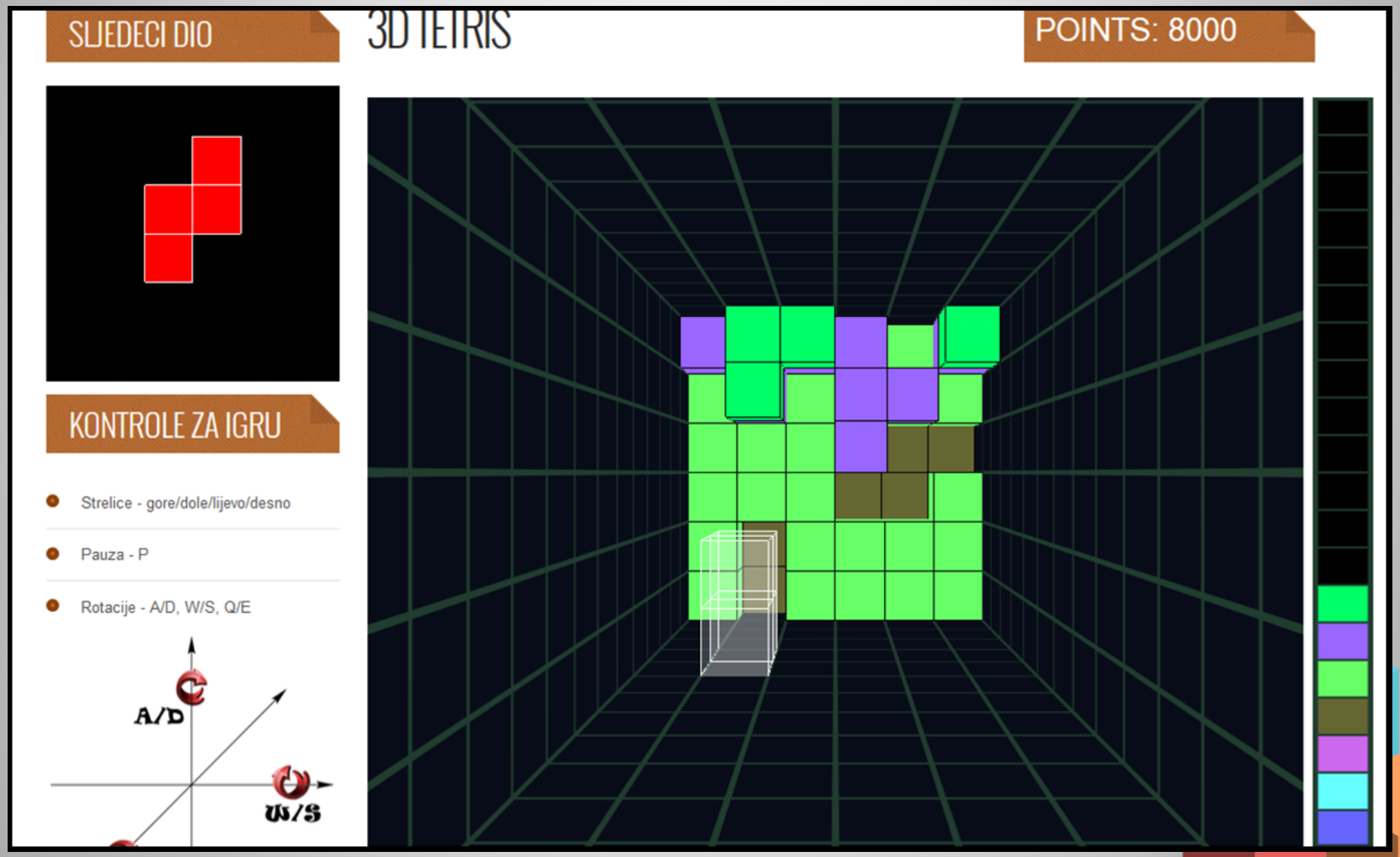
- Transformacija bloka u manje dijelove prilikom pada na dno

```
Tetris.addStaticBlock = function (x, y, z) {
    var mesh = new THREE.Mesh(new THREE.BoxGeometry(Tetris.blockSize, Tetris.blockSize,
                    Tetris.blockSize),
                    new THREE.MeshBasicMaterial({color: Tetris.zColors[z]})
    )

    mesh.position.x = (x - Tetris.boundingBoxConfig.splitX / 2) * Tetris.blockSize + Tetris.blockSize / 2;
    mesh.position.y = (y - Tetris.boundingBoxConfig.splitY / 2) * Tetris.blockSize + Tetris.blockSize / 2;
    mesh.position.z = (z - Tetris.boundingBoxConfig.splitZ / 2) * Tetris.blockSize + Tetris.blockSize / 2;

    Tetris.scene.add(mesh);
    Tetris.staticBlocks[x][y][z] = mesh;
  };
```

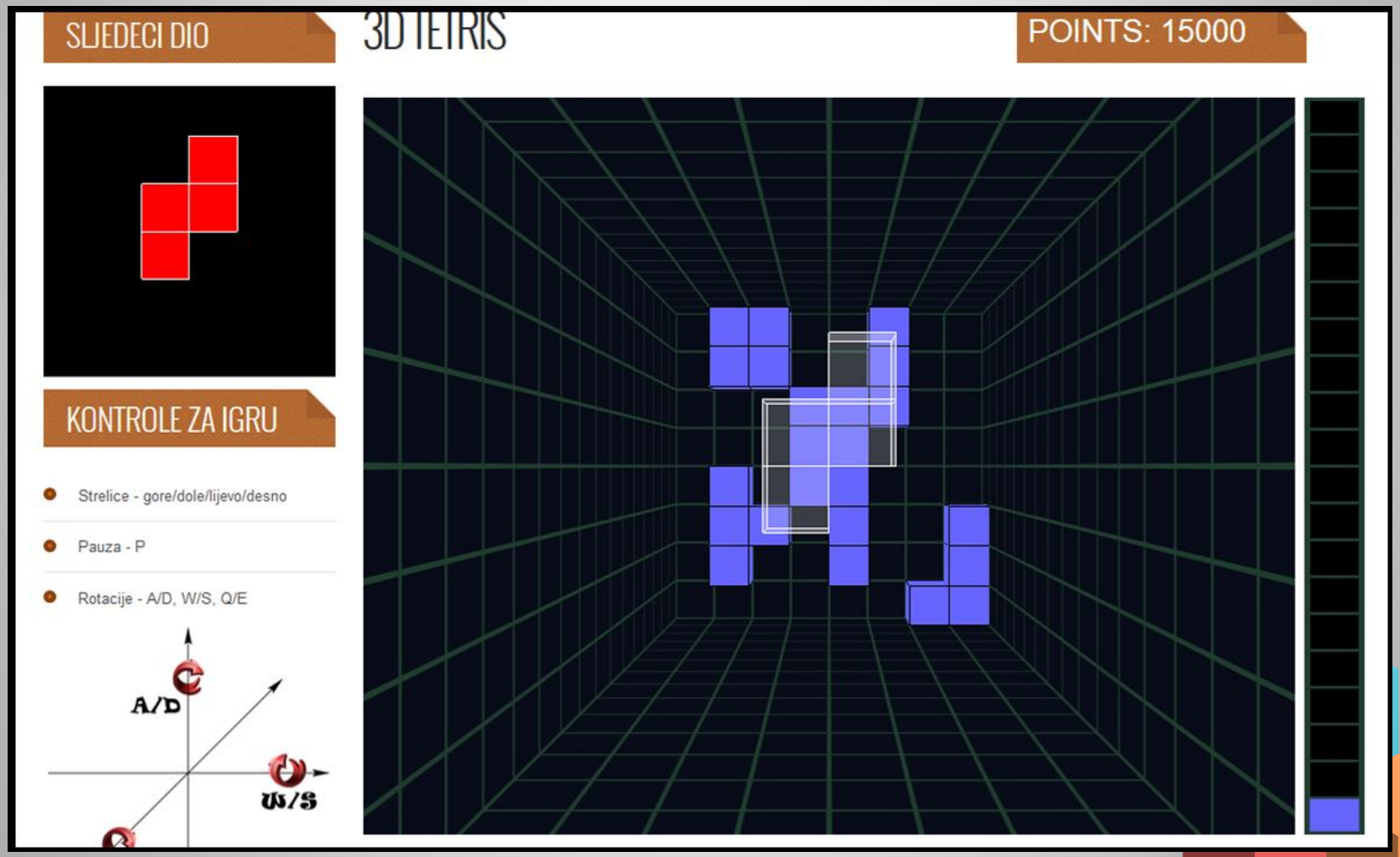
Nekoliko screenshot-ova



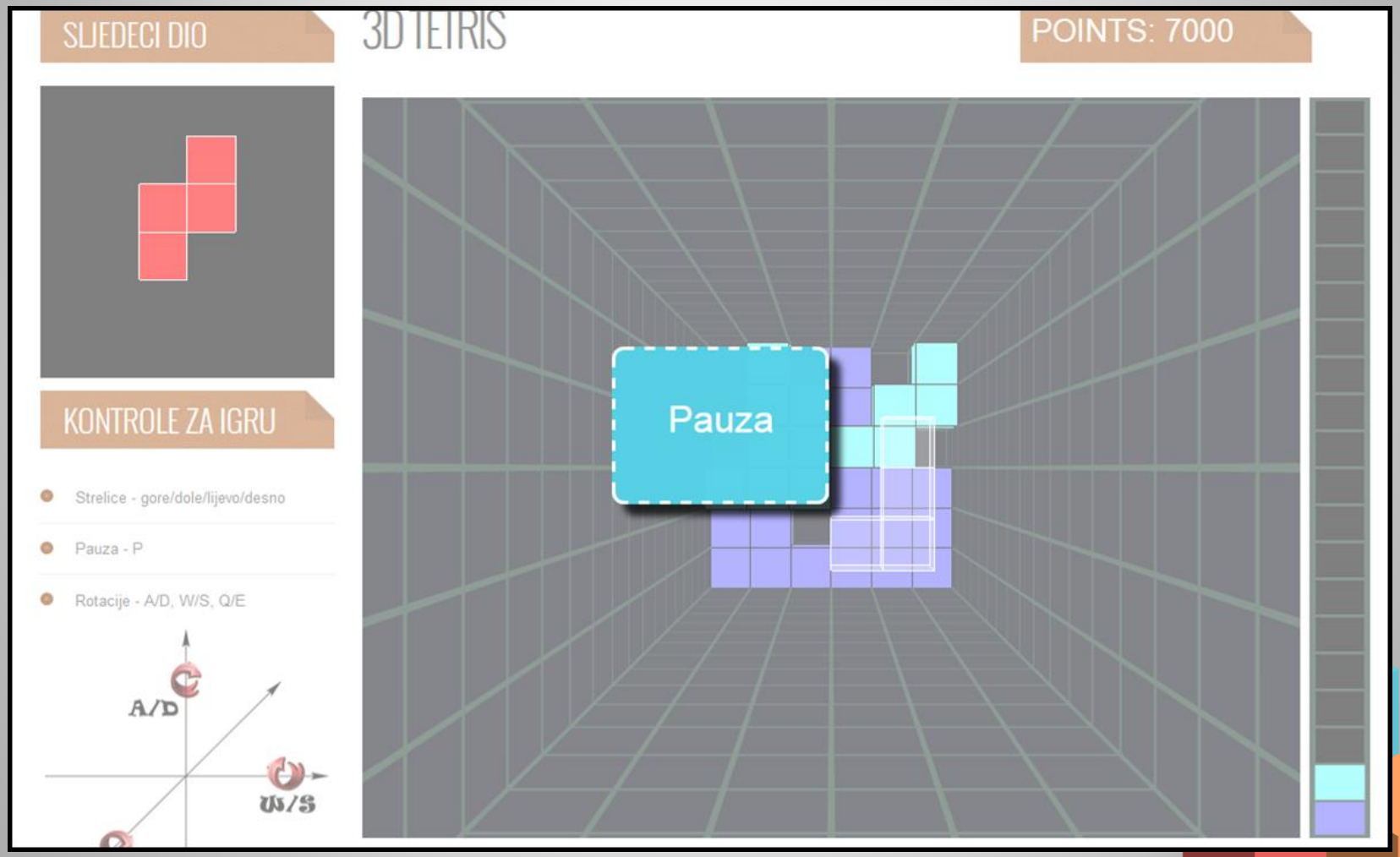
Nekoliko screenshot-ova



Nekoliko screenshot-ova



Nekoliko screenshot-ova



Nekoliko screenshot-ova