Tetris

```
function iterGame(frametime) {
    frameDiff = frametime-lastFrameTime;
    lastFrameTime = frametime;
    update(frameDiff);
    render(frameDiff);
    if(gameRunning)
        requestAnimFrame( iterGame );
    else
        console.log("quitting... score was : " + score);
}
```

Sér um keyrslu leiksins, game running verður false ef þú tapar

```
var lastFrameTime = 0;
var frameDiff;
var lastStraight = true;
function update(frametime) {
    currentBlock.update(frametime);
    if(currentBlock._isDeadNow) {
        if(lastStraight) {
            currentBlock = new CurvedTrisis({
                cx : mapwidth/2,
                cy : mapheight-1,
                cz : mapbreadth/2,
                colorBuffer : colorBuffers[bufferIterator++]
            });
        }else{
            currentBlock = new StraightTrisis({
                cx : mapwidth/2,
                cy : mapheight-1,
                cz : mapbreadth/2,
                colorBuffer : colorBuffers[bufferIterator++]
            })
        }
        bufferIterator %= colorBuffers.length;

    }


}
```

Update er það fyrsta sem er gert, sér um að hreyfa núverandi block, og gera nýjan ef þörf krefur

```
function render(frametime)
{
    gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    //checkInputs();
    var mvstack = [];

    // staÃƒÂºsetja ÃƒÂ¡horfanda og meÃƒÂ°hÃƒÂ¶ndla mÃƒÂºsarhreyfingu
    var mv = lookAt( vec3(0.0, 0.0, zDist), vec3(0.0, 0.0, 0.0), vec3(0.0, 1.0, 0.0) );
    mv = mult( mv, rotate( parseFloat(spinX), [1, 0, 0] ) );
    mv = mult( mv, rotate( parseFloat(spinY), [0, 1, 0] ) );
    mvstack.push(mv);
    spatialManager.render(mv);
    mv = mvstack.pop();
    container.render(mv);
    currentBlock.render(mv);

}
```

Á eftir því er render, sem sér um að teikna allt

Entity er grunn einingin, er í raun abstract

```
Entity.prototype.findCoordsRotation = function(x,y,z,xRot,yRot,zRot) {
    xRot = radians(xRot);
    yRot = radians(yRot);
    zRot = radians(zRot);
    var x2 = x;
    var y2 = y*Math.cos(xRot) + z*(-Math.sin(xRot));
    var z2 = y*Math.sin(xRot) + z*Math.cos(xRot);

    var x3 = x2*Math.cos(yRot) + z2*Math.sin(yRot);
    var y3 = y2;
    var z3 = x2*(-Math.sin(yRot)) + z2*Math.cos(yRot);

    var x4 = x3*Math.cos(zRot) + y3*(-Math.sin(zRot));
    var y4 = x3*Math.sin(zRot) + y3*Math.cos(zRot);
    var z4 = z3;
    return vec3(Math.round(x4),Math.round(y4),Math.round(z4));
}
Entity.prototype.getPosCubes = function() {
    var pos = this.getPos(); // fer einn til vinstri og einn upp    x-rot
    pos = vec3(pos.cx,pos.cy,pos.cz);
    var pos1 = add(pos, this.findCoordsRotation(this.cubes[0].offsetx,this.cubes[0].offsety,this.cubes[0].offsetz,this.xRotation,this.yRotation,this.zRotation));
    var pos2 = add(pos, this.findCoordsRotation(this.cubes[2].offsetx,this.cubes[2].offsety,this.cubes[2].offsetz,this.xRotation,this.yRotation,this.zRotation));
    return {
        cube1 : pos,
        cube2 : pos1,
        cube3 : pos2
    };
};

Entity.prototype.setPosCubes = function(){
    var cubePos = this.getPosCubes();
    var i = 0;
    for(var pos in cubePos) {
        this.cubes[i].setPos(cubePos[pos]);
        i++;
    }
};
Entity.prototype.mapCubes = function() {
    var canFit = true;
    for(var i = 0; i < this.cubes.length; i++) {
        canFit = canFit && spatialManager.canFit(this.cubes[i]);
    }
    if(!canFit) return false;
    for(var i = 0; i < this.cubes.length; i++) {
        spatialManager.register(this.cubes[i]);
    }
    return true;
};
Entity.prototype.unmapCubes = function() {
    for(var i = 0; i < this.cubes.length; i++) {
        spatialManager.unregister(this.cubes[i]);
    }
};
Entity.prototype.setTo = function(prev) {
    this.xRotation = prev.xRotation;
    this.yRotation = prev.yRotation;
    this.zRotation = prev.zRotation;
    this.cx = prev.cx;
    this.cy = prev.cy;
    this.cz = prev.cz;
}
```

Sér um færslu kubba á milli og uppfærslu staðsetningu

Ofan á hana eru byggð straight trisis og curved trisis sem að innihalda 3 blocks hver

```
function StraightTrisis(descr) {
    this.setup(descr);
    this["cubes"] = [];
    for(var i = 0; i < 3; i++) {
        this["cubes"][i] = new Entity({
            offsetx: 1-i,
            offsety: 0,
            offsetz: 0,
            colorBuffer: descr.colorBuffer
        });
    }
    this.setPosCubes();
}
StraightTrisis.prototype = new Entity();
```

Spatial manager sér í rauninni um mest allt

```
spatialManager.prototype.render = function(mv) {
    var mvstack = [];
    mv = mult(mv, scale4(0.2,0.2,0.2));
    for(var y = 0; y < mapheight; y++) {
        for(var x = 0; x < mapwidth; x++) {
            for(var z = 0; z < mapbreadth; z++) {
                if(this.map[y][x][z]) {
                    mvstack.push(mv);
                    mv = mult(mv,translate((3-x)/2,(-10+y)/2, (3-z)/2));
                    util.drawCube(mv, this.map[y][x][z], vBuffer);
                    mv = mvstack.pop();
                }
            }
        }
    }
};
```

Teiknar allt sem að búið er að skrá í borðið

```
spatialManager.prototype.register = function(cube) {
    var x = cube.cx;
    var y = cube.cy;
    var z = cube.cz;
    if(this.outOfBounds(x,y,z)) return false;
    this.map[y][x][z] = cube.colorBuffer;
}
spatialManager.prototype.unregister = function(cube) {
    var x = cube.cx;
    var y = cube.cy;
    var z = cube.cz;
    if(this.outOfBounds(x,y,z)) return;
    this.map[y][x][z] = false;
}
spatialManager.prototype.canFit = function(cube) {
    var x = cube.cx;
    var y = cube.cy;
    var z = cube.cz;
    if(this.outOfBounds(x,y,z) || this.map[y][x][z]) return false;
    return true;
}
```

Skráir kubba og afskráir þá, og athugar hvort mögulegt sé að fara þangað

```
spatialManager.prototype.addFloor = function() {
    this.map[19] = [];
    for(var x = 0; x < mapwidth; x++) {
        this.map[19][x] = [];
        for(var z = 0; z < mapbreadth; z++) {
            this.map[19][x][z] = false;
        }
    }
}
spatialManager.prototype.checkFloor = function(y) {
    var full = true;
    for(var x = 0; full && x < mapwidth; x++) {
        for(var z = 0; z < mapbreadth; z++) {
            if(!this.map[y][x][z]){
                var full = false;
            }
        }
    }
    if(full) {
        this.map.splice(y,1);
        this.addFloor();
        score += 100;
        document.getElementById("score").innerHTML = "Score: " + score;
        return true;
    }
    return false;
}
```

Ef að hæð er full sér þetta um að eyða henni út og uppfæra score-ið