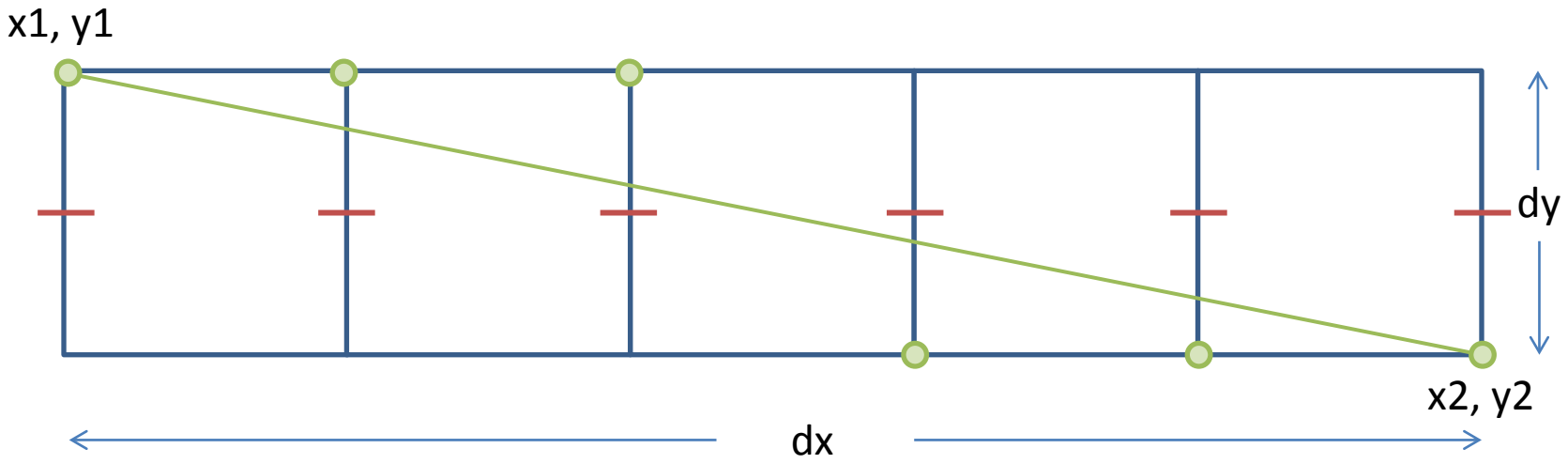
Computergrafik 1

Blatt 7

Aufgabe 1

```
dx := x2-x1; dy := y2-y1
d := 2*dy – dx; DO := 2*dy;
dNO := 2*(dy - dx)
x := x1; y := y1
setpixel (x,y)
fehler := d
WHILE x < x2
    x := x + 1
    IF fehler <= 0 THEN
        fehler := fehler + DO
    ELSE
        y := y + 1
        fehler = fehler + dNO
    END IF
    setpixel (x,y)
END WHILE
```

Idee



Idee: Wenn grüne Linie < Mittelpunkt

dann setze PixelOst

sonst setze PixelSüdOst

Umsetzung

```
<!doctype html>
<html lang="en">
    <head>
        <title>Bresenham</title>


        <style>
            #content{width:500px;height:500px; margin:10px auto;}
            #canvas{}
            body {font-family: 'Open Sans', sans-serif;}
            h1 {font-family: 'Fredericka the Great', cursive;
            text-align:center;
            }
            p {
            text-align:center;
            }
        </style>

        <link href='http://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'>
        <link href='http://fonts.googleapis.com/css?family=Fredericka+the+Great' rel='stylesheet' type='text/css'>


        <script>
```

Die ganze Arbeit findet hier statt…

```
        </script>

    </head>

    <body onload="init()">

    <div id="content">

        <h1>Bresenham-Algorithm</h1>
        <p><small>Draw straight lines on the grid below.</small></p>
        <canvas id="canvas" width="500" height="500">

        </canvas>
    </div>
    </body>

</html>
```

Das canvas-Element

```
<script>

var canvas;
var context;

var pixelSize = 10;

var startX;
var startY;

var endX;
var endY;

function init(){
    initCanvas();
}

function initCanvas(){
canvas = document.getElementById('canvas');
context = canvas.getContext('2d');
context.fillStyle = "#ffffff";
context.strokeStyle = "#000000";
context.fillRect(0,0,500,500);
canvas.addEventListener('mousedown', onMouseDown);
canvas.addEventListener('mouseup', onMouseUp);
drawGrid();
}
```

Seitenlänge der virtuellen Pixel

Startpunkt der Linie

Endpunkt der Linie

Maus-Listener

Die Maus-Listener

```
51          function onMouseDown(evt){
52
53                  var x = evt.pageX;
54                  var y = evt.pageY;
55
56                  x -= canvas.offsetLeft;
57                  y -= canvas.offsetTop;
58
59                  startX = x;
60                  startY = y;
61
62          }
63
64          function onMouseUp(evt){
65
66                  var x = evt.pageX;
67                  var y = evt.pageY;
68
69                  x -= canvas.offsetLeft;
70                  y -= canvas.offsetTop;
71
72                  endX = x;
73                  endY = y;
74
75                  /*drawLine(startX,startY,endX,endY);*/
76
77                  bresenham(startX,startY,endX,endY);
78
79          }
```

Mausposition != Position im Canvas!

Bresenham-Algorithmus

```
99
100     function bresenham(x1,y1,x2,y2){
101
102
103         x1 = x1 - (x1%pixelSize);
104         x2 = x2 - (x2%pixelSize);
105         y1 = y1 - (y1%pixelSize);
106         y2 = y2 - (y2%pixelSize);
107         // Differences and Error Check
108         var dx = x2 - x1;
109         var dy = y2 - y1;
110
111
112         var d = 2*dy - dx;
113         var deltaE = 2*dy;
114         var deltaNE = 2*(dy-dx);
115
116         var x = x1;
117         var y = y1;
118
119         context.fillStyle = "#008800";
120         context.fillRect(x,y,pixelSize,pixelSize);
121
122         while (x < x2){
123             if(d<=0){
124                 d = d + deltaE;
125                 x = x + pixelSize;
126             }
127             else{
128                 d = d + deltaNE;
129                 x = x + pixelSize;
130                 y = y + pixelSize;
131             }
132             context.fillRect(x,y,pixelSize,pixelSize);
133         }
134     }
```

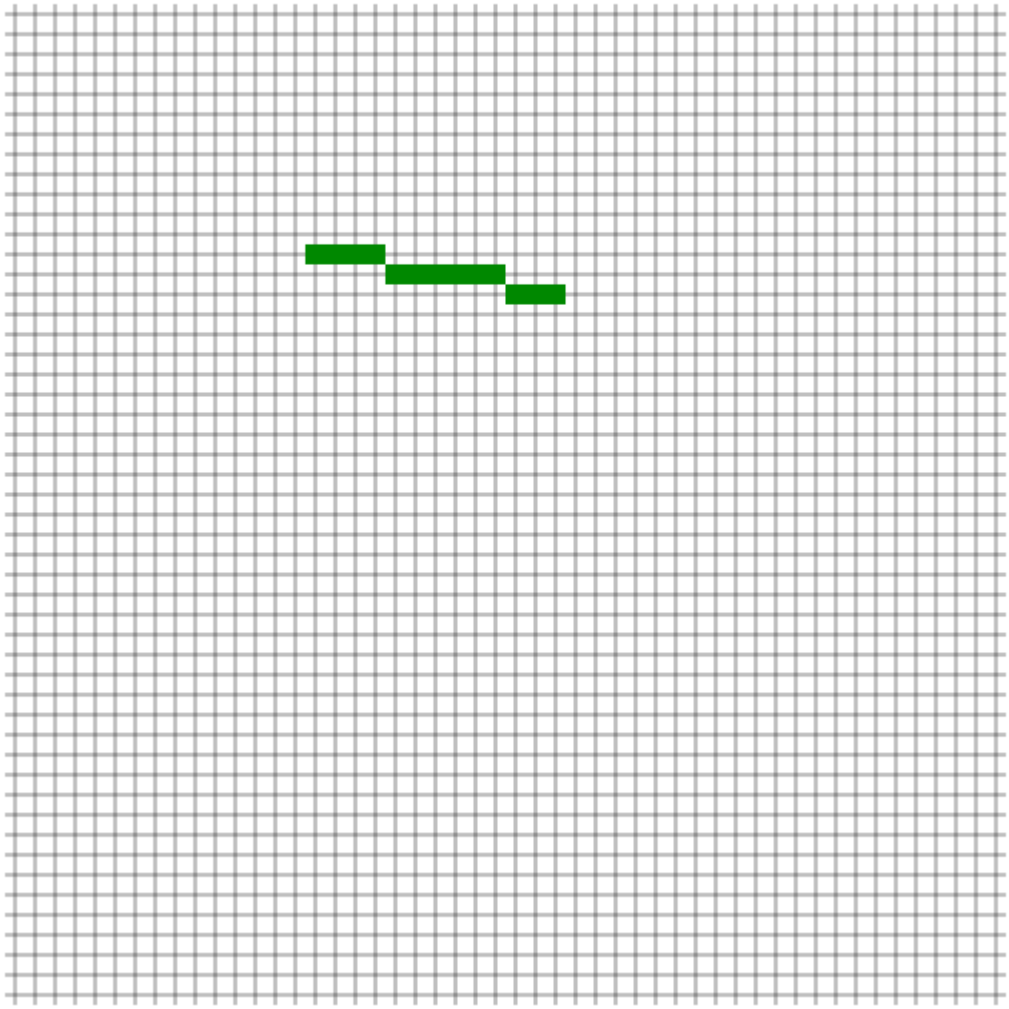
Virtuell vergrößerte Pixel

Ersten „Pixel“ setzen

Weitere „Pixel“ setzen

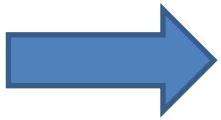
Bresenham-Algorithm

Draw straight lines on the grid below.



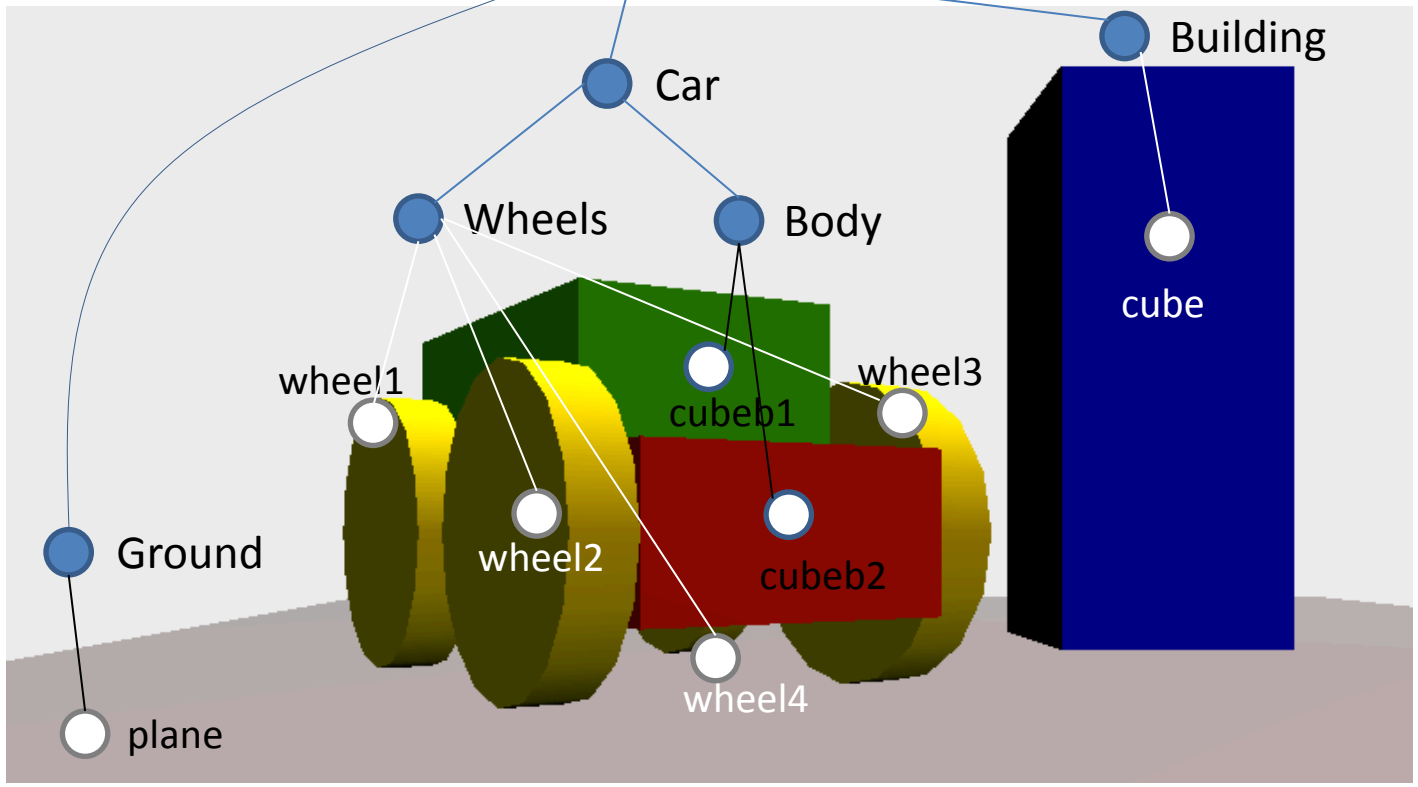
Aufgabe 2

- Ein Szenegraph ist eine Sammlung von Knoten, die in einer Graphstruktur organisiert sind
- Dieser Graph ist zyklenfrei und hat genau eine Wurzel (gerichteter Baum)
- Der Wurzelknoten enthält die Gesamtszene
- Jeder Knoten hat eine eigene Transformationsmatrix
- Wird ein Knoten transformiert, werden auch alle untergeordneten Knoten transformiert

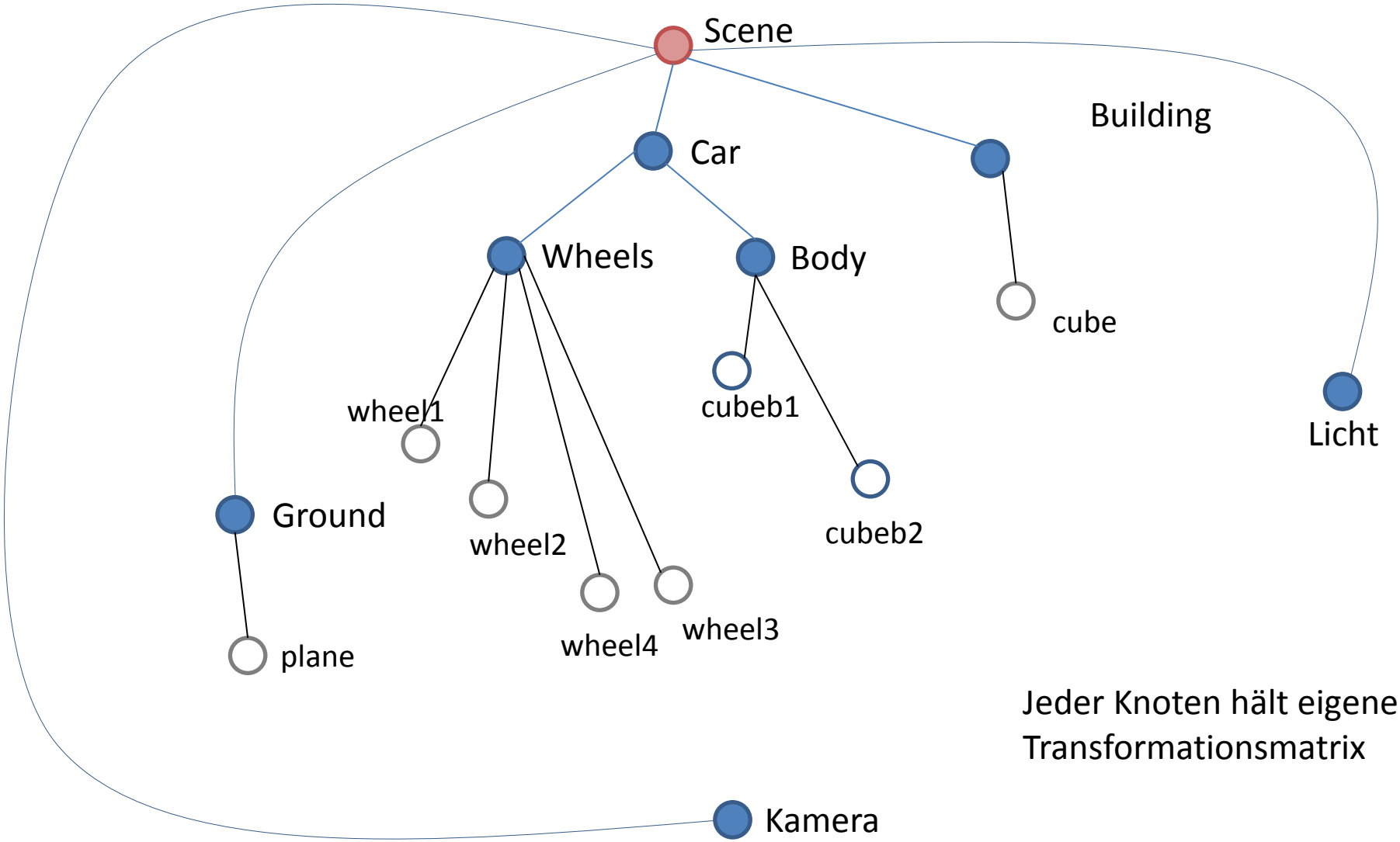


Vereinfachte hierarchische Modellierung

Aufgabe 2


Scene
Building
Car
Wheels
Body
cube
wheel1
wheel3
cubeb1
Licht
Ground
wheel2
cubeb2
wheel4
plane
Kamera

Aufgabe 2



Jeder Knoten hält eigene Transformationsmatrix

Aufgabe 3

Szenegraph mit THREE.js:

1. HTML-Dokument mit Link auf die Datei THREE.js
2. Eine Szene
3. Einen Renderer
4. Eine Kamera
5. Licht
6. Objekte

HTML-Dokument

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <title>CG1 &Uuml;bungsblatt 7</title>
6
7      <script src="Three.js"></script>
8
9      <script type="text/javascript">
```

Eine Szene

```
/*Eine Szene erzeugen*/
var scene = new THREE.Scene();
```

Ein Renderer

```
13
14 ▼    function init(){
15          var renderer = new THREE.WebGLRenderer({antialias: true});
16          renderer.setSize(width,height);
17
18          document.body.appendChild(renderer.domElement);
19
20          renderer.setClearColorHex(0xEEEEEE, 1.0);
21          renderer.clear();
22
```

Eine Kamera

```
/*Eine Kamera erzeugen und platzieren*/
var camera = new THREE.PerspectiveCamera(45, width/height, 1, 10000);

camera.position.x = 3;
camera.position.y = 1;
camera.position.z = 5;
```

-> Kamera-Knoten mit Transformation

Licht

```
/*Eine Lichtquelle*/
var directionalLight = new THREE.DirectionalLight (0xffffff, 0.5);
directionalLight.position.set(0,0,1);
```

-> gerichtetes Licht: Intensität + Richtung (siehe Vorlesung)

Objekte

```
/*Einen Würfel erzeugen und positionieren*/
var cube = new THREE.Mesh(new THREE.CubeGeometry(2,2,2),
                          new THREE.MeshLambertMaterial({color:0xFFFF00}));

cube.rotation.y = Math.PI/6;
cube.position.x = 3;
cube.position.y = 1;
cube.position.z = -3;
```

-> Geometrie-Knoten mit Transformation

Szene aufbauen und rendern

```
46          /*Der Szene den Würfel, das Licht und die Kamera hinzufügen*/
47          scene.add(cube);
48          scene.add(directionalLight);
49          scene.add(camera);
50
51          /*rendern*/
52          renderer.render(scene, camera);
53      }
54      </script>
```

Ergebnis