Wir programmieren einfache Spiele (Animationen) mit Html5

Name: _____ Abgabedatum: _____ Note: _____

Html 5 ist ideal um Spiele oder Animationen zu programmieren. Diese Mini-Einführung dient dazu, einige wichtige Elemente und Strategien aufzuzeigen. Weitere Hinweise für solche Programmieraufgaben finden sich auf: www.w3schools.com bzw. www.w3resource.com/html5-canvas/ Für 34,95 Euro erhält man auch ein Fachbuch zu diesem Thema: HTML5-Spiele-Entwicklung

Eure Programmieraufgabe: Es soll eine einfache Breakout-Variante programmiert werden. Das Grundgerüst ist schon vorhanden. Eure konkreten Teilaufgaben sind:

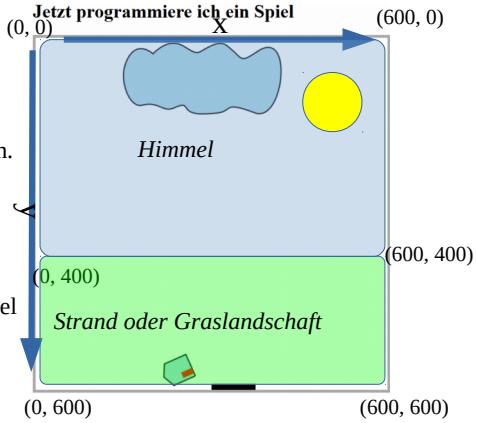
Aufgabe 1: Statt des Balls soll ein Haus durch die Gegend fliegen. Das Haus soll auch noch rotieren. Das Haus soll eine Tür und 3 Fenster haben.

Aufgabe 2: Eine Wolke soll auch im Spielverlauf erscheinen und sich über dem ‚Haus' mitbewegen. Das Programmfragment findet ihr unter www.meudela.de/Informatik/Wolke

Aufgabe 3: Im Hintergrund soll eine Landschaft mit einer Sonne, Himmel und Graslandschaft erstellt werden (Kreise bzw. Rechtecke mit Farbverläufen). Alternativ ist ein Mond und ein Sternenhimmel denkbar.

Aufgabe 4: Programmiere ein Highscore, trifft das Paddle, wird der Score um 150 Punke erhöht.

Aufgabe 5: Bringe ein Hinweis bei Spielende. Evtl. zeichne weitere Elemente dazu.



Bewertung:

Aufgabe 1: ____

Aufgabe 2: ____

Aufgabe 3: ____

Aufgabe 4: ____

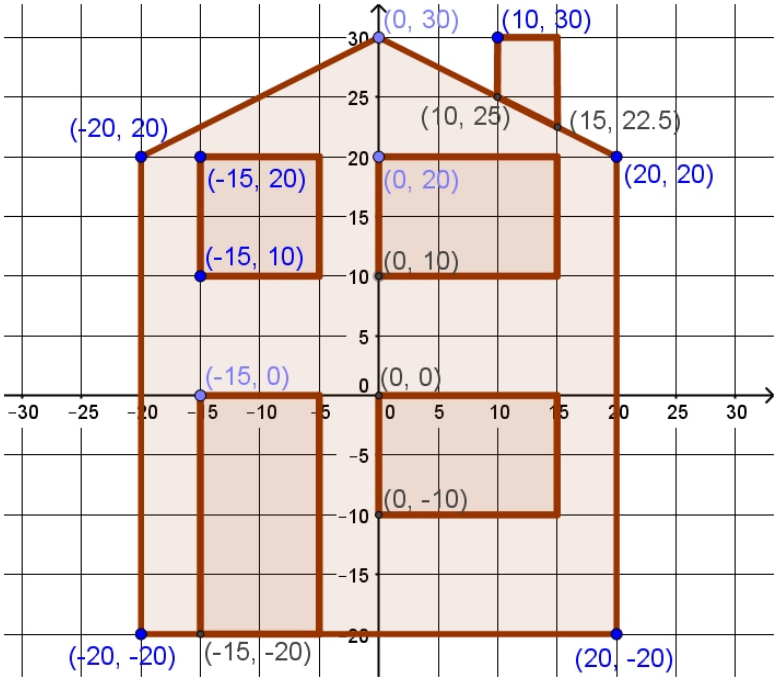
Aufgabe 5: ____

Zusatz: _____

Voraussetzungen für die Lösung der Aufgaben:
Es gibt ein Html5 Programmgerüst, welches bereits die Leinwand einrichtet und einen Ball fliegen lässt. www.meudela.de/Informatik/Grundspiel. In diesem Programmgerüst wird bereits das Spielfeld (canvas) erstellt und ein einfachstes Breakout-Spiel erstellt. Auch das Haus ist schon vorhanden und dreht sich.

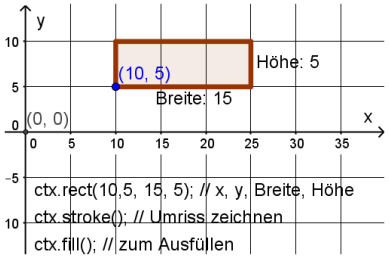
```
<!DOCTYPE html>
<html>
<body>
<h1>Jetzt programmiere ich ein Spiel</h1>
<canvas id="myCanvas" width="600" height="600" style="border:5px solid #aaaaaa;">
Your browser does not support the HTML5 canvas tag.</canvas>
<script src="jquery-2.1.1.min.js"></script>
<script>
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");

var x = 150; // Startposition x für den Ball
var y = 150; // Startposition y für den Ball
var dx = 2; // Startgeschwindigkeit des Balls in x-Richtung
var dy = 4; // Startgeschwindigkeit des Balls in y-Richtung

var WIDTH=600, HEIGHT=600; // Größe des Canvas
var paddlex=WIDTH/2; // x-Koordinate des Paddles
var paddleh = 10; // Paddlehöhe
var paddlew = 75; // Paddlebreite
var intervalId = 0;
// Gefüllte Kreise zeichnen mit Mittelpunkt (x,y) und dem Radius r
function circle(x,y,r) {
  ctx.beginPath();
  ctx.arc(x, y, r, 0, Math.PI*2, true);
  ctx.closePath();
  ctx.fill();
}

// Gefüllte Rechtecke zeichnen mit Mittelpunkt (x,y) und dem Radius r
function rect(x,y,w,h) {
  ctx.beginPath();
  ctx.rect(x,y,w,h);
  ctx.closePath();
  ctx.fill();
}

function clear() {ctx.clearRect(0, 0, WIDTH, HEIGHT); } // Canvasfläche löschen

function init() {  // Zu Beginn wird die Animation gestartet
  intervalId = setInterval(draw, 10); // Alle 10 ms wird nun draw aufgerufen.
  return intervalId;
}
var rightDown = false;
var leftDown = false;
//set rightDown or leftDown if the right or left keys are down
function onKeyDown(evt) {
  if (evt.keyCode == 39) rightDown = true;
  else if (evt.keyCode == 37) leftDown = true;
}
//and unset them when the right or left key is released
function onKeyUp(evt) {
  if (evt.keyCode == 39) rightDown = false;
  else if (evt.keyCode == 37) leftDown = false;
}
$(document).keydown(onKeyDown);
$(document).keyup(onKeyUp);

function draw() { // Diese Funktion wird ab Spielbeginn alle 10ms aufgerufen
  clear();  // Canvas löschen
  // ***************************************
  // Hier kommen all deine Ergänzungen hin.
  // ***************************************
  // Du programmierst: Himmel zeichnen (mit Farbverlauf) (Aufgabe 3)
  // Du programmierst: eine schöne Sonne am Himmel
  // Du programmierst: eine Graslandschaft unten

  // Zeichne den Ball
  //circle(x, y, 10);
  // oder zeichne ein sich drehendes Haus. (Ergänze mit Aufgabe 1)
  ctx.save();
  ctx.lineWidth = 2;
// Rotierendes Objekt in mathematischen Koordinaten zeichnen
  ctx.translate(x,y);   ctx.scale(1,-1); // y-Koordinaten gehen mit positiven Werten nach oben
  ctx.rotate(x * Math.PI / 180);
  ctx.beginPath();
  ctx.moveTo(-20,20);
  ctx.lineTo(-20,-20); ctx.lineTo(-20,20);ctx.lineTo(0,30);ctx.lineTo(20,20);
  ctx.lineTo(20,-20);ctx.lineTo(-20,-20);
  ctx.fillStyle = 'lightblue';
  ctx.fill();
  ctx.stroke();
  // Die Tür nun zeichnen
  ctx.fillStyle = 'red';
  rect(-15,-20,10,20);
// Du programmierst: Fenster 1, Fenster 2, Fenster 3

  ctx.restore(); // ursprüngliches Koordinatensystem wieder herstellen (positive y gehen nach unten)

// Aufgabe 3: Ab hier die Wolke programmieren

  //Richtungstasten: Paddle nach rechts oder links bewegen
  if (rightDown) paddlex += 5;
  else if (leftDown) paddlex -= 5;
  rect(paddlex, HEIGHT-paddleh, paddlew, paddleh); // Paddle zeichnen

  if (x + dx > WIDTH || x + dx < 0) dx = -dx; // Aufgabe 4 hier ergänzen

  if (y + dy < 0)  dy = -dy;
  else if (y + dy > HEIGHT) {
    if (x > paddlex && x < paddlex + paddlew)
      dy = -dy;
    else {
      clearInterval(intervalId);  // Animation stoppen
                  // Aufgabe 5: Hier das Spielende programmieren
                  }
  }
  x += dx;  y += dy;
}
init();
</script>
</body>
</html>
```

Hinweise zu Aufgabe 1: Das Haus ist mit 3 Fenstern zu versehen (Rechteck, z.B. in der Farbe blau). Damit dies gelingt, hilft das folgende Koordinatensystem. Auch der Schornstein fehlt noch. Übrigens wurde das Haus zunächst mit einem Linienzug, ‚einem Path, gezeichnet: **ctx.beginPath(); ctx.moveTo(x,y); ctx.lineTo(x2,y2); ctx.lineTo(x3,y3); ...** und schließlich mit **ctx.fill();** ausgefüllt und mit **ctx.stroke();** die Umrisse gezeichnet.



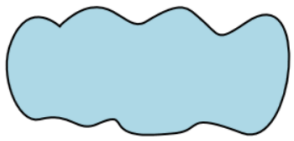
ctx.save(); speichert die aktuellen Leinwandparameter (canvas ab).

Durch **ctx.scale(1, -1);** wird dafür gesorgt, dass das übliche mathematische Koordinatensystem hier eingeschaltet wird. Das heißt: **Positive y-Werte gehen auch nach oben, das gilt auch für Rechtecke** (also die x/y-Koordinate ist nun links unten)**!**



Am Ende steht **ctx.restore();** und die vorigen Leinwandparameter sind wieder aktuell.

Ein wichtiges Spielelement ist **`ctx.translate(x,y);`** Dieser Befehl verschiebt das Haus um x-Pixel nach rechts und y Pixel nach unten.
Das Haus wird gedreht durch: **`ctx.rotate(x * Math.PI / 180);`** Dabei ist **x der Drehwinkel** (x=360 entspricht einer Volldrehung). **Das Zentrum der Rotation ist die Koordinate (0, 0)!**



Hinweise zu Aufgabe 2: Das Programmschnipsel für die Wolke:

```
// #Wolke zeichnen
        ctx.save();
        ctx.transform(1, 0, 0, 1, -50, -50);
        ctx.beginPath();
        ctx.strokeStyle = 'black';
        ctx.lineWidth = 2;
        ctx.fillStyle = 'lightblue';
        ctx.moveTo(104, 77);
        ctx.bezierCurveTo(151, 33, 165, 86, 186, 73);
        ctx.bezierCurveTo(222,51,226,60,238,70);
        ctx.bezierCurveTo(257, 86, 253, 93, 281, 74);
        ctx.bezierCurveTo(345, 29, 328, 196, 278, 178);
        ctx.bezierCurveTo(223, 157, 240, 175, 215, 178);
        ctx.bezierCurveTo(215, 178, 188, 181, 178, 179);

        ctx.bezierCurveTo(151, 175, 169, 157, 141, 169);
        ctx.bezierCurveTo(120, 179, 116, 157, 85, 165);
        ctx.bezierCurveTo(64, 170, 45, 122, 58, 92);
        ctx.bezierCurveTo(75, 49, 105, 78, 105, 78);
        ctx.fill();
        ctx.stroke();
        ctx.restore();
```

Hinweise zu Aufgabe 3: Verwendet die typischen Rechteckfiguren und Kreisfiguren. Farbverläufe sind wichtig, da sie das ganze Spiel auflockern.

`ctx.scale(x,y);` vergrößert und verkleinert Objekte. x gilt für die x-Richtung, y für die y-Richtung. Werte größer als eins vergrößern, Werte zwischen 0 und 1 verkleinern die Objekte. Für uns wichtig: Damit das Koordinatensystem mit positiven y-Werten nach oben geht ist ctx.scale(1, -1) notwendig.

Wenn man wieder eine Verschiebung (translate), eine Rotation oder eine Größenänderung aufheben, so empfiehlt sich der Einsatz von ctx.save(); und ctx.restore();. ctx.save() merkt zu Beginn sich den Grundzustand (keine Verschiebung, keine Rotation, normale Größe). Dann kann man für einige Objekte diese Bewegungen anwenden und schließlich mit ctx.restore(); wieder den Grundzustand einstellen.

Hinweis zu Aufgabe 4: Geschweifte Klammern { } sind notwendig. Verwende eine Variable mit dem Namen Score. Der Score wird mit

```
ctx.font = "30px Arial"; ctx.fillText("Punkte: "+Score,10,50);
```

angezeigt.

Überlege und setze um: Was kannst du noch in dein Programm einbauten? Realisere insbesondere ein anderes drehendes Objekt: einen Roboter, ein Ufo, einen Smiley ☺?