```
<html>
<div class="container">
  <canvas id="game"></canvas>
  <div id="introduction">
    <p>Main Balon Terbang</p>
    <p>Tahan mouse klik kiri untuk terbang</p>
  </div>
  <button id="restart">MAINLAGI</button>
</div>

<a id="youtube" target="_top" href="https://inf.mankps.sch.id">
  <span>Belajar Membuat Game Bersama Pak bihann</span>
</a>
<div id="youtube-card">
  Informatika itu menyenangkan
</div>

<style>>
html,
body {
  height: 100%;
  margin: 0;
}

body {
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  cursor: pointer;
}

.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100%;
}

#introduction {
  width: 200px;
  height: 150px;
  position: absolute;
  font-weight: 600;
  font-size: 0.8em;
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  text-align: center;
  transition: opacity 2s;
}

#restart {
```

```
  width: 120px;
  height: 120px;
  position: absolute;
  border-radius: 50%;
  color: white;
  background-color: red;
  border: none;
  font-weight: 700;
  font-size: 1.2em;
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  display: none;
  cursor: pointer;
}

#youtube,
#youtube-card {
  display: none;
  color: black;
}

@media (min-height: 425px) {
  /** Youtube logo by https://codepen.io/alvaromontoro */
  #youtube {
    z-index: 50;
    width: 100px;
    display: block;
    height: 70px;
    position: fixed;
    bottom: 20px;
    right: 20px;
    background: red;
    border-radius: 50% / 11%;
    transform: scale(0.8);
    transition: transform 0.5s;
  }

  #youtube:hover,
  #youtube:focus {
    transform: scale(0.9);
    color: black;
  }

  #youtube::before {
    content: "";
    display: block;
    position: absolute;
    top: 7.5%;
    left: -6%;
```

```
    width: 112%;
    height: 85%;
    background: red;
    border-radius: 9% / 50%;
  }

  #youtube::after {
    content: "";
    display: block;
    position: absolute;
    top: 20px;
    left: 40px;
    width: 45px;
    height: 30px;
    border: 15px solid transparent;
    box-sizing: border-box;
    border-left: 30px solid white;
  }

  #youtube span {
    font-size: 0;
    position: absolute;
    width: 0;
    height: 0;
    overflow: hidden;
  }

  #youtube:hover + #youtube-card {
    z-index: 49;
    display: block;
    position: fixed;
    bottom: 12px;
    right: 10px;
    padding: 25px 130px 25px 25px;
    width: 300px;
    background-color: white;
  }
}

</style>

<script>
/*



IT Komputer MAN Kapuas
*/
```

```
// Game data
let gameStarted; // Boolean

let balloonX;
let balloonY;

let verticalVelocity; // Current vertical velocity of the balloon
let horizontalVelocity; // Current horizontal velocity of the balloon

let fuel; // Percentage of fuel left
let heating; // Boolean: Is the mouse down or not?

let trees; // Metadata of the trees in an array
let backgroundTrees; // Metadata of the trees on the hills in the background

// Configuration
const mainAreaWidth = 400;
const mainAreaHeight = 375;
let horizontalPadding = (window.innerWidth - mainAreaWidth) / 2;
let verticalPadding = (window.innerHeight - mainAreaHeight) / 2;

const hill1BaseHeight = 80;
const hill1Speed = 0.2;
const hill1Amplitude = 10;
const hill1Stretch = 1;
const hill2BaseHeight = 50;
const hill2Speed = 0.2;
const hill2Amplitude = 15;
const hill2Stretch = 0.5;
const hill3BaseHeight = 15;
const hill3Speed = 1;
const hill3Amplitude = 10;
const hill3Stretch = 0.2;

const canvas = document.getElementById("game");
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;

const ctx = canvas.getContext("2d");

const introductionElement = document.getElementById("introduction");
const restartButton = document.getElementById("restart");

// Add a custom sin function that takes degrees instead of radians
Math.sinus = function (degree) {
  return Math.sin((degree / 180) * Math.PI);
};
```

```
// Initialize layout
resetGame();

// Resets game variables and layouts but does not start the game (game starts on keypress)
function resetGame() {
  // Reset game progress
  gameStarted = false;
  heating = false;
  verticalVelocity = 5;
  horizontalVelocity = 5;
  balloonX = 0;
  balloonY = 0;
  fuel = 100;

  introductionElement.style.opacity = 1;
  restartButton.style.display = "none";

  trees = [];
  for (let i = 1; i < window.innerWidth / 50; i++) generateTree();

  backgroundTrees = [];
  for (let i = 1; i < window.innerWidth / 30; i++) generateBackgroundTree();

  draw();
}

function generateBackgroundTree() {
  const minimumGap = 30;
  const maximumGap = 150;

  // X coordinate of the right edge of the furthest tree
  const lastTree = backgroundTrees[backgroundTrees.length - 1];
  let furthestX = lastTree ? lastTree.x : 0;

  const x =
    furthestX +
    minimumGap +
    Math.floor(Math.random() * (maximumGap - minimumGap));

  const treeColors = ["#6D8821", "#8FAC34", "#98B333"];
  const color = treeColors[Math.floor(Math.random() * 3)];

  backgroundTrees.push({ x, color });
}

function generateTree() {
  const minimumGap = 50; // Minimum distance between two trees
```

```
  const maximumGap = 600; // Maximum distance between two trees

  const x = trees.length
    ? trees[trees.length - 1].x +
      minimumGap +
      Math.floor(Math.random() * (maximumGap - minimumGap))
    : 400;

  const h = 60 + Math.random() * 80; // Height

  const r1 = 32 + Math.random() * 16; // Radius
  const r2 = 32 + Math.random() * 16;
  const r3 = 32 + Math.random() * 16;
  const r4 = 32 + Math.random() * 16;
  const r5 = 32 + Math.random() * 16;
  const r6 = 32 + Math.random() * 16;
  const r7 = 32 + Math.random() * 16;

  const treeColors = ["#6D8821", "#8FAC34", "#98B333"];
  const color = treeColors[Math.floor(Math.random() * 3)];

  trees.push({ x, h, r1, r2, r3, r4, r5, r6, r7, color });
}

resetGame();

// If space was pressed restart the game
window.addEventListener("keydown", function (event) {
  if (event.key == " ") {
    event.preventDefault();
    resetGame();
    return;
  }
});

window.addEventListener("mousedown", function () {
  heating = true;

  if (!gameStarted) {
    introductionElement.style.opacity = 0;
    gameStarted = true;
    window.requestAnimationFrame(animate);
  }
});

window.addEventListener("mouseup", function () {
  heating = false;
});
```

```
window.addEventListener("resize", function () {
  canvas.width = window.innerWidth;
  canvas.height = window.innerHeight;
  horizontalPadding = (window.innerWidth - mainAreaWidth) / 2;
  verticalPadding = (window.innerHeight - mainAreaHeight) / 2;
  draw();
});

// The main game loop
function animate() {
  if (!gameStarted) return;

  const velocityChangeWhileHeating = 0.4;
  const velocityChangeWhileCooling = 0.2;

  if (heating && fuel > 0) {
    if (verticalVelocity > -8) {
      // Limit maximum rising spead
      verticalVelocity -= velocityChangeWhileHeating;
    }
    fuel -= 0.002 * -balloonY;
  } else if (verticalVelocity < 5) {
    // Limit maximum descending spead
    verticalVelocity += velocityChangeWhileCooling;
  }

  balloonY += verticalVelocity; // Move the balloon up or down
  if (balloonY > 0) balloonY = 0; // The balloon landed on the ground
  if (balloonY < 0) balloonX += horizontalVelocity; // Move balloon to the right if not on the
ground

  // If a tree moves out of the picture replace it with a new one
  if (trees[0].x - (balloonX - horizontalPadding) < -100) {
    trees.shift(); // Remove first item in array
    generateTree(); // Add a new item to the array
  }

  // If a tree on the background hill moves out of the picture replace it with a new one
  if (
    backgroundTrees[0].x - (balloonX * hill1Speed - horizontalPadding) <
    -40
  ) {
    backgroundTrees.shift(); // Remove first item in array
    generateBackgroundTree(); // Add a new item to the array
  }

  draw(); // Re-render the whole scene
```

```
  // If the balloon hit a tree OR ran out of fuel and landed then stop the game
  const hit = hitDetection();
  if (hit || (fuel <= 0 && balloonY >= 0)) {
    restartButton.style.display = "block";
    return;
  }

  window.requestAnimationFrame(animate);
}

function draw() {
  ctx.clearRect(0, 0, window.innerWidth, window.innerHeight);

  drawSky(); // Fill the background with a gradient

  ctx.save();
  ctx.translate(0, verticalPadding + mainAreaHeight);
  drawBackgroundHills();

  ctx.translate(horizontalPadding, 0);

  // Center main canvas area to the middle of the screen
  ctx.translate(-balloonX, 0);

  // Draw scene
  drawTrees();
  drawBalloon();

  // Restore transformation
  ctx.restore();

  // Header is last because it's on top of everything else
  drawHeader();
}

restartButton.addEventListener("click", function (event) {
  event.preventDefault();
  resetGame();
  restartButton.style.display = "none";
});

function drawCircle(cx, cy, radius) {
  ctx.beginPath();
  ctx.arc(cx, cy, radius, 0, 2 * Math.PI);
  ctx.fill();
}
```

```
function drawTrees() {
  trees.forEach(({ x, h, r1, r2, r3, r4, r5, r6, r7, color }) => {
    ctx.save();
    ctx.translate(x, 0);

    // Trunk
    const trunkWidth = 40;
    ctx.fillStyle = "#885F37";
    ctx.beginPath();
    ctx.moveTo(-trunkWidth / 2, 0);
    ctx.quadraticCurveTo(-trunkWidth / 4, -h / 2, -trunkWidth / 2, -h);
    ctx.lineTo(trunkWidth / 2, -h);
    ctx.quadraticCurveTo(trunkWidth / 4, -h / 2, trunkWidth / 2, 0);
    ctx.closePath();
    ctx.fill();

    // Crown
    ctx.fillStyle = color;
    drawCircle(-20, -h - 15, r1);
    drawCircle(-30, -h - 25, r2);
    drawCircle(-20, -h - 35, r3);
    drawCircle(0, -h - 45, r4);
    drawCircle(20, -h - 35, r5);
    drawCircle(30, -h - 25, r6);
    drawCircle(20, -h - 15, r7);

    ctx.restore();
  });
}

function drawBalloon() {
  ctx.save();

  ctx.translate(balloonX, balloonY);

  // Cart
  ctx.fillStyle = "#DB504A";
  ctx.fillRect(-30, -40, 60, 10);
  ctx.fillStyle = "#EA9E8D";
  ctx.fillRect(-30, -30, 60, 30);

  // Cables
  ctx.strokeStyle = "#D62828";
  ctx.lineWidth = 2;
  ctx.beginPath();
  ctx.moveTo(-24, -40);
  ctx.lineTo(-24, -60);
  ctx.moveTo(24, -40);
```

```
  ctx.lineTo(24, -60);
  ctx.stroke();

  // Balloon
  ctx.fillStyle = "#fcf7f7";
  ctx.beginPath();
  ctx.moveTo(-30, -60);
  ctx.quadraticCurveTo(-80, -120, -80, -160);
  ctx.arc(0, -160, 80, Math.PI, 0, false);
  ctx.quadraticCurveTo(80, -120, 30, -60);
  ctx.closePath();
  ctx.fill();

  ctx.restore();
}

function drawHeader() {
  // Fuel meter
  ctx.strokeStyle = fuel <= 30 ? "red" : "white";
  ctx.strokeRect(30, 30, 150, 30);
  ctx.fillStyle = fuel <= 30 ? "rgba(255,0,0,0.5)" : "rgba(150,150,200,0.5)";
  ctx.fillRect(30, 30, (150 * fuel) / 100, 30);

  // Score
  const score = Math.floor(balloonX / 30);
  ctx.fillStyle = "black";
  ctx.font = "bold 32px Tahoma";
  ctx.textAlign = "end";
  ctx.textBaseline = "top";
  ctx.fillText(`${score} m`, window.innerWidth - 30, 30);
}

function drawSky() {
  var gradient = ctx.createLinearGradient(0, 0, 0, window.innerHeight);
  gradient.addColorStop(0, "#AADBEA");
  gradient.addColorStop(1, "#FEF1E1");
  ctx.fillStyle = gradient;
  ctx.fillRect(0, 0, window.innerWidth, window.innerHeight);
}

function drawBackgroundHills() {
  // Draw hills
  drawHill(
    hill1BaseHeight,
    hill1Speed,
    hill1Amplitude,
    hill1Stretch,
    "#AAD155" // #95C629"
```

```
  );
  drawHill(
    hill2BaseHeight,
    hill2Speed,
    hill2Amplitude,
    hill2Stretch,
    "#84B249" // "#659F1C"
  );

  drawHill(
    hill3BaseHeight,
    hill3Speed,
    hill3Amplitude,
    hill3Stretch,
    "#26532B"
  );

  // Draw background trees
  backgroundTrees.forEach((tree) => drawBackgroundTree(tree.x, tree.color));
}

// A hill is a shape under a stretched out sinus wave
function drawHill(baseHeight, speedMultiplier, amplitude, stretch, color) {
  ctx.beginPath();
  ctx.moveTo(0, window.innerHeight);
  ctx.lineTo(0, getHillY(0, baseHeight, amplitude, stretch));
  for (let i = 0; i <= window.innerWidth; i++) {
    ctx.lineTo(i, getHillY(i, baseHeight, speedMultiplier, amplitude, stretch));
  }
  ctx.lineTo(window.innerWidth, window.innerHeight);
  ctx.fillStyle = color;
  ctx.fill();
}

function drawBackgroundTree(x, color) {
  ctx.save();
  ctx.translate(
    (-balloonX * hill1Speed + x) * hill1Stretch,
    getTreeY(x, hill1BaseHeight, hill1Amplitude)
  );

  const treeTrunkHeight = 5;
  const treeTrunkWidth = 2;
  const treeCrownHeight = 25;
  const treeCrownWidth = 10;

  // Draw trunk
  ctx.fillStyle = "#7D833C";
```

```
  ctx.fillRect(
    -treeTrunkWidth / 2,
    -treeTrunkHeight,
    treeTrunkWidth,
    treeTrunkHeight
  );

  // Draw crown
  ctx.beginPath();
  ctx.moveTo(-treeCrownWidth / 2, -treeTrunkHeight);
  ctx.lineTo(0, -(treeTrunkHeight + treeCrownHeight));
  ctx.lineTo(treeCrownWidth / 2, -treeTrunkHeight);
  ctx.fillStyle = color;
  ctx.fill();

  ctx.restore();
}

function getHillY(x, baseHeight, speedMultiplier, amplitude, stretch) {
  const sineBaseY = -baseHeight;
  return (
    Math.sinus((balloonX * speedMultiplier + x) * stretch) * amplitude +
    sineBaseY
  );
}

function getTreeY(x, baseHeight, amplitude) {
  const sineBaseY = -baseHeight;
  return Math.sinus(x) * amplitude + sineBaseY;
}

function hitDetection() {
  const cartBottomLeft = { x: balloonX - 30, y: balloonY };
  const cartBottomRight = { x: balloonX + 30, y: balloonY };
  const cartTopRight = { x: balloonX + 30, y: balloonY - 40 };

  for (const { x, h, r1, r2, r3, r4, r5 } of trees) {
    const treeBottomLeft = { x: x - 20, y: -h - 15 };
    const treeLeft = { x: x - 30, y: -h - 25 };
    const treeTopLeft = { x: x - 20, y: -h - 35 };
    const treeTop = { x: x, y: -h - 45 };
    const treeTopRight = { x: x + 20, y: -h - 35 };

    if (getDistance(cartBottomLeft, treeBottomLeft) < r1) return true;
    if (getDistance(cartBottomRight, treeBottomLeft) < r1) return true;
    if (getDistance(cartTopRight, treeBottomLeft) < r1) return true;

    if (getDistance(cartBottomLeft, treeLeft) < r2) return true;
```

```
    if (getDistance(cartBottomRight, treeLeft) < r2) return true;
    if (getDistance(cartTopRight, treeLeft) < r2) return true;

    if (getDistance(cartBottomLeft, treeTopLeft) < r3) return true;
    if (getDistance(cartBottomRight, treeTopLeft) < r3) return true;
    if (getDistance(cartTopRight, treeTopLeft) < r3) return true;

    if (getDistance(cartBottomLeft, treeTop) < r4) return true;
    if (getDistance(cartBottomRight, treeTop) < r4) return true;
    if (getDistance(cartTopRight, treeTop) < r4) return true;

    if (getDistance(cartBottomLeft, treeTopRight) < r5) return true;
    if (getDistance(cartBottomRight, treeTopRight) < r5) return true;
    if (getDistance(cartTopRight, treeTopRight) < r5) return true;
  }
}

function getDistance(point1, point2) {
  return Math.sqrt((point2.x - point1.x) ** 2 + (point2.y - point1.y) ** 2);
}
</scrip>>
</html>
```