Govt.ITI Kadma

कौशलं बलम्

(CH.DADRI)

Session :- 2024-2025

Projects File :- Car Racing Game In Html

Submitted To: -

Mr.Vinod Kumar
Computer
Instructor

Submitted By :-

Jyoti , Anju , Sonika , Tarun Kumar , Nitesh , Sachin

DECLARATION

We (Jyoti , Anju , Sonika , Tarun Kumar , Nitesh , Sachin) are student of **COPA Trade** at **Government ITI Kadma**, hereby declare that the project titled **"Car Racing Game"** is my original work and has been prepared as part of the curriculum requirements for the **COPA Traders (Ncvt)** course. This project is submitted for educational purposes only.

I confirm that all the information, data, and content used in this project have been duly referenced and credited to their respective sources. No part of this project has been copied or reproduced from any other source without proper acknowledgment.

This project has been completed under the guidance of ***Sh. Vinod Kumar***, and I am solely responsible for any errors or omissions in this work.

I also declare that this project has not been submitted elsewhere for any other purpose.

Student Name: Jyoti , Anju , Sonika , Tarun Kumar , Nitesh , Sachin
COPA Traders (Ncvt), Government ITI Kadma
Signature :- Jyoti , Anju , Sonika , Tarun Kumar , Nitesh , Sachin
Date: _____

Supervisor/Instructor Declaration

This project has been reviewed and approved as part of the academic requirements for the **COPA Trade (Ncvt)** course at **Government ITI Kadma**.

Instructor Name: SH .Vinod Kumar
Signature: _____
Date: _____

Acknowledgement

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this project report for the COPA (Computer Operator and Programming Assistant) trade at Government ITI, Kadma. This project would not have been possible without the guidance, support, and encouragement of numerous individuals and organizations.

First and foremost, we extend our heartfelt thanks to the **Government ITI, Kadma**, for providing us with the opportunity to undertake this project and for offering the necessary resources and infrastructure. We are deeply indebted to our esteemed **Principal** and **Instructors** for their constant guidance, valuable insights, and encouragement throughout the project.

We would also like to thank our **Project Guide** for their unwavering support, patience, and expert advice, which helped us navigate through the challenges and complexities of this project. Their constructive feedback and suggestions were instrumental in shaping the final outcome.

Our sincere appreciation goes to our **classmates and peers** for their cooperation, motivation, and shared knowledge, which made this journey enjoyable and enriching. Special thanks to the **non-teaching staff** of the institute for their assistance and support in various aspects of the project.

Lastly, we are grateful to our **family and friends** for their constant encouragement and belief in our abilities, which kept us motivated throughout the project.

This project has been a significant learning experience, and we are thankful to everyone who played a role, directly or indirectly, in its successful completion.

Thank You.

Jyoti , Anju , Sonika , Tarun Kumar , Nitesh , Sachin /Group No.;-3
Government ITI, Kadma
Date :-

Program Code :-

```
<!DOCTYPE html>
 <html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Dash</title>
 <link rel="icon" href="icon.png">
    <link
      rel="stylesheet"

href="https://fonts.googleapis.com/css?family=Press+Start+2P"
    />
    <style>
      body {
        background: #222;
        font-family: "Press Start 2P", monospace;
        font-smooth: never;
        height: 98vh;
      }
```

```
/* UI */
.topUI {
  position: absolute;
  z-index: 1000; /* need this cause clip-path changes stack context */
  transform: translate(-50%, 25px);
  text-shadow: -2px 0 black, 0 2px black, 2px 0 black, 0 -2px black;
  letter-spacing: 2px;
  color: #fff;
  font-size: 17px;
}
.topUI::before {
  display: inline-block;
  height: 17px;
  padding: 1px 2px;
  line-height: 19px;
  font-size: 17px;
  background: #fff;
```

```
  text-shadow: none;
  font-weight: 900;
  letter-spacing: 0;
  border-radius: 6px;
  margin-right: 30px;
  border: 2px solid #7dd8c9;
}
#time {
  left: 13%;
  color: #f4f430;
}
#time::before {
  content: "TIME";
  color: #f57214;
}
#score {
  left: 45%;
}
#score::before {
```

```
  content: "SCORE";
  color: #a61a9d;
}
#lap {
  left: 88%;
  width: 45%;
}
#lap::before {
  content: "LAP";
  color: #0082df;
}
#tacho {
  position: absolute;
  text-align: right;
  width: 23%;
  bottom: 5%;
  z-index: 2000;
  color: #e62e13;
```

```
      text-shadow: -2px 0 black, 0 2px black, 2px 0 black, 0 -2px
black;
      letter-spacing: 2px;
      font-size: 23px;
    }
    #tacho::after {
      content: "km/h";
      color: #fab453;
      font-size: 18px;
      margin-left: 5px;
    }
    /*
 road
*/
    #game {
      position: relative;
      margin: 0 auto;
      overflow: hidden;
      background: #222;
```

```
  user-select: none;
  transition: opacity 10s;
}
#road {
  transition: opacity 2s;
  transition-timing-function: steps(8, end);
}
#road * {
  position: absolute;
  image-rendering: pixelated;
}
#hero {
  background-repeat: no-repeat;
  background-position: -110px 0;
  z-index: 2000;
  transform: scale(1.4);
}
#cloud {
  background-size: auto 100%;
```

```
      width: 100%;
      height: 57%;
    }
    /*
 home
*/
    #road {
      position: absolute;
      width: 100%;
      height: 100%;
    }
    #home {
      position: absolute;
      color: #fff;
      width: 100%;
      height: 100%;
      z-index: 1000; /* need this cause clip-path changes stack context */
    }
```

```
#highscore {
  position: absolute;
  width: 100%;
  height: 20%;
  bottom: 0;
  column-count: 3;
  column-fill: auto;
}
#highscore * {
  color: #9e95a8;
  margin: 0 0 6px 27px;
}
h1 {
  position: absolute;
  left: 50%;
  top: 25%;
  transform: translate(-50%, -50%);
  font-size: 5em;
  background: -webkit-linear-gradient(#25d8b1, #e2bbf0);
```

```
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
#text {
  position: absolute;
  left: 50%;
  top: 50%;
  transform: translate(-50%, -50%);
  font-size: 1.2em;
  color: #d9bbf3;
  text-shadow: 0 0 black, 0 2px black, 2px 0 black, 0 0 black;
}
.blink {
  animation: blinker 2s steps(4, end) infinite;
}
@keyframes blinker {
  50% {
    opacity: 0;
  }
```

```
    }
    /*
 Guide
*/
    #controls {
      color: #868686;
      font-size: 13px;
      line-height: 13px;
      margin: 10px;
      text-align: center;
    }
    #controls > span {
      margin-left: 20px;
    }
    #controls > span > span {
      border: 2px solid #868686;
      border-radius: 5px;
      padding: 7px;
      margin-right: 10px;
```

```
    display: inline-block;
   }
   #controls > span:last-child > span {
    transform: rotate(90deg);
   }
  </style>
  <script>
   window.console = window.console || function (t) {};
  </script>
  <script>
   if (document.location.search.match(/type=embed/gi)) {
    window.parent.postMessage("resize", "*");
   }
  </script>
</head>
<body translate="no">
  <div id="game">
   <div id="road">
    <div id="cloud"></div>
```

```
    <div id="hero"></div>
  </div>
  <div id="hud">
    <span id="time" class="topUI">0</span>
    <span id="score" class="topUI">0</span>
    <span id="lap" class="topUI">0'00"000</span>
    <span id="tacho">0</span>
  </div>
  <div id="home">
    <h1>Dash</h1>
    <p id="text"></p>
    <div id="highscore"></div>
  </div>
</div>
<div id="controls">
  <span><span>C</span>insert coin</span>
  <span><span>M</span>mute</span>
  <span><span>&lt;</span><span>&gt;</span>move</span>
```

```
<span><span>&lt;</span><span>&gt;</span>accelerate</span>
  </div>
  <script src="javascript/1.js"></script>
  <script id="rendered-js">
    // ------------------------------------------------------
    // assets
    // ------------------------------------------------------
    const ASSETS = {
      COLOR: {
        TAR: ["#959298", "#9c9a9d"],
        RUMBLE: ["#959298", "#f5f2f6"],
        GRASS: ["#eedccd", "#e6d4c5"],
      },
      IMAGE: {
        TREE: {
          src: "images/tree.png",
          width: 132,
          height: 192,
```

```
},
HERO: {
  src: "images/hero.png",
  width: 110,
  height: 56,
},
CAR: {
  src: "images/car04.png",
  width: 50,
  height: 36,
},
FINISH: {
  src: "images/finish.png",
  width: 339,
  height: 180,
  offset: -0.5,
},
SKY: {
  src: "images/cloud.jpg",
```

```
    },
  },
  AUDIO: {
    theme:

"https://s3-us-west-
2.amazonaws.com/s.cdpn.io/155629/theme.mp3",
    engine:

"https://s3-us-west-
2.amazonaws.com/s.cdpn.io/155629/engine.wav",
    honk:
"https://s3-us-west-
2.amazonaws.com/s.cdpn.io/155629/honk.wav",
    beep:
"https://s3-us-west-
2.amazonaws.com/s.cdpn.io/155629/beep.wav",
  },
};
// ------------------------------------------------------
```

```
// helper functions
// ------------------------------------------------------
Number.prototype.pad = function (numZeros, char = 0) {
  let n = Math.abs(this);
  let zeros = Math.max(0, numZeros - Math.floor(n).toString().length);
  let zeroString = Math.pow(10, zeros)
    .toString()
    .substr(1)
    .replace(0, char);
  return zeroString + n;
};
Number.prototype.clamp = function (min, max) {
  return Math.max(min, Math.min(this, max));
};
const timestamp = (_) => new Date().getTime();
const accelerate = (v, accel, dt) => v + accel * dt;
const isCollide = (x1, w1, x2, w2) => (x1 - x2) ** 2 <= (w2 + w1) ** 2;
```

```
function getRand(min, max) {
  return (Math.random() * (max - min) + min) | 0;
}
function randomProperty(obj) {
  let keys = Object.keys(obj);
  return obj[keys[(keys.length * Math.random()) << 0]];
}
function drawQuad(element, layer, color, x1, y1, w1, x2, y2,
w2) {
  element.style.zIndex = layer;
  element.style.background = color;
  element.style.top = y2 + `px`;
  element.style.left = x1 - w1 / 2 - w1 + `px`;
  element.style.width = w1 * 3 + `px`;
  element.style.height = y1 - y2 + `px`;
  let leftOffset = w1 + x2 - x1 + Math.abs(w2 / 2 - w1 / 2);
  element.style.clipPath = `polygon(${leftOffset}px 0, ${
    leftOffset + w2
  }px 0, 66.66% 100%, 33.33% 100%)`;
```

```
}
const KEYS = {};
const keyUpdate = (e) => {
  KEYS[e.code] = e.type === `keydown`;
  e.preventDefault();
};
addEventListener(`keydown`, keyUpdate);
addEventListener(`keyup`, keyUpdate);
function sleep(ms) {
  return new Promise(function (resolve, reject) {
    setTimeout((_) => resolve(), ms);
  });
}
// ------------------------------------------------
// objects
// ------------------------------------------------
class Line {
  constructor() {
    this.x = 0;
```

```
        this.y = 0;
        this.z = 0;
        this.X = 0;
        this.Y = 0;
        this.W = 0;
        this.curve = 0;
        this.scale = 0;
        this.elements = [];
        this.special = null;
      }
      project(camX, camY, camZ) {
        this.scale = camD / (this.z - camZ);
        this.X = (1 + this.scale * (this.x - camX)) * halfWidth;
        this.Y = Math.ceil(((1 - this.scale * (this.y - camY)) * height
/ 2);
        this.W = this.scale * roadW * halfWidth;
      }
      clearSprites() {
```

```
        for (let e of this.elements) e.style.background =
"transparent";
      }
      drawSprite(depth, layer, sprite, offset) {
        let destX = this.X + this.scale * halfWidth * offset;
        let destY = this.Y + 4;
        let destW = (sprite.width * this.W) / 265;
        let destH = (sprite.height * this.W) / 265;
        destX += destW * offset;
        destY += destH * -1;
        let obj = layer instanceof Element ? layer :
this.elements[layer + 6];
        obj.style.background = `url('${sprite.src}') no-repeat`;
        obj.style.backgroundSize = `${destW}px ${destH}px`;
        obj.style.left = destX + `px`;
        obj.style.top = destY + `px`;
        obj.style.width = destW + `px`;
        obj.style.height = destH + `px`;
        obj.style.zIndex = depth;
```

```
  }
}
class Car {
  constructor(pos, type, lane) {
    this.pos = pos;
    this.type = type;
    this.lane = lane;
    var element = document.createElement("div");
    road.appendChild(element);
    this.element = element;
  }
}
class Audio {
  constructor() {
    this.audioCtx = new AudioContext();
    // volume
    this.destination = this.audioCtx.createGain();
    this.volume = 1;
    this.destination.connect(this.audioCtx.destination);
```

```
  this.files = {};
  let _self = this;
  this.load(ASSETS.AUDIO.theme, "theme", function (key) {
    let source = _self.audioCtx.createBufferSource();
    source.buffer = _self.files[key];
    let gainNode = _self.audioCtx.createGain();
    gainNode.gain.value = 0.6;
    source.connect(gainNode);
    gainNode.connect(_self.destination);
    source.loop = true;
    source.start(0);
  });
}
get volume() {
  return this.destination.gain.value;
}
set volume(level) {
  this.destination.gain.value = level;
}
```

```
play(key, pitch) {
  if (this.files[key]) {
    let source = this.audioCtx.createBufferSource();
    source.buffer = this.files[key];
    source.connect(this.destination);
    if (pitch) source.detune.value = pitch;
    source.start(0);
  } else this.load(key, () => this.play(key));
}
load(src, key, callback) {
  let _self = this;
  let request = new XMLHttpRequest();
  request.open("GET", src, true);
  request.responseType = "arraybuffer";
  request.onload = function () {
    _self.audioCtx.decodeAudioData(
      request.response,
      function (beatportBuffer) {
        _self.files[key] = beatportBuffer;
```

```
        callback(key);
      },
      function () {}
    );
  };
  request.send();
 }
}
// ---------------------------------------------------
// global varriables
// ---------------------------------------------------
const highscores = [];
const width = 800;
const halfWidth = width / 2;
const height = 500;
const roadW = 4000;
const segL = 200;
const camD = 0.2;
const H = 1500;
```

```
const N = 70;
const maxSpeed = 200;
const accel = 38;
const breaking = -80;
const decel = -40;
const maxOffSpeed = 40;
const offDecel = -70;
const enemy_speed = 8;
const hitSpeed = 20;
const LANE = {
  A: -2.3,
  B: -0.5,
  C: 1.2,
};
const mapLength = 15000;
// loop
let then = timestamp();
const targetFrameRate = 1000 / 25; // in ms
let audio;
```

```
// game
let inGame,
  start,
  playerX,
  speed,
  scoreVal,
  pos,
  cloudOffset,
  sectionProg,
  mapIndex,
  countDown;
let lines = [];
let cars = [];
// ------------------------------------------------
// map
// ------------------------------------------------
function getFun(val) {
  return (i) => val;
}
```

```
function genMap() {
  let map = [];
  for (var i = 0; i < mapLength; i += getRand(0, 50)) {
    let section = {
      from: i,
      to: (i = i + getRand(300, 600)),
    };
    let randHeight = getRand(-5, 5);
    let randCurve = getRand(5, 30) * (Math.random() >= 0.5 ?
1 : -1);
    let randInterval = getRand(20, 40);
    if (Math.random() > 0.9)
      Object.assign(section, {
        curve: (_) => randCurve,
        height: (_) => randHeight,
      });
    else if (Math.random() > 0.8)
      Object.assign(section, {
        curve: (_) => 0,
```

```
      height: (i) => Math.sin(i / randInterval) * 1000,
    });
  else if (Math.random() > 0.8)
    Object.assign(section, {
      curve: (_) => 0,
      height: (_) => randHeight,
    });
  else
    Object.assign(section, {
      curve: (_) => randCurve,
      height: (_) => 0,
    });
  map.push(section);
}
map.push({
  from: i,
  to: i + N,
  curve: (_) => 0,
  height: (_) => 0,
```

```
    special: ASSETS.IMAGE.FINISH,
  });
  map.push({ from: Infinity });
  return map;
}
let map = genMap();
// ------------------------------------------------------
// additional controls
// ------------------------------------------------------
addEventListener(`keyup`, function (e) {
  if (e.code === "KeyM") {
    e.preventDefault();
    audio.volume = audio.volume === 0 ? 1 : 0;
    return;
  }
  if (e.code === "KeyC") {
    e.preventDefault();
    if (inGame) return;
    sleep(0)
```

```
.then((_) => {
  text.classList.remove("blink");
  text.innerText = 3;
  audio.play("beep");
  return sleep(1000);
})
.then((_) => {
  text.innerText = 2;
  audio.play("beep");
  return sleep(1000);
})
.then((_) => {
  reset();
  home.style.display = "none";
  road.style.opacity = 1;
  hero.style.display = "block";
  hud.style.display = "block";
  audio.play("beep", 500);
  inGame = true;
```

```
    });
  return;
 }
 if (e.code === "Escape") {
  e.preventDefault();
  reset();
 }
});
// ------------------------------------------------------
// game loop
// ------------------------------------------------------
function update(step) {
 // prepare this iteration
 pos += speed;
 while (pos >= N * segL) pos -= N * segL;
 while (pos < 0) pos += N * segL;
 var startPos = (pos / segL) | 0;
 let endPos = (startPos + N - 1) % N;
 scoreVal += speed * step;
```

```
countDown -= step;
// left / right position
playerX -= (lines[startPos].curve / 5000) * step * speed;
if (KEYS.ArrowRight)
  (hero.style.backgroundPosition = "-220px 0"),
    (playerX += 0.007 * step * speed);
else if (KEYS.ArrowLeft)
  (hero.style.backgroundPosition = "0 0"),
    (playerX -= 0.007 * step * speed);
else hero.style.backgroundPosition = "-110px 0";
playerX = playerX.clamp(-3, 3);
// speed
if (inGame && KEYS.ArrowUp) speed = accelerate(speed,
accel, step);
else if (KEYS.ArrowDown) speed = accelerate(speed,
breaking, step);
else speed = accelerate(speed, decel, step);
if (Math.abs(playerX) > 0.55 && speed >= maxOffSpeed) {
  speed = accelerate(speed, offDecel, step);
```

```
}
speed = speed.clamp(0, maxSpeed);
// update map
let current = map[mapIndex];
let use = current.from < scoreVal && current.to > scoreVal;
if (use) sectionProg += speed * step;
lines[endPos].curve = use ? current.curve(sectionProg) : 0;
lines[endPos].y = use ? current.height(sectionProg) : 0;
lines[endPos].special = null;
if (current.to <= scoreVal) {
  mapIndex++;
  sectionProg = 0;
  lines[endPos].special = map[mapIndex].special;
}
// win / lose + UI
if (!inGame) {
  speed = accelerate(speed, breaking, step);
  speed = speed.clamp(0, maxSpeed);
} else if (countDown <= 0 || lines[startPos].special) {
```

```
      tacho.style.display = "none";
      home.style.display = "block";
      road.style.opacity = 0.4;
      text.innerText = "INSERT COIN";
      highscores.push(lap.innerText);
      highscores.sort();
      updateHighscore();
      inGame = false;
    } else {
      time.innerText = (countDown | 0).pad(3);
      score.innerText = (scoreVal | 0).pad(8);
      tacho.innerText = speed | 0;
      let cT = new Date(timestamp() - start);
      lap.innerText =
`${cT.getMinutes()}'${cT.getSeconds().pad(2)}"${cT
        .getMilliseconds()
        .pad(3)}`;
    }
    // sound
```

```
    if (speed > 0) audio.play("engine", speed * 4);
    // draw cloud
    cloud.style.backgroundPosition = `${
      (cloudOffset -= lines[startPos].curve * step * speed * 0.13)
| 0
    }px 0`;
    // other cars
    for (let car of cars) {
      car.pos = (car.pos + enemy_speed * step) % N;
      // respawn
      if ((car.pos | 0) === endPos) {
        if (speed < 30) car.pos = startPos;
        else car.pos = endPos - 2;
        car.lane = randomProperty(LANE);
      }
      // collision
      const offsetRatio = 5;
      if (
        (car.pos | 0) === startPos &&
```

```
        isCollide(playerX * offsetRatio + LANE.B, 0.5, car.lane,
0.5)
      ) {
        speed = Math.min(hitSpeed, speed);
        if (inGame) audio.play("honk");
      }
    }
    // draw road
    let maxy = height;
    let camH = H + lines[startPos].y;
    let x = 0;
    let dx = 0;
    for (let n = startPos; n < startPos + N; n++) {
      let l = lines[n % N];
      let level = N * 2 - n;
      // update view
      l.project(
        playerX * roadW - x,
        camH,
```

```
      startPos * segL - (n >= N ? N * segL : 0)
    );
    x += dx;
    dx += l.curve;
    // clear assets
    l.clearSprites();
    // first draw section assets
    if (n % 10 === 0) l.drawSprite(level, 0, ASSETS.IMAGE.TREE,
-2);
    if ((n + 5) % 10 === 0)
      l.drawSprite(level, 0, ASSETS.IMAGE.TREE, 1.3);
    if (l.special)
      l.drawSprite(level, 0, l.special, l.special.offset || 0);
    for (let car of cars)
      if ((car.pos | 0) === n % N)
        l.drawSprite(level, car.element, car.type, car.lane);
    // update road
    if (l.Y >= maxy) continue;
    maxy = l.Y;
```

```
let even = ((n / 2) | 0) % 2;
let grass = ASSETS.COLOR.GRASS[even * 1];
let rumble = ASSETS.COLOR.RUMBLE[even * 1];
let tar = ASSETS.COLOR.TAR[even * 1];
let p = lines[(n - 1) % N];
drawQuad(
  l.elements[0],
  level,
  grass,
  width / 4,
  p.Y,
  halfWidth + 2,
  width / 4,
  l.Y,
  halfWidth
);
drawQuad(
  l.elements[1],
  level,
```

```
  grass,
  (width / 4) * 3,
  p.Y,
  halfWidth + 2,
  (width / 4) * 3,
  l.Y,
  halfWidth
);
drawQuad(
  l.elements[2],
  level,
  rumble,
  p.X,
  p.Y,
  p.W * 1.15,
  l.X,
  l.Y,
  l.W * 1.15
);
```

```
        drawQuad(l.elements[3], level, tar, p.X, p.Y, p.W, l.X, l.Y,
l.W);
        if (!even) {
          drawQuad(
            l.elements[4],
            level,
            ASSETS.COLOR.RUMBLE[1],
            p.X,
            p.Y,
            p.W * 0.4,
            l.X,
            l.Y,
            l.W * 0.4
          );
          drawQuad(
            l.elements[5],
            level,
            tar,
            p.X,
```

```
        p.Y,
        p.W * 0.35,
        l.X,
        l.Y,
        l.W * 0.35
      );
    }
  }
}
// ------------------------------------------------------
// init
// ------------------------------------------------------
function reset() {
  inGame = false;
  start = timestamp();
  countDown = map[map.length - 2].to / 130 + 10;
  playerX = 0;
  speed = 0;
  scoreVal = 0;
```

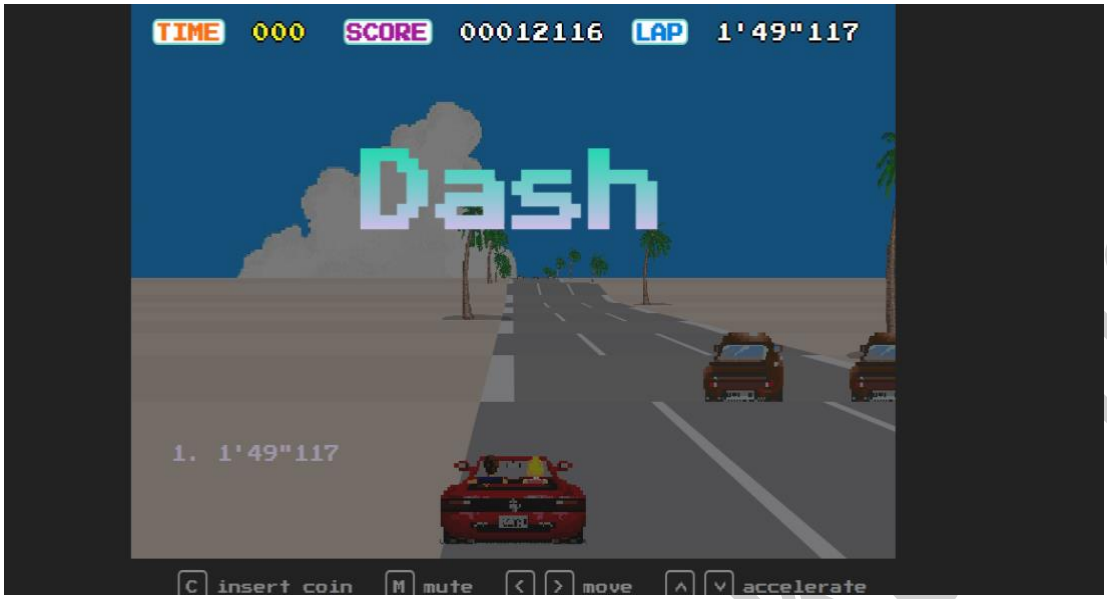
```
    pos = 0;
    cloudOffset = 0;
    sectionProg = 0;
    mapIndex = 0;
    for (let line of lines) line.curve = line.y = 0;
    text.innerText = "INSERT COIN";
    text.classList.add("blink");
    road.style.opacity = 0.4;
    hud.style.display = "none";
    home.style.display = "block";
    tacho.style.display = "block";
  }
  function updateHighscore() {
    let hN = Math.min(12, highscores.length);
    for (let i = 0; i < hN; i++) {
      highscore.children[i].innerHTML = `${(i + 1).pad(2,
" ")}. ${
        highscores[i]
      }`;
```

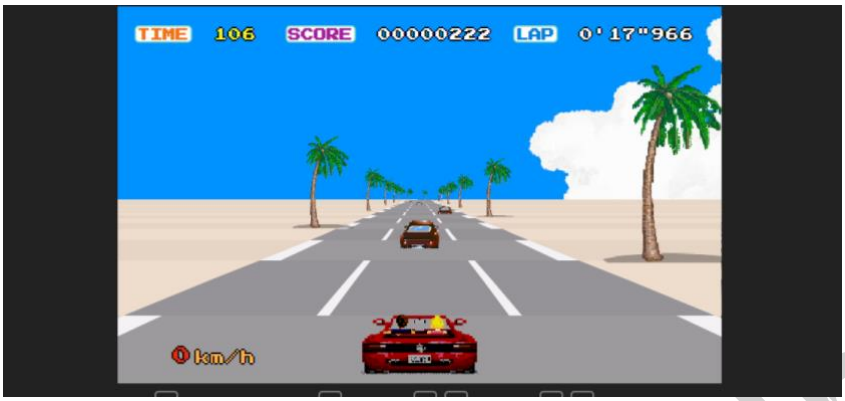
```
    }
  }
  function init() {
    game.style.width = width + "px";
    game.style.height = height + "px";
    hero.style.top = height - 80 + "px";
    hero.style.left = halfWidth - ASSETS.IMAGE.HERO.width / 2
+ "px";
    hero.style.background = `url(${ASSETS.IMAGE.HERO.src})`;
    hero.style.width = `${ASSETS.IMAGE.HERO.width}px`;
    hero.style.height = `${ASSETS.IMAGE.HERO.height}px`;
    cloud.style.backgroundImage =
`url(${ASSETS.IMAGE.SKY.src})`;
    audio = new Audio();
    Object.keys(ASSETS.AUDIO).forEach((key) =>
      audio.load(ASSETS.AUDIO[key], key, (_) => 0)
    );
    cars.push(new Car(0, ASSETS.IMAGE.CAR, LANE.C));
    cars.push(new Car(10, ASSETS.IMAGE.CAR, LANE.B));
```

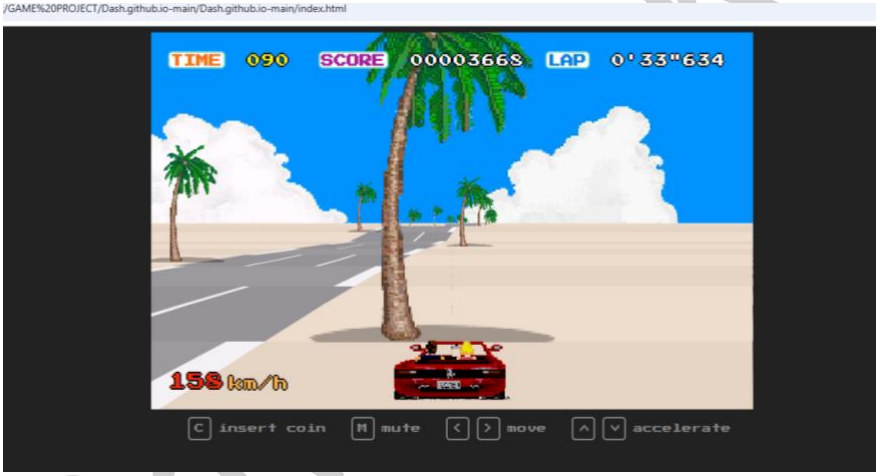
```
cars.push(new Car(20, ASSETS.IMAGE.CAR, LANE.C));
cars.push(new Car(35, ASSETS.IMAGE.CAR, LANE.C));
cars.push(new Car(50, ASSETS.IMAGE.CAR, LANE.A));
cars.push(new Car(60, ASSETS.IMAGE.CAR, LANE.B));
cars.push(new Car(70, ASSETS.IMAGE.CAR, LANE.A));
for (let i = 0; i < N; i++) {
  var line = new Line();
  line.z = i * segL + 270;
  for (let j = 0; j < 6 + 2; j++) {
    var element = document.createElement("div");
    road.appendChild(element);
    line.elements.push(element);
  }
  lines.push(line);
}
for (let i = 0; i < 12; i++) {
  var element = document.createElement("p");
  highscore.appendChild(element);
}
```

```
    updateHighscore();
    reset();
    // START GAME LOOP
    (function loop() {
      requestAnimationFrame(loop);
      let now = timestamp();
      let delta = now - then;
      if (delta > targetFrameRate) {
        then = now - (delta % targetFrameRate);
        update(delta / 1000);
      }
    })();
  }
  init();
  //# sourceURL=pen.js
</script>
<script src="javascript/null.js"></script>
</body>
</html>
```

OUTPUT




TIME 106 SCORE 00000222 LAP 0'17"966
0 km/h


/GAME%20PROJECT/Dash.github.io-main/Dash.github.io-main/index.html
TIME 090 SCORE 00003668 LAP 0'33"634
158 km/h
C insert coin M mute move accelerate