MODUL KELAS INDUSTRI – SNAKE GAME

1. Buat folder snake-typescript
2. Buka folder menggunakan VS Code lalu buka terminal
3. Install Vite Typescript: npm create vite@latest . (jangan lupa [titik] agar tidak membuat sub-folder)
   - o Select a framework:
   - | **Vanilla**
   - o Select a variant:
   - | **TypeScript**
   - o Install with npm and start now?
   - | **Yes**

4. Edit index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Snake Game - Logika Pemrograman</title>
  <style>
    body {
      background-color: #0f172a;
      color: #f8fafc;
      font-family: system-ui, -apple-system, sans-serif;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      height: 100vh;
      margin: 0;
      overflow: hidden; /* Mencegah scroll total pada halaman */
    }

    h1 {
      margin: 0 0 4px 0;
      font-size: 1.25rem;
      font-weight: 600;
    }

    canvas {
      border: 4px solid #334155;
      border-radius: 8px;
      background-color: #1e293b;
      box-shadow: 0 10px 25px -5px rgba(0, 0, 0, 0.3);
    }
  </style>
</head>
<body>

  <h1>Snake Game</h1>
  <div id="scoreBox">Skor: 0</div>

  <!-- Canvas Game -->
  <canvas id="gameCanvas" width="400" height="400"></canvas>

  <script type="module" src="/src/main.ts"></script>
</body>
</html>
```

5. Edit main.ts

A. Koordinat dan Canvas

```
// A1. Data Koordinat
interface Point {
  x: number;  // Koordinat X
  y: number;  // Koordinat Y
}

// A2. Grid dan Canvas
const CANVAS_SIZE = 400;                        // Ukuran canvas
const GRID_SIZE = 20;                           // Ukuran tiap kotak
const TILE_COUNT = CANVAS_SIZE / GRID_SIZE;     // Jumlah kotak

const canvas = document.getElementById("gameCanvas") as HTMLCanvasElement;
const ctx = canvas.getContext("2d")!;
```

B. Menggambar Ular dan Fungsi Utama

```
// B1. State awal ular
let ular: Point[] = [
  { x: 10, y: 10 },   // Index 0  Kepala
  { x: 9, y: 10 },    // Index 1  Badan
  { x: 8, y: 10 },    // Index 2  Ekor
];

// B2. Gambar Ular
function gambarUlar(): void {
  // Gambar ular
  ular.forEach((segment, index) => {
    // Kepala ular dibedakan warnanya agar jelas (opsional untuk visual)
    ctx.fillStyle = index === 0 ? "#e63930ff" : "#22c55e";

    // Menggambar kotak berdasarkan koordinat grid dikalikan ukuran tile
    ctx.fillRect(
      segment.x * GRID_SIZE,
      segment.y * GRID_SIZE,
      GRID_SIZE - 2, // dikurangi 2 agar ada efek garis antar kotak
      GRID_SIZE - 2,
    );
  });
}

// Z1. Fungsi Utama (game loop)
function main() {
  // Bersihkan canvar
  ctx.fillStyle = '#222';
  ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

  gambarUlar();
}

// Z2. Menjalankan game loop setiap 200 milidetik
setInterval(main, 200);
```

C. Arah Gerak Ular

```
// C1. Arah gerak awal (dx: sumbu X, dy: sumbu Y)
let dx: number = 1;
let dy: number = 0;

// C2. Gerak Ular
function gerakUlar(): void {
  // Buat kepala baru berdasarkan posisi kepala saat ini ditambahkan arah (dx, dy)
  const kepala: Point = {
    x: ular[0].x + dx,
    y: ular[0].y + dy
  };

  // SISIPKAN kepala baru di awal array (unshift)
  ular.unshift(kepala);

  // HAPUS elemen terakhir array (pop)
  ular.pop();
}
```

```
// Z1. Fungsi Utama (game loop)
function main() {
  // Bersihkan layar
  ctx.fillStyle = '#1e293b';
  ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

  gerakUlar();       // Panggil fungsi gerak ular
  gambarUlar();
}
```

D. Input Keyboard

```
// D. Input Keybord
window.addEventListener('keydown', (e: KeyboardEvent) => {
  // Mencegah scroll default browser saat menekan tombol panah
  if (['ArrowUp', 'ArrowDown', 'ArrowLeft', 'ArrowRight', ' '].includes(e.key)) {
    e.preventDefault();
  }

  const keyPressed = e.key;

  // Validasi: Cegah arah berlawanan langsung
  const keAtas = dy === -1;
  const keBawah = dy === 1;
  const keKiri = dx === 1;
  const keKanan = dx === -1;

  if (keyPressed === 'ArrowLeft' && !keKanan) {
    dx = -1;
    dy = 0;
  }
  if (keyPressed === 'ArrowUp' && !keBawah) {
    dx = 0;
    dy = -1;
  }
  if (keyPressed === 'ArrowRight' && !keKiri) {
    dx = 1;
    dy = 0;
  }
  if (keyPressed === 'ArrowDown' && !keAtas) {
    dx = 0;
    dy = 1;
  }
});
```

E. Menggambar makanan

```
// E. State Makanan
let apel: Point = { x: 5, y: 5 };

function gambarApel(): void {
  ctx.fillStyle = '#c9e413ff'; // Warna kuning untuk makanan
  ctx.fillRect(
    apel.x * GRID_SIZE,
    apel.y * GRID_SIZE,
    GRID_SIZE - 2,
    GRID_SIZE - 2
  );
}
```

```
// Z1. Fungsi Utama (game loop)
function main() {
  // Bersihkan layar
  ctx.fillStyle = '#1e293b';
  ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

  gerakUlar();
  gambarApel();     // Panggil fungsi gamber apel
  gambarUlar();
}
```

F. Acak posisi apel

```
// F. Membuat makanan di posisi acak
function acakApel(): void {
  // Menghasilkan angka acak dari 0 hingga (TILE_COUNT - 1)
  apel = {
    x: Math.floor(Math.random() * TILE_COUNT),
    y: Math.floor(Math.random() * TILE_COUNT)
  };

  // Pastikan makanan tidak muncul di atas badan ular!
  ular.forEach(segment => {
    if (segment.x === apel.x && segment.y === apel.y) {
      acakApel(); // Rekursi ulang jika menumpuk
    }
  });
}
```

G. Ular makan apel (edit bagian gerak ular)

```
// C2. Gerak Ular
function gerakUlar(): void {
  // Buat kepala baru berdasarkan posisi kepala saat ini ditambahkan arah (dx, dy)
  const kepala: Point = {
    x: ular[0].x + dx,
    y: ular[0].y + dy
  };

  // SISIPKAN kepala baru di awal array (unshift)
  ular.unshift(kepala);

  // Deteksi tabrakan kepala ular dengan makanan
  const makanApel = kepala.x === apel.x && kepala.y === apel.y;

  if (makanApel) {
    acakApel();   // Buat makanan baru
  } else {
    ular.pop();   // Hapus ekor seperti biasa jika tidak makan
  }
}
```

H. Jika menabrak dinding

```
// G. GameOver
function gameOver(): boolean {
  const kepala = ular[0];

  // Tabrak Dinding
  const tabrakKiri = kepala.x < 0;
  const tabrakKanan = kepala.x >= TILE_COUNT;
  const tabrakAtas = kepala.y < 0;
  const tabrakBawah = kepala.y >= TILE_COUNT;

  if (tabrakKiri || tabrakKanan || tabrakAtas || tabrakBawah) {
    return true;
  }

  // Tabrak Tubuh Sendiri
  for (let i = 1; i < ular.length; i++) {
    if (kepala.x === ular[i].x && kepala.y === ular[i].y) {
      return true;
    }
  }

  return false;
}
```

```
// H. Awal Skor & Kecepatan
let skor: number = 0;
let kecepatan: number = 200; // Mulai dari 200ms per frame
let waktuMain: number;
```

```
// Z1. Fungsi Utama (game loop)
function main() {
  gerakUlar();

  if (gameOver()) {
    clearTimeout(waktuMain); // Hentikan perulangan game loop

    // Tampilkan informasi Game Over di canvas atau alert sederhana
    ctx.fillStyle = "#1e293b";
    ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

    ctx.fillStyle = "#f8fafc";
    ctx.font = "bold 24px system-ui";
    ctx.textAlign = "center";
    ctx.fillText("GAME OVER", CANVAS_SIZE / 2, CANVAS_SIZE / 2);
    return;
  }

  // Render ulang jika permainan masih berjalan
  ctx.fillStyle = "#1e293b";
  ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

  gambarApel(); // Panggil fungsi gamber apel
  gambarUlar();
}
```

I. Menampilkan Skor dan Tambah Kecepatan

index.html

```
…
  <style>
…
    #scoreBox {
      font-size: 1rem;
      font-weight: bold;
      color: #4ade80;
      margin-bottom: 8px;
    }
…
  </style>
</head>
<body>
  <h1>Snake Game</h1>
  <div id="scoreBox">Skor: 0</div>

  <!-- Canvas Game -->
  <canvas id="gameCanvas" width="400" height="400"></canvas>

  <script type="module" src="/src/main.ts"></script>
</body>
</html>
```

main.ts

```
// I. Ambil elemen skor dari DOM & Update Skor
const scoreElement = document.getElementById("scoreBox") as HTMLElement;

function updateSkor(): void {
  if (scoreElement) {
    scoreElement.innerText = `Skor: ${skor}`;
  }
}
```

```
// C2. Gerak Ular
function gerakUlar(): void {
  // Buat kepala baru berdasarkan posisi kepala saat ini ditambahkan arah (dx, dy)
  const kepala: Point = {
    x: ular[0].x + dx,
    y: ular[0].y + dy,
  };

  // SISIPKAN kepala baru di awal array (unshift)
  ular.unshift(kepala);

  // Deteksi tabrakan kepala ular dengan makanan
  const makanApel = kepala.x === apel.x && kepala.y === apel.y;

  if (makanApel) {
    acakApel();
    skor += 10;
    updateSkor();

    // Logika Peningkatan Kecepatan:
    if (kecepatan > 50) {
      kecepatan -= 5;
    }
  } else {
    ular.pop();
  }
}
```

```
// Z1. Fungsi Utama (game loop)
function main() {
  gerakUlar();

  if (gameOver()) {
    clearTimeout(waktuMain); // Hentikan perulangan game loop

    // Tampilkan informasi Game Over di canvas atau alert sederhana
    ctx.fillStyle = "#1e293b";
    ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

    ctx.fillStyle = "#f8fafc";
    ctx.font = "bold 24px system-ui";
    ctx.textAlign = "center";
    ctx.fillText("GAME OVER", CANVAS_SIZE / 2, CANVAS_SIZE / 2);
    return;
  }

  // Render ulang jika permainan masih berjalan
  ctx.fillStyle = "#1e293b";
  ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

  gambarApel(); // Panggil fungsi gamber apel
  gambarUlar();

  waktuMain = setTimeout(main, kecepatan);
}

// Z2. Menjalankan game loop
main();
```

J. Tombol Main Lagi

index.html

```
…
  <style>
…
    /* Kontainer tombol dengan tinggi tetap agar tidak menggeser layout */
    .button-container {
      height: 40px; /* Tinggi konstan untuk tombol */
      margin-top: 10px;
      display: flex;
      align-items: center;
      justify-content: center;
    }

    #restartBtn {
      padding: 6px 16px;
      font-size: 0.875rem;
      font-weight: bold;
      background-color: #ef4444;
      color: white;
      border: none;
      border-radius: 6px;
      cursor: pointer;
      display: none; /* Disembunyikan di awal */
      transition: background-color 0.2s;
    }

    #restartBtn:hover {
      background-color: #dc2626;
    }
  </style>
</head>
```

```
<body>
  <h1>Snake Game</h1>
  <div id="scoreBox">Skor: 0</div>

  <!-- Canvas Game -->
  <canvas id="gameCanvas" width="400" height="400"></canvas>

  <!-- Kontainer Tombol di Bawah Canvas -->
  <div class="button-container">
    <button id="restartBtn">Main Lagi</style>
  </div>

  <script type="module" src="/src/main.ts"></script>
</body>
</html>
```

main.ts

```
// J. Tombol Main Lagi
const restartBtn = document.getElementById("restartBtn") as HTMLButtonElement;

// Fungsi untuk mereset seluruh state game ke kondisi awal
function resetGame(): void {
  // Reset posisi ular ke tengah/awal
  ular = [
    { x: 10, y: 10 },
    { x: 9, y: 10 },
    { x: 8, y: 10 },
  ];
  // Reset arah, skor, dan kecepatan
  dx = 1;
  dy = 0;
  skor = 0;
  kecepatan = 200;

  updateSkor();
  acakApel();

  // Sembunyikan tombol restart saat game berjalan kembali
  restartBtn.style.display = "none";

  // Batalkan timeout sebelumnya jika ada, lalu mulai ulang loop
  clearTimeout(waktuMain);
  main();
}

// Event listener untuk tombol restart
restartBtn.addEventListener("click", () => {
  resetGame();
});
```

```
// Z1. Fungsi Utama (game loop)
function main() {
  gerakUlar();

  if (gameOver()) {
    clearTimeout(waktuMain); // Hentikan perulangan game loop

    // Tampilkan informasi Game Over di canvas atau alert sederhana
    ctx.fillStyle = "#1e293b";
    ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

    ctx.fillStyle = "#f8fafc";
    ctx.font = "bold 24px system-ui";
    ctx.textAlign = "center";
```

```
    ctx.fillText("GAME OVER", CANVAS_SIZE / 2, CANVAS_SIZE / 2);

    restartBtn.style.display = "inline-block";

    return;
  }

  // Render ulang jika permainan masih berjalan
  ctx.fillStyle = "#1e293b";
  ctx.fillRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);

  gambarApel(); // Panggil fungsi gamber apel
  gambarUlar();

  waktuMain = setTimeout(main, kecepatan);
}
```