PROGRAMACIÓN WEB

Alumno:

Jeremi Yahir Narvaez Miranda

Josue Francisco Gonzales Molina

ING. Laura Cleofás Sánchez

Semestre: 6°

Carrera: I.S.C.

Grupo: 3601

Ciclo Escolar: Marzo 2025 – Julio 2025

Contenido

Introducción

Este proyecto HTML implementa una animación visual usando la etiqueta <canvas> para representar una escena animada compuesta por una casa, árboles, nubes, un sol animado y gallos. El objetivo es simular un entorno campestre con elementos que aparecen progresivamente, generando una experiencia visual dinámica y entretenida. Está dirigido a quienes desean practicar animaciones gráficas en 2D utilizando JavaScript puro y sin librerías externas.

Desarrollo

Estructura HTML

El archivo HTML define una estructura básica:

- La etiqueta <canvas> con ID canvas de 800x600 píxeles es el lienzo sobre el cual se dibuja toda la animación.
- El título del documento es “Casa con árboles y gallos”.

Estilos CSS

Los estilos CSS aseguran una visualización sin barras de desplazamiento (overflow: hidden) y un fondo celeste que simula el cielo:

```css
body {
  margin: 0;
  overflow: hidden;
}
canvas {
  display: block;
  background: #87CEEB;
}
```

Lógica JavaScript

El núcleo de la animación se encuentra en un bloque <script> con múltiples funciones:

Lógica general de animación

- Se define un contador animationStep que va del 0 al 13 (14 pasos en total).
- Cada paso agrega un nuevo elemento visual a la escena.
- Se utiliza requestAnimationFrame para animar en tiempo real.
- Hay un reinicio automático luego de un tiempo.

Funciones de dibujo principales

Fondo y entorno

```
function drawBackground() {
  const skyGradient = ctx.createLinearGradient(0, 0, 0, canvas.height);
  skyGradient.addColorStop(0, '#87CEEB');
  skyGradient.addColorStop(1, '#4682B4');
  ctx.fillStyle = skyGradient;
  ctx.fillRect(0, 0, canvas.width, canvas.height);

  ctx.fillStyle = '#32CD32';
  ctx.fillRect(0, canvas.height - grassHeight, canvas.width, grassHeight);

  ctx.strokeStyle = '#228B22';
  ctx.lineWidth = 2;
  for (let i = 0; i < 50; i++) {
    const x = Math.random() * canvas.width;
    const height = 10 + Math.random() * 20;
    ctx.beginPath();
    ctx.moveTo(x, canvas.height - grassHeight);
    ctx.quadraticCurveTo(
      x + 5 - Math.random() * 10,
      canvas.height - grassHeight - height,
      x,
      canvas.height - grassHeight - height
    );
    ctx.stroke();
  }
}
```

Casa

```
function drawHouseBase() {
  const houseHeight = 150;
  const progress = Math.min(1, animationStep / 1);
  const height = houseHeight * progress;

  ctx.fillStyle = '#FFD700';
  ctx.fillRect(300, canvas.height - grassHeight - height, 200, height);
}

function drawRoof() {
  const progress = Math.min(1, (animationStep - 1) / 1);
  const baseY = canvas.height - grassHeight - 150;
  const leftX = 300;
  const rightX = 500;
  const peakY = baseY - 50;
```

```
  ctx.beginPath();
  ctx.moveTo(leftX, baseY);
  ctx.lineTo(rightX, baseY);

  if (progress < 0.5) {
    const x = leftX + (rightX - leftX) * (progress / 0.5);
    ctx.lineTo(x, baseY);
  } else {
    ctx.lineTo(rightX, baseY);
    const y = baseY - (baseY - peakY) * ((progress - 0.5) / 0.5);
    ctx.lineTo((leftX + rightX) / 2, y);
  }

  ctx.closePath();
  ctx.fillStyle = '#B22222';
  ctx.fill();
  ctx.strokeStyle = '#8B0000';
  ctx.stroke();
```

Vegetación

```
function drawTree(x, baseY) {
  const trunkHeight = 80;
  const trunkWidth = 25;
  const leavesRadius = 40;

  let progress = 1;
  if (x === 100 && animationStep === 8) progress = Math.min(1, (animationStep - 7) / 1);
  if (x === 180 && animationStep === 9) progress = Math.min(1, (animationStep - 8) / 1);
  if (x === 600 && animationStep === 10) progress = Math.min(1, (animationStep - 9) / 1);
  if (x === 680 && animationStep === 11) progress = Math.min(1, (animationStep - 10) / 1);

  const trunkH = trunkHeight * progress;
  ctx.fillStyle = '#8B4513';
  ctx.fillRect(x, baseY - trunkH, trunkWidth, trunkH);
```

Animales

```
function drawRooster(x, y) {
  const time = Date.now() / 500;
  const headBob = Math.sin(time * 3) * 3;

  ctx.fillStyle = '#FF0000';
  ctx.beginPath();
  ctx.ellipse(x, y - headBob, 20, 15, 0, 0, Math.PI * 2);
  ctx.fill();

  ctx.fillStyle = '#FF4500';
  ctx.beginPath();
  ctx.arc(x + 20, y - 20 - headBob, 10, 0, Math.PI * 2);
  ctx.fill();
```

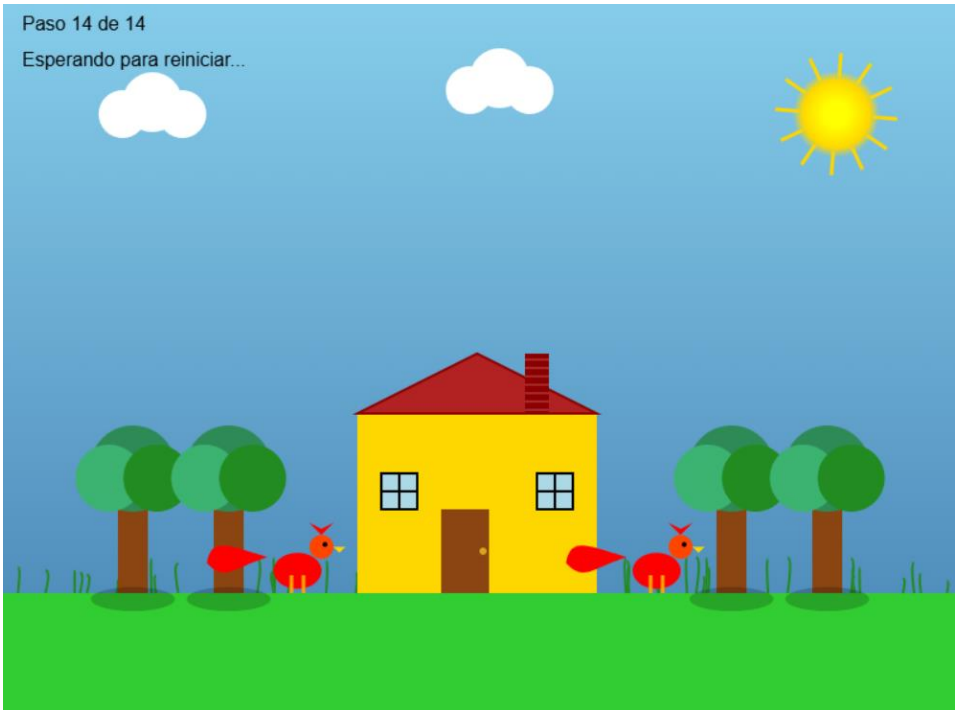
```
ctx.fillStyle = '#FF0000';
ctx.beginPath();
ctx.moveTo(x + 20, y - 30 - headBob);
ctx.lineTo(x + 30, y - 40 - headBob);
ctx.lineTo(x + 20, y - 35 - headBob);
ctx.lineTo(x + 10, y - 40 - headBob);
ctx.closePath();
ctx.fill();

ctx.fillStyle = '#FFD700';
ctx.beginPath();
ctx.moveTo(x + 30, y - 20 - headBob);
ctx.lineTo(x + 40, y - 20 - headBob);
ctx.lineTo(x + 35, y - 15 - headBob);
ctx.closePath();
ctx.fill();
```

Texto en pantalla

- drawInstructions() muestra el número de paso actual y mensajes de instrucción o espera.

Interacción del usuario

El evento keydown permite al usuario iniciar la animación si está en el paso 0. Después, el flujo se maneja automáticamente hasta el reinicio.



Conclusión

El código representa una animación escalonada bien estructurada que demuestra el uso avanzado del canvas de HTML5. El autor aplica una metodología basada en pasos para mostrar progresivamente los elementos de la escena, ofreciendo claridad y organización. Se destacan técnicas como la animación por fotogramas (requestAnimationFrame), el uso de gradientes, transformaciones y dibujo de formas complejas como curvas Bézier. Este proyecto sirve tanto como una herramienta educativa para aprender gráficos en 2D como una base para animaciones más complejas en futuros desarrollos web.