

Wisej.Web.Canvas y el modelo Canvas 2D de HTML5

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Leer la documentación de Canvas 2D del navegador y escribir el equivalente en Wisej.NET: miembros en PascalCase en Wisej.Web.Canvas, ángulos en grados donde el navegador usa radianes, Save y Restore alrededor de las transformaciones, degradados, patrones, sombras y recortes, y Redraw como el único lugar donde se reconstruye una escena. Qué significa esto para un desarrollador .NET con experiencia que ya construye pantallas Wisej.NET y ahora necesita gráficos personalizados, y cómo leer este módulo.



Enfoque de producción

Leer la documentación de Canvas 2D del navegador y escribir el equivalente en Wisej.NET: miembros en PascalCase en Wisej.Web.Canvas, ángulos en grados donde el navegador usa radianes, Save y Restore alrededor de las transformaciones, degradados, patrones, sombras y recortes, y Redraw como el único lugar donde se reconstruye una escena.

1. El problema en producción

Todo lo que VisualOperationsStudio ha dibujado hasta ahora se producía en el servidor: el indicador del Módulo 2 y las celdas Health y Trend del Módulo 3 se renderizan con `System.Drawing` y llegan al usuario como imágenes. La vista de topología que la aplicación necesita a continuación son decenas de nodos, aristas y etiquetas pequeños que el usuario acabará desplazando, ampliando y seleccionando, que es justamente el trabajo que un canvas de navegador hace bien.

`Wisej.Web.Canvas` te entrega esa superficie desde C# y te cobra a cambio un modo de fallo distinto. Los píxeles pertenecen al navegador, nada en el servidor los recuerda y un redimensionado puede borrarlos. El código que dibuja una sola vez se ve perfecto en el diseñador y pierde su escena la primera vez que un usuario arrastra el borde de una ventana.

2. Conceptos clave

Concepto	Por qué importa
<code>Wisej.Web.Canvas</code>	Un control de Wisej que expone a C# la superficie 2D de modo inmediato del navegador; sus llamadas son comandos de dibujo que ejecuta el navegador, no operaciones de <code>System.Drawing.Graphics</code> .
Evento <code>Redraw</code>	El enganche que dispara la superficie cuando necesita otra vez su contenido; como Wisej.NET no conserva los comandos, el controlador tiene que reconstruir la escena entera.
Correspondencia en PascalCase	Los miembros se corresponden por mayúsculas y minúsculas — <code>fillRect()</code> con <code>FillRect()</code> , <code>strokeStyle</code> con <code>StrokeStyle</code> —, mientras que los valores CSS se convierten en tipos .NET como <code>Color</code> , <code>Rectangle</code> , <code>Font</code> e <code>Image</code> .
Ángulos en grados	Wisej.NET documenta los argumentos de ángulo en grados por simplicidad allí donde el navegador usa radianes, lo que cambia todas las llamadas trasladadas a <code>Arc()</code> y <code>Rotate()</code> .
<code>Save()</code> / <code>Restore()</code>	Apilan y desapilan todo el estado de renderizado —estilos, transformación, recorte, alfa, ajustes de línea y de texto— para que un cambio local no se filtre a la forma siguiente.
<code>TextFont</code> , <code>TextAlign</code> , <code>TextBaseline</code>	El sustituto en Wisej.NET de <code>font</code> , <code>textAlign</code> y <code>textBaseline</code> del navegador; la fuente es un <code>System.Drawing.Font</code> real.
<code>LiveUpdate</code>	False de forma predeterminada, así que los comandos viajan con la actualización normal de la interfaz de la petición; true envía cada llamada de inmediato, lo que encaja con una revelación progresiva y arruina un bucle de animación.

3. Desarrollo de la lección

Lee la documentación del navegador y después escribe el C#

El Canvas de Wisej.NET es una fachada programable desde el servidor sobre el contexto 2D del navegador, así que se aprende traduciendo ejemplos. La mayor parte es mecánica: `beginPath()` pasa a ser `BeginPath()`, `globalAlpha` pasa a ser `GlobalAlpha`, `setLineDash()` pasa a ser `SetLineDash()`. Donde el navegador espera una cadena CSS, Wisej.NET espera un valor .NET: un color es un `Color`, una fuente es una `Font` y una imagen es un `Image` o una cadena de origen de imagen.

La mitad que una referencia del navegador no puede contarte es que la lista de miembros documentada de Wisej.NET no es toda la API moderna de Canvas. `measureText`, `getImageData`, `roundRect`, `ellipse` y `createConicGradient` no aparecen en ella. Traducir incluye, por tanto, un paso adicional: confirmar que el miembro existe en la versión contra la que compilas. Cuando no existe, la respuesta honesta es un Widget o JavaScript en el cliente, nunca un nombre de método adivinado.

Redraw es el único lugar donde existe tu escena

Wisej.NET no conserva las llamadas de dibujo de Canvas. Los comandos se emiten, el navegador los pinta y la participación del servidor termina ahí. Cuando la superficie se redimensiona o se actualiza, el mapa de bits puede quedar limpio, y no existe ninguna copia en caché que lo devuelva. Ese único hecho separa el código de Canvas que funciona del que solo parece funcionar.

Así que la escena vive en objetos C#, un método la reconstruye por completo y `Redraw` llama a ese método. En el laboratorio, `DrawPlayground` empieza con un `ClearRect` sobre toda la superficie y vuelve a emitir todos los comandos, que es por lo que da el mismo resultado en la primera visualización, tras un cambio de valor y tras un redimensionado.

```
private void canvasPlayground_Redraw(object sender, EventArgs e) => DrawPlayground();

private void DrawPlayground()
{
    canvasPlayground.ClearRect(0, 0, canvasPlayground.Width, canvasPlayground.Height);

    canvasPlayground.Save();
    canvasPlayground.Translate(240, 160);
    canvasPlayground.Rotate(45); // browser: ctx.rotate(Math.PI / 4)
    canvasPlayground.FillStyle = Color.Coral;
    canvasPlayground.FillRect(-60, -24, 120, 48);
    canvasPlayground.Restore();
}
```

Dos hábitos rompen esto. Dibujar en `Load` se ejecuta una sola vez, así que la escena desaparece tras el primer redimensionado. Un controlador que solo redibuja lo que ha cambiado es peor: funciona mientras desarrollas y deja una superficie medio vacía delante de un usuario.

Trazados, rellenos, texto, degradados y sombras

Un trazado se construye primero y se pinta después. `MoveTo()` y `LineTo()` añaden geometría pero no pintan nada; `Stroke()` o `Fill()` hacen el pintado con los actuales `StrokeStyle`, `LineWidth`, `LineCap` y `LineJoin`. `BeginPath()` inicia un trazado independiente, y omitirlo es el error clásico al trasladar código: un contorno posterior incluye en silencio la forma anterior. `FillRect()`, `StrokeRect()` y `ClearRect()` actúan de inmediato, mientras que `Rect()` se añade al trazado que estás construyendo.

`CreateLinearGradient()` y `CreateRadialGradient()` devuelven objetos que asignas a `FillStyle` o a `StrokeStyle`, contruidos a partir de un array de paradas de color, y `CreatePattern()` recibe un `Image` o un origen de imagen más un `CanvasRepeatPattern`. El texto pasa por `TextFont`, `TextAlign` y `TextBaseline` antes de `FillText()`; las imágenes, por las sobrecargas de `DrawImage()`. La profundidad sale de las propiedades de sombra, de `GlobalAlpha` y de `GlobalCompositeOperation`: la composición es sutil y depende del navegador, así que prueba todo lo que llesves a producción con ella.

Save, Restore y la pila de transformaciones

El estado de renderizado de Canvas es mucho más amplio que la transformación, y `Save()` lo apila entero mientras que `Restore()` lo devuelve. `Translate()`, `Rotate()` y `Scale()` modifican la transformación

actual, `Transform()` compone otra matriz sobre ella, y `SetTransform()` restablece la identidad y aplica la que le pases.

Delimita cada nodo: guarda, posiciona, dibuja, restaura. Sin la restauración, todas las formas posteriores heredan el sistema de coordenadas desplazado y la escena se desvía de una forma exasperante de depurar. Un recorte se comporta igual, porque pasa a formar parte del estado actual, así que un recorte temporal es siempre guardar, trazado, `Clip()`, dibujar, restaurar. Un solo `Restore()` devuelve:

- `FillStyle`, `StrokeStyle` y `GlobalAlpha`
- la matriz de transformación actual y la región de recorte activa
- `LineWidth`, `LineCap`, `LineJoin` y el patrón de guiones
- los ajustes de texto y las propiedades de sombra

Grados en lugar de radianes, y cuándo `LiveUpdate` se gana su sitio

La documentación de dibujo personalizado de Wisej.NET señala una desviación deliberada: los argumentos de ángulo se expresan en grados. Un ejemplo que escribe `ctx.rotate(Math.PI / 4)` pasa a ser `Rotate(45)`, y una circunferencia que va de `0` a `Math.PI * 2` pasa a ir de `0` a `360` en `Arc()`. Copiar el valor en radianes compila perfectamente y gira la forma una fracción de grado, que es por lo que el laboratorio pide la conversión en un comentario junto a la línea trasladada.

La última diferencia documentada es la entrega. `LiveUpdate` es false de forma predeterminada, así que los comandos emitidos durante una petición viajan con la actualización de interfaz de esa petición: un viaje de ida y vuelta, un redibujado. En true, cada llamada se envía de inmediato, lo que resulta genuinamente útil cuando una operación larga en el servidor debe ir revelando sus pasos, y caro en todo lo demás. No es un camino hacia la animación dirigida por el servidor. Mantenlo desactivado en el render principal para que el coste sea algo que hayas elegido.



Regla de dibujo: primero la superficie

Elige la superficie de renderizado antes de escribir una sola llamada de dibujo. Pregúntate dónde se producen los píxeles y qué cruza la red: un controlador de Paint de un control se ejecuta en el servidor y envía una imagen, `CellPaint` hace lo mismo dentro de una celda de cuadrícula, `Canvas` envía comandos de dibujo que ejecuta el navegador y no conserva nada tras un redimensionado, y `System.Drawing.Managed` produce bytes de imagen sin ningún control de por medio. Mantén el modelo semántico en objetos C#, haz que cada renderizador sea una función determinista de ese modelo, invalida solo cuando haya cambiado una dependencia visual, y deja siempre un equivalente textual de todo lo importante que exista únicamente como píxeles.



Objetivos de aprendizaje

Al final de este módulo podrás:

- ✓ Explicar la correspondencia entre `CanvasRenderingContext2D` y `Wisej.Web.Canvas`.
- ✓ Usar `Redraw` como el enganche canónico de reconstrucción de la escena.
- ✓ Dibujar trazados, rectángulos, texto, imágenes, degradados, patrones, sombras y contenido compuesto.

- ✓ Usar Save/Restore y los métodos de transformación de forma segura.
- ✓ Entender diferencias documentadas como la nomenclatura de C# y los ángulos en grados.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Construye en VisualOperationsStudio una página de pruebas de Canvas que traslade un conjunto de ejemplos de Canvas 2D del navegador a la API documentada de Wisej.Web.Canvas. Haz que todo dependa de un método DrawPlayground invocado por el evento Redraw del Canvas, para que la escena vuelva a aparecer tras un redimensionado, y empíezalo con un ClearRect sobre toda la superficie. Dibuja un trazado con contorno, rectángulos rellenos, un rótulo colocado, un degradado lineal y un degradado radial; después envuelve cada sección transformada en Save() y Restore() y demuestra Translate, Scale, Rotate y SetTransform. Añade una región recortada, una sección con globalAlpha, una línea discontinua y una sombra, traslada un ejemplo del navegador que gira $\text{Math.PI} / 4$ al valor en grados que Wisej espera, y mantén LiveUpdate desactivado en el render principal mientras lo muestras en un pequeño ejemplo progresivo.

Punto de partida

Abre VisualOperationsStudio tal como lo dejó el Módulo 3: el control `TelemetryGauge` se dibuja en el servidor, `gridOperations` tiene las columnas Health y Trend dibujadas por el usuario, y la página del Módulo 1 sigue mostrando un valor de telemetría en cuatro superficies. Añade una nueva `CanvasPlaygroundPage` con un único `Wisej.Web.Canvas` acoplado para rellenarla, más un pequeño panel de botones; de momento nada en esa página dibuja nada.

Enfoque sugerido

- ✓ Escribe primero `DrawPlayground()` y suscríbelo a `canvasPlayground.Redraw` en el constructor. Abre el método con `ClearRect(0, 0, canvasPlayground.Width, canvasPlayground.Height)`, dibuja una polilínea con contorno usando `BeginPath`, `MoveTo`, `LineTo` y `Stroke`, ejecuta la página y redimensiona el navegador antes de añadir nada más.
- ✓ Añade la sección de rellenos: dos formas `FillRect`, un rótulo colocado con `TextFont`, `TextAlign` y `TextBaseline`, y después una barra rellena con `CreateLinearGradient` y una esfera rellena con `CreateRadialGradient`. Contrasta cada miembro con la API de Canvas de Wisej.NET antes de escribirlo.
- ✓ Construye la fila de transformaciones como cuatro bloques delimitados. Cada uno es `Save()`, después `Translate`, `Scale`, `Rotate` o `SetTransform`, después una pequeña forma etiquetada y, por último, `Restore()`. Toma un ejemplo del navegador que llame a `ctx.rotate(Math.PI / 4)`, escribe `Rotate(45)` y pon la conversión en un comentario en la misma línea.
- ✓ Añade la sección de estado: una región recortada construida como un trazado seguido de `Clip()` dentro de un par save/restore, un bloque dibujado con un `GlobalAlpha` reducido, una línea discontinua con `SetLineDash` y una tarjeta dibujada con `ShadowColor`, `ShadowBlur` y los desplazamientos de sombra. Comenta a propósito un `Restore()`, observa cómo se desplaza el resto de la escena y después devuélvelo a su sitio.

- ✓ Deja `LiveUpdate` en false para `DrawPlayground` . Dale a un botón un bucle corto en el servidor que active `LiveUpdate` , dibuje un paso de progreso por iteración y vuelva a desactivarlo; después anota la diferencia de tráfico y los casos en los que merece la pena hacer ese intercambio.

Entregables requeridos

- ✓ Área de pruebas de Canvas cuya escena completa dibuja un método `DrawPlayground` llamado desde `Redraw`
- ✓ Trazados, rectángulos, un rótulo colocado, un degradado lineal y un degradado radial usando únicamente miembros documentados de `Wisej.Web.Canvas`
- ✓ `Save()` y `Restore()` alrededor de cada sección con `Translate`, `Scale`, `Rotate` y `SetTransform`
- ✓ Recorte, alfa, una línea discontinua y una sombra, más una conversión de radianes a grados escrita en un comentario
- ✓ Una demostración de `LiveUpdate` separada del render principal, con una nota sobre cuándo es apropiado

Preguntas de repaso

- ✓ Tu área de pruebas se ve bien hasta que un compañero redimensiona el navegador, tras lo cual solo queda visible la última forma que dibujaste. ¿Qué te dice eso sobre de dónde venía el resto de la escena?
- ✓ Necesitas el ancho en píxeles de un rótulo para dimensionar una insignia a su medida. La API de Canvas de `Wisej.NET` no incluye ningún miembro de medición de texto: describe dos formas de conseguir igualmente el diseño que quieres y qué cuesta cada una.
- ✓ Quien revisa tu trabajo pregunta por qué el recorte de tu sección de estado está envuelto en `Save()` y `Restore()` , si las formas posteriores se dibujan en un sitio completamente distinto. Respóndele.



Regla de dibujo: primero la superficie

Elige la superficie de renderizado antes de escribir una sola llamada de dibujo. Pregúntate dónde se producen los píxeles y qué cruza la red: un controlador de Paint de un control se ejecuta en el servidor y envía una imagen, `CellPaint` hace lo mismo dentro de una celda de cuadrícula, Canvas envía comandos de dibujo que ejecuta el navegador y no conserva nada tras un redimensionado, y `System.Drawing.Managed` produce bytes de imagen sin ningún control de por medio. Mantén el modelo semántico en objetos C#, haz que cada renderizador sea una función determinista de ese modelo, invalida solo cuando haya cambiado una dependencia visual, y deja siempre un equivalente textual de todo lo importante que exista únicamente como píxeles.