¡Perfecto! Entendido. Es una muy buena distinción, y tienes toda la razón. No queremos que todas las cajas tengan una imagen si solo algunas la necesitan.

Tu idea de tener una imagen **junto al texto solo en algunas interacciones** es común y le da mucho juego a la narrativa. Para lograr esto, haremos una pequeña modificación en cómo tu juego maneja el currentDialogue y luego ajustaremos la lógica de dibujo.

**Cómo Funcionará (Cambios Conceptuales)**

1. **Objeto de Diálogo Completo:** En lugar de que currentDialogue solo contenga el texto (una string), haremos que sea un **objeto**. Este objeto contendrá el text, y si la interacción tiene una imagen, también contendrá la imagePath (y eventualmente, la imagen ya cargada).
2. **Carga Condicional de la Imagen (Primer Paso):** Cuando tu personaje active una interacción, el juego "verá" si ese punto de interacción tiene una imagePath definida en sus propiedades (las que pusiste en Tiled).
3. **Dibujo Condicional:** En la función drawMap (o en gameLoop donde se dibuja el diálogo), verificaremos si el currentDialogue activo tiene una imagen. Si la tiene, la dibujamos y ajustamos la posición del texto. Si no, solo dibujamos el texto como hasta ahora.

**Modificaciones en js/game.js**

**1. Abre tu archivo js/game.js.**

**2. Modifica cómo se establece currentDialogue cuando una interacción es activada:**

* Desplázate hasta la sección Lógica de interacciones dentro de gameLoop.
* Cambia la línea donde se asigna el currentDialogue para que sea un objeto.

JavaScript

// js/game.js

// ... dentro de gameLoop(timestamp) { ...

// Lógica de interacciones (solo si no hay diálogo activo y no se ha teletransportado)

const playerPolygonForInteraction = getPlayerPolygon();

if (!currentDialogue) {

for (const interactionId in mapInteractions) {

const interaction = mapInteractions[interactionId];

if (!interaction.once || !interaction.hasBeenTriggered) {

if (checkCollision(playerPolygonForInteraction, interaction.polygon)) {

// ¡MODIFICADO!: currentDialogue ahora es un OBJETO

currentDialogue = {

text: interaction.text,

imagePath: interaction.imagePath || null, // Guardamos la ruta de la imagen, si existe

// Aquí más adelante podríamos guardar la imagen ya cargada si fuera necesario

};

interaction.hasBeenTriggered = true;

break;

}

}

}

}

// ...

**3. Modifica la sección de dibujo del cuadro de diálogo (if (currentDialogue) { ... }) para que dibuje la imagen condicionalmente y ajuste el texto:**

* Busca este bloque al final de gameLoop.
* Reemplaza su contenido con el siguiente. Presta mucha atención a cómo se gestiona imagePath y cómo afecta al boxX y boxWidth del texto.

JavaScript

// js/game.js

// ... dentro de function gameLoop(timestamp) { ...

// ... (código existente antes del dibujo) ...

// --- SECCIÓN MODIFICADA!: Dibujar el Cuadro de Texto de Diálogo (Abajo y Ancho Adaptable con IMAGEN OPCIONAL) ---

if (currentDialogue) {

ctx.save(); // Guarda el estado actual del canvas

const dpr = window.devicePixelRatio || 1;

ctx.setTransform(dpr, 0, 0, dpr, 0, 0);

const cssCanvasWidth = canvas.width / dpr;

const cssCanvasHeight = canvas.height / dpr;

// 1. Dimensiones y Márgenes Responsivos del Cuadro General

const responsiveHeightRatio = 0.22;

const dialogueHeight = cssCanvasHeight \* responsiveHeightRatio;

const horizontalMargin = Math.max(15, cssCanvasWidth \* 0.02);

const boxWidth = cssCanvasWidth - (horizontalMargin \* 2);

const boxX = horizontalMargin;

const boxY = cssCanvasHeight - dialogueHeight - horizontalMargin;

const cornerRadius = 15;

// 2. Calcular el tamaño de fuente responsivo

const baseFontSize = 28;

const minFontSize = 16;

const fontSize = Math.max(minFontSize, baseFontSize \* (cssCanvasWidth / 1200));

const dismissFontSize = Math.max(12, fontSize \* 0.7);

const lineHeight = fontSize \* 1.2; // Altura de línea para el texto ajustado

// 3. Dibujar el fondo y borde del cuadro general

ctx.fillStyle = 'rgba(20, 20, 20, 0.9)';

ctx.beginPath();

ctx.roundRect(boxX, boxY, boxWidth, dialogueHeight, cornerRadius);

ctx.fill();

ctx.strokeStyle = '#ADD8E6';

ctx.lineWidth = 3;

ctx.beginPath();

ctx.roundRect(boxX, boxY, boxWidth, dialogueHeight, cornerRadius);

ctx.stroke();

// 4. Lógica para dibujar la imagen (si existe) y ajustar el área de texto

let imageToDraw = null; // Almacenará la imagen CADA VEZ

let imageDrawWidth = 0;

let imageDrawHeight = 0;

let imageMargin = 0; // Margen entre la imagen y el texto

// Si la interacción tiene una ruta de imagen...

if (currentDialogue.imagePath) {

// Aquí es donde deberíamos CARGAR la imagen si no está ya en caché.

// Por ahora, la cargamos cada vez (no ideal para rendimiento si son muchas,

// pero funcional para la prueba. Luego te explico cómo cachearlas).

imageToDraw = new Image(); // Crea una nueva imagen

imageToDraw.src = currentDialogue.imagePath; // Asigna la ruta

// Solo dibuja si la imagen ya está cargada

if (imageToDraw.complete) {

// Calcular el tamaño de la imagen proporcionalmente a la altura del diálogo

imageDrawHeight = dialogueHeight \* 0.8; // La imagen ocupará el 80% de la altura de la caja

imageDrawWidth = imageToDraw.width \* (imageDrawHeight / imageToDraw.height);

// Asegurar que la imagen no sea demasiado ancha

const maxImageWidth = boxWidth \* 0.3; // Máximo 30% del ancho de la caja

if (imageDrawWidth > maxImageWidth) {

imageDrawWidth = maxImageWidth;

imageDrawHeight = imageToDraw.height \* (imageDrawWidth / imageToDraw.width); // Recalcular altura

}

imageMargin = 15; // Margen entre la imagen y el borde de la caja/texto

const imageX = boxX + dialoguePadding; // Posición X: margen de la caja + padding

const imageY = boxY + (dialogueHeight - imageDrawHeight) / 2; // Posición Y: centrada verticalmente

ctx.drawImage(imageToDraw, imageX, imageY, imageDrawWidth, imageDrawHeight);

} else {

// Si la imagen no está cargada, podrías dibujar un placeholder o un mensaje de carga.

// Por ahora, no haremos nada si no está completa para evitar errores visuales.

}

}

// 5. Ajustar el área de texto y dibujar el texto principal

let textStartX = boxX + dialoguePadding; // Punto de inicio X del texto (por defecto, con padding)

let textMaxWidth = boxWidth - (dialoguePadding \* 2); // Ancho disponible para el texto

if (imageToDraw && imageToDraw.complete) { // Si se dibujó una imagen...

textStartX = boxX + dialoguePadding + imageDrawWidth + imageMargin; // Mueve el texto a la derecha de la imagen

textMaxWidth = boxWidth - (dialoguePadding \* 2) - imageDrawWidth - imageMargin; // Reduce el ancho disponible

}

// Centrar el texto horizontalmente dentro de su nueva área disponible

const textDrawX = textStartX + textMaxWidth / 2;

ctx.fillStyle = '#FFFFFF';

ctx.font = `${fontSize}px Arial`;

ctx.textAlign = 'center'; // Alinear al centro del área de texto

ctx.textBaseline = 'top'; // Importante para wrapText

// Calcular la altura total que ocupará el texto ajustado para centrarlo verticalmente

// (Esta es una simulación de wrapText para obtener el número de líneas sin dibujarlas aún)

const tempContext = canvas.getContext('2d'); // Usar un contexto temporal o el mismo

tempContext.font = `${fontSize}px Arial`; // Asegurarse de que la fuente sea la misma para la medición

const wordsToMeasure = currentDialogue.text.split(' ');

let tempLine = '';

let tempLinesCount = 0;

for (let n = 0; n < wordsToMeasure.length; n++) {

const testLine = tempLine + wordsToMeasure[n] + ' ';

const testWidth = tempContext.measureText(testLine).width;

if (testWidth > textMaxWidth && n > 0) { // Usar textMaxWidth aquí

tempLinesCount++;

tempLine = wordsToMeasure[n] + ' ';

} else {

tempLine = testLine;

}

}

tempLinesCount++; // Cuenta la última línea

const calculatedTextBlockHeight = tempLinesCount \* lineHeight;

// Calcular el Y inicial para centrar el bloque de texto verticalmente

const textYStart = boxY + (dialogueHeight - calculatedTextBlockHeight) / 2;

// Ahora, dibujar el texto ajustado

wrapText(ctx, currentDialogue.text, textDrawX, textYStart, textMaxWidth, lineHeight);

// 6. Dibujar el indicador de "presiona para continuar"

ctx.font = `${dismissFontSize}px Arial`;

ctx.fillStyle = '#CCCCCC';

const dismissTextY = boxY + dialogueHeight - (dialogueHeight \* 0.2);

ctx.textAlign = 'center';

ctx.textBaseline = 'middle';

ctx.fillText("Presiona ESPACIO o toca para continuar", boxX + boxWidth / 2, dismissTextY);

ctx.restore();

}

requestAnimationFrame(gameLoop);

}

// ...

**4. Guarda tu archivo js/game.js.**

**Puntos Importantes a Considerar con esta Solución:**

* **Carga de Imágenes (¡Temporalmente No Óptimo!):** En el punto 4. Lógica para dibujar la imagen (si existe), he puesto una línea imageToDraw = new Image(); imageToDraw.src = currentDialogue.imagePath;. Esto está cargando la imagen **cada vez** que se dibuja el cuadro de diálogo, lo cual es muy ineficiente y puede causar **parpadeos o retrasos** la primera vez que aparece un diálogo con una imagen.
  + **Solución Futura:** Cuando estés listo, el siguiente paso óptimo sería **precargar** estas imágenes de retrato cuando el mapa se carga (similar a cómo cargas los sprites del jugador o del teletransportador), y luego pasarlas como portraitImage: miImagenYaCargada en el objeto currentDialogue. Pero para probar que funciona, este método de carga es suficiente.
* **Ruta de las Imágenes de Retrato:** Asegúrate de que las imagePath que defines en Tiled para tus objetos de interacción sean **rutas correctas** a tus archivos de imagen (ej., img/personajes/heroe\_portrait.png).
* **Posición de la Imagen y el Texto:** Puedes ajustar los valores de imageDrawHeight, imageMargin, dialoguePadding y las posiciones X/Y para que la imagen y el texto se vean exactamente como deseas dentro del cuadro.

**Pasos Finales y CRUCIALES (¡La Recarga!):**

1. **Guarda js/game.js** con estas modificaciones.
2. **Abre index.html.**
3. **Cambia el número de versión (cache-buster)** en la línea que carga game.js a un número **NUEVO y DIFERENTE** (ej., v=74). Esto es **ABSOLUTAMENTE ESENCIAL** para que el navegador cargue la versión actualizada y no una de la caché.
   * <script type="module" src="./js/game.js?v=74"></script>
   * **Guarda index.html.**
4. **Realiza una recarga FORZADA en tu navegador (¡esto es muy importante!):**
   * Cierra **TODAS** las pestañas de tu navegador que tengan el juego abierto.
   * Detén tu servidor local.
   * Abre el navegador, abre las Herramientas de Desarrollador (F12 o Ctrl+Shift+I), ve a la pestaña Network y **MARCA la casilla Disable cache**.
   * Haz clic derecho en el botón de recargar la página y selecciona Empty Cache and Hard Reload (Vaciar caché y Recarga Forzada).
   * Inicia tu servidor y abre el juego en una nueva pestaña.

Ahora, cuando una interacción con un imagePath se active, deberías ver la imagen junto al texto, y el texto debería ajustarse para no salirse de la caja, tanto en PC como en móvil. ¡Dime cómo te va con esta nueva versión!