```
/**
 * ═══════════════════════════════════════════════════════════════════════
 * KURO :: LIQUID GLASS ENGINE — React Provider & Components
 *
 * Injects SVG displacement filters for real refraction.
 * Provides hooks and wrapper components for glass materials.
 * Apple iOS 26 HIG-compliant.
 * ═══════════════════════════════════════════════════════════════════════
 */

import React, { createContext, useContext, useEffect, useRef, useState, useCallba

/* ─── CONTEXT ──────────────────────────────────────────────────────────── */

const LiquidGlassContext = createContext({
  theme: 'dark',
  refractionEnabled: true,
  performanceMode: 'balanced', // 'minimal' | 'balanced' | 'maximum'
  setTheme: () => {},
  setPerformanceMode: () => {},
});

export const useLiquidGlass = () => useContext(LiquidGlassContext);


/* ─── SVG FILTER DEFINITIONS ──────────────────────────────────────────────
   These create the displacement maps that simulate light bending
   through convex glass surfaces (Apple's preferred profile).

   Architecture:
   1. feTurbulence → generates organic noise for edge distortion
   2. feDisplacementMap → warps the backdrop using the noise
   3. feSpecularLighting → adds rim-light specular highlights
   4. feComposite/feBlend → composites all layers together
   ──────────────────────────────────────────────────────────────────────── */

const SVG_FILTERS = `
<svg xmlns="http://www.w3.org/2000/svg" width="0" height="0" style="position:abso
  <defs>

    <!-- STANDARD REFRACTION: convex lens with edge bevel -->
    <filter id="lg-refraction-filter" x="-5%" y="-5%" width="110%" height="110%"
            color-interpolation-filters="sRGB">
      <!-- Generate bevel displacement noise -->
```

```
  <feTurbulence type="fractalNoise" baseFrequency="0.015 0.015"
                numOctaves="3" seed="1" result="noise"/>

  <!-- Subtle edge displacement (convex profile) -->
  <feDisplacementMap in="SourceGraphic" in2="noise"
                     scale="12" xChannelSelector="R" yChannelSelector="G"
                     result="displaced"/>

  <!-- Specular rim light -->
  <feSpecularLighting in="noise" surfaceScale="2" specularConstant="0.6"
                      specularExponent="25" lighting-color="#ffffff" result="
    <fePointLight x="200" y="50" z="300"/>
  </feSpecularLighting>

  <!-- Composite specular onto displaced -->
  <feComposite in="specular" in2="displaced" operator="in" result="specMask"/
  <feBlend in="displaced" in2="specMask" mode="screen" result="final"/>
</filter>

<!-- FROSTED REFRACTION: heavier blur + displacement -->
<filter id="lg-frosted-filter" x="-5%" y="-5%" width="110%" height="110%"
        color-interpolation-filters="sRGB">
  <feGaussianBlur in="SourceGraphic" stdDeviation="6" result="blurred"/>
  <feTurbulence type="fractalNoise" baseFrequency="0.02 0.02"
                numOctaves="2" seed="3" result="noise"/>
  <feDisplacementMap in="blurred" in2="noise"
                     scale="8" xChannelSelector="R" yChannelSelector="G"
                     result="displaced"/>
  <feSpecularLighting in="noise" surfaceScale="1.5" specularConstant="0.4"
                      specularExponent="20" lighting-color="#ffffff" result="
    <fePointLight x="150" y="80" z="250"/>
  </feSpecularLighting>
  <feComposite in="specular" in2="displaced" operator="in" result="specMask"/
  <feBlend in="displaced" in2="specMask" mode="screen"/>
</filter>

<!-- CLEAR REFRACTION: minimal, just edge bending -->
<filter id="lg-clear-filter" x="-2%" y="-2%" width="104%" height="104%"
        color-interpolation-filters="sRGB">
  <feTurbulence type="fractalNoise" baseFrequency="0.01 0.01"
                numOctaves="2" seed="7" result="noise"/>
  <feDisplacementMap in="SourceGraphic" in2="noise"
                     scale="6" xChannelSelector="R" yChannelSelector="G"/>
</filter>

<!-- TINTED REFRACTION: color-shifted displacement -->
<filter id="lg-tinted-filter" x="-5%" y="-5%" width="110%" height="110%"
```

```
            color-interpolation-filters="sRGB">
      <feTurbulence type="fractalNoise" baseFrequency="0.018 0.018"
                    numOctaves="3" seed="5" result="noise"/>
      <feDisplacementMap in="SourceGraphic" in2="noise"
                         scale="10" xChannelSelector="R" yChannelSelector="G"
                         result="displaced"/>
      <feColorMatrix in="displaced" type="matrix"
        values="1  0  0.05 0  0
                0  1  0.02 0  0
                0.1  0  1  0  0
                0  0  0  1  0"
        result="tinted"/>
      <feSpecularLighting in="noise" surfaceScale="2" specularConstant="0.5"
                          specularExponent="30" lighting-color="#d8b4fe" result="
        <fePointLight x="180" y="60" z="280"/>
      </feSpecularLighting>
      <feComposite in="specular" in2="tinted" operator="in" result="specMask"/>
      <feBlend in="tinted" in2="specMask" mode="screen"/>
    </filter>

    <!-- DYNAMIC DISPLACEMENT MAP: Generated radial gradient for capsule shapes -
    <radialGradient id="lg-capsule-grad" cx="50%" cy="50%" r="50%">
      <stop offset="0%"   stop-color="rgb(128,128,0)" stop-opacity="1"/>
      <stop offset="70%"  stop-color="rgb(128,128,0)" stop-opacity="1"/>
      <stop offset="85%"  stop-color="rgb(180,180,0)" stop-opacity="1"/>
      <stop offset="95%"  stop-color="rgb(220,40,0)"  stop-opacity="1"/>
      <stop offset="100%" stop-color="rgb(128,128,0)" stop-opacity="0"/>
    </radialGradient>

  </defs>
</svg>
`;


/* ─── PROVIDER ──────────────────────────────────────────────────────── */

export function LiquidGlassProvider({ children, defaultTheme = 'dark', defaultPer
  const [theme, setTheme] = useState(defaultTheme);
  const [performanceMode, setPerformanceMode] = useState(defaultPerformance);
  const [refractionEnabled, setRefractionEnabled] = useState(true);
  const svgInjected = useRef(false);

  // Inject SVG filters into DOM once
  useEffect(() => {
    if (svgInjected.current) return;
    const container = document.createElement('div');
    container.id = 'lg-svg-filters';
```

```
  container.setAttribute('aria-hidden', 'true');
  container.style.cssText = 'position:absolute;width:0;height:0;overflow:hidden
  container.innerHTML = SVG_FILTERS;
  document.body.prepend(container);
  svgInjected.current = true;

  return () => {
    const el = document.getElementById('lg-svg-filters');
    if (el) el.remove();
  };
}, []);

// Apply theme to root
useEffect(() => {
  document.documentElement.setAttribute('data-theme', theme);
}, [theme]);

// Performance mode: disable refraction on low-end devices
useEffect(() => {
  if (performanceMode === 'minimal') {
    setRefractionEnabled(false);
    document.documentElement.style.setProperty('--lg-blur-standard', '12px');
    document.documentElement.style.setProperty('--lg-blur-heavy', '24px');
    document.documentElement.style.setProperty('--lg-saturate', '1.1');
  } else if (performanceMode === 'balanced') {
    setRefractionEnabled(true);
    document.documentElement.style.setProperty('--lg-blur-standard', '40px');
    document.documentElement.style.setProperty('--lg-blur-heavy', '80px');
    document.documentElement.style.setProperty('--lg-saturate', '1.6');
  } else {
    setRefractionEnabled(true);
    document.documentElement.style.setProperty('--lg-blur-standard', '60px');
    document.documentElement.style.setProperty('--lg-blur-heavy', '100px');
    document.documentElement.style.setProperty('--lg-saturate', '1.8');
  }
}, [performanceMode]);

// Detect GPU capability on mount
useEffect(() => {
  try {
    const canvas = document.createElement('canvas');
    const gl = canvas.getContext('webgl') || canvas.getContext('experimental-we
    if (!gl) {
      setPerformanceMode('minimal');
      return;
    }
    const ext = gl.getExtension('WEBGL_debug_renderer_info');
```

```
      if (ext) {
        const renderer = gl.getParameter(ext.UNMASKED_RENDERER_WEBGL).toLowerCase
        // Downgrade for integrated GPUs on mobile
        if (renderer.includes('mali') || renderer.includes('powervr') || renderer
          setPerformanceMode('minimal');
        }
      }
    } catch (e) { /* fallback: keep balanced */ }
  }, []);

  const value = useMemo(() => ({
    theme,
    refractionEnabled,
    performanceMode,
    setTheme,
    setPerformanceMode,
  }), [theme, refractionEnabled, performanceMode]);

  return (
    <LiquidGlassContext.Provider value={value}>
      {children}
    </LiquidGlassContext.Provider>
  );
}


/* ═══════════════════════════════════════════════════════════════════
   COMPONENT WRAPPERS — Use these to apply glass to any element
   ═══════════════════════════════════════════════════════════════════ */

/**
 * <Glass> — Universal glass container
 *
 * @param {string} variant - 'regular' | 'clear' | 'tinted' | 'frosted'
 * @param {string} shape   - 'panel' | 'pill' | 'toolbar' | 'dock' | 'window' | '
 * @param {boolean} animate - Apply materialization on mount
 * @param {string} tint    - CSS color for custom tinting
 * @param {object} style   - Additional inline styles
 * @param {string} className - Additional class names
 */
export function Glass({
  children,
  variant = 'regular',
  shape = '',
  animate = false,
  tint = null,
  style = {},
```

```
  className = '',
  as: Component = 'div',
  ...props
}) {
  const { refractionEnabled } = useLiquidGlass();

  const variantClass = `lg-${variant}`;
  const shapeClass = shape ? `lg-${shape}` : '';
  const animClass = animate ? 'lg-materialize' : '';
  const refractClass = refractionEnabled && variant !== 'clear' ? 'lg-refract' :

  const tintStyle = tint ? {
    '--lg-accent-glass': tint.replace(')', ', 0.15)').replace('rgb(', 'rgba('),
    background: `${tint}15`,
  } : {};

  return (
    <Component
      className={`${variantClass} ${shapeClass} ${animClass} ${refractClass} ${cl
      style={{ ...tintStyle, ...style }}
      {...props}
    >
      {children}
    </Component>
  );
}


/**
 * <GlassToolbar> — Floating action bar (like Apple's iOS 26 toolbar)
 */
export function GlassToolbar({ children, animate = true, className = '', ...props
  return (
    <Glass variant="regular" shape="toolbar" animate={animate} className={classNa
      {children}
    </Glass>
  );
}


/**
 * <GlassDock> — Bottom dock with heavier blur
 */
export function GlassDock({ children, className = '', ...props }) {
  return (
    <Glass variant="regular" shape="dock" animate className={className} {...props
      {children}
```

```
    </Glass>
  );
}


/**
 * <GlassPanel> — Side panels, modals, sheets
 */
export function GlassPanel({ children, className = '', ...props }) {
  return (
    <Glass variant="regular" shape="panel" animate className={className} {...prop
      {children}
    </Glass>
  );
}


/**
 * <GlassWindow> — App window container with titlebar
 */
export function GlassWindow({ title, children, className = '', onClose, onMinimiz
  return (
    <Glass variant="regular" shape="window" animate className={className} {...pro
      <div className="lg-window-titlebar">
        <div style={{ display: 'flex', gap: '6px' }}>
          {onClose && (
            <button onClick={onClose} style={{
              width: 12, height: 12, borderRadius: '50%', border: 'none',
              background: '#ff5f57', cursor: 'pointer'
            }}/>
          )}
          {onMinimize && (
            <button onClick={onMinimize} style={{
              width: 12, height: 12, borderRadius: '50%', border: 'none',
              background: '#febc2e', cursor: 'pointer'
            }}/>
          )}
          {onMaximize && (
            <button onClick={onMaximize} style={{
              width: 12, height: 12, borderRadius: '50%', border: 'none',
              background: '#28c840', cursor: 'pointer'
            }}/>
          )}
        </div>
        {title && (
          <span style={{
            flex: 1, textAlign: 'center',
```

```
            color: 'var(--lg-text-secondary)',
            fontSize: '13px', fontWeight: 500,
            letterSpacing: '0.01em'
          }}>
            {title}
          </span>
        )}
      </div>
      <div className="lg-window-body">
        {children}
      </div>
    </Glass>
  );
}


/**
 * <GlassNotification> — Toast / notification
 */
export function GlassNotification({ children, className = '', ...props }) {
  return (
    <Glass variant="frosted" shape="notification" animate className={className} {
      {children}
    </Glass>
  );
}


/* ═══════════════════════════════════════════════════════════════════
   HOOKS
   ═══════════════════════════════════════════════════════════════════ */

/**
 * useScrollResponsive — Shrinks glass elements on scroll (Apple tab bar behavior
 * Returns a ref to attach to the scrollable container
 */
export function useScrollResponsive(targetRef, { threshold = 10 } = {}) {
  const [scrolled, setScrolled] = useState(false);

  useEffect(() => {
    const el = targetRef?.current;
    if (!el) return;

    let ticking = false;
    const onScroll = () => {
      if (!ticking) {
        requestAnimationFrame(() => {
```

```
          setScrolled(el.scrollTop > threshold);
          ticking = false;
        });
        ticking = true;
      }
    };

    el.addEventListener('scroll', onScroll, { passive: true });
    return () => el.removeEventListener('scroll', onScroll);
  }, [targetRef, threshold]);

  return scrolled;
}


/**
 * useSpecularTilt — Moves the specular highlight based on pointer/device orienta
 * Returns style object to spread onto the glass element
 */
export function useSpecularTilt(ref) {
  const [specStyle, setSpecStyle] = useState({});

  useEffect(() => {
    const el = ref?.current;
    if (!el) return;

    // Pointer-based (desktop)
    const onPointer = (e) => {
      const rect = el.getBoundingClientRect();
      const x = ((e.clientX - rect.left) / rect.width) * 360;
      const y = ((e.clientY - rect.top) / rect.height) * 360;
      const angle = Math.atan2(y - 180, x - 180) * (180 / Math.PI) + 180;
      setSpecStyle({ '--lg-specular-angle': `${angle}deg` });
    };

    // Device orientation (mobile)
    const onOrientation = (e) => {
      if (e.gamma === null) return;
      const angle = ((e.gamma + 90) / 180) * 360;
      setSpecStyle({ '--lg-specular-angle': `${angle}deg` });
    };

    el.addEventListener('pointermove', onPointer, { passive: true });
    window.addEventListener('deviceorientation', onOrientation, { passive: true }

    return () => {
      el.removeEventListener('pointermove', onPointer);
```

```
      window.removeEventListener('deviceorientation', onOrientation);
    };
  }, [ref]);

  return specStyle;
}


/**
 * useMaterialization — Trigger materialize/dematerialize animations
 */
export function useMaterialization() {
  const [visible, setVisible] = useState(false);

  const materialize = useCallback(() => setVisible(true), []);
  const dematerialize = useCallback(() => setVisible(false), []);

  const className = visible ? 'lg-materialize' : 'lg-dematerialize';

  return { visible, materialize, dematerialize, className };
}


export default LiquidGlassProvider;
```