

# Image Sphere

```
import React, { useState, useEffect, useRef, useMemo } from "react";
import { Menu, X, ChevronDown } from "lucide-react";

// Inline Button component
const Button = React.forwardRef<HTMLButtonElement,
React.ButtonHTMLAttributes<HTMLButtonElement>>(>
  ({ className = "", children, ...props }, ref) => {
    return (
      <button
        ref={ref}
        className={`inline-flex items-center justify-center rounded-md px-4
py-2 text-sm font-medium transition-colors ${className}`}
        {...props}
      >
        {children}
      </button>
    );
  }
);
Button.displayName = "Button";

// BlurText animation component
interface BlurTextProps {
  text: string;
  delay?: number;
  animateBy?: "words" | "letters";
  direction?: "top" | "bottom";
  className?: string;
  style?: React.CSSProperties;
}

const BlurText: React.FC<BlurTextProps> = ({
  text,
  delay = 50,
  animateBy = "words",
  direction = "top",
  className = "",
  style,
}) => {
  const [inView, setInView] = useState(false);
  const ref = useRef<HTMLParagraphElement>(null);

  useEffect(() => {
    const observer = new IntersectionObserver(
```

```

    ([entry]) => {
      if (entry.isIntersecting) {
        setInView(true);
      }
    },
    { threshold: 0.1 }
  );

  if (ref.current) {
    observer.observe(ref.current);
  }

  return () => {
    if (ref.current) {
      observer.unobserve(ref.current);
    }
  };
}, []);

const segments = useMemo(() => {
  return animateBy === "words" ? text.split(" ") : text.split("");
}, [text, animateBy]);

return (
  <p ref={ref} className={`inline-flex flex-wrap ${className}`} style={style}>
    {segments.map((segment, i) => (
      <span
        key={i}
        style={{
          display: "inline-block",
          filter: inView ? "blur(0px)" : "blur(10px)",
          opacity: inView ? 1 : 0,
          transform: inView ? "translateY(0)" : `translateY(${direction === "top" ?
            "-20px" : "20px"})`,
          transition: `all 0.5s ease-out ${i * delay}ms`,
        }}
      >
        {segment}
        {animateBy === "words" && i < segments.length - 1 ? "\u00A0" : ""}
      </span>
    ))}
  </p>
);
};

export default function Component() {
  const [isDark, setIsDark] = useState(true);

```

```

const [isMenuOpen, setIsMenuOpen] = useState(false);
const menuRef = useRef<HTMLDivElement>(null);
const buttonRef = useRef<HTMLButtonElement>(null);

useEffect(() => {
  document.documentElement.classList.add("dark");
}, []);

useEffect(() => {
  const handleClickOutside = (event: MouseEvent) => {
    if (
      isMenuOpen &&
      menuRef.current &&
      buttonRef.current &&
      !menuRef.current.contains(event.target as Node) &&
      !buttonRef.current.contains(event.target as Node)
    ) {
      setIsMenuOpen(false);
    }
  };

  document.addEventListener("mousedown", handleClickOutside);
  return () => document.removeEventListener("mousedown",
handleClickOutside);
}, [isMenuOpen]);

const toggleTheme = () => {
  const newTheme = !isDark;
  setIsDark(newTheme);
  if (newTheme) {
    document.documentElement.classList.add("dark");
  } else {
    document.documentElement.classList.remove("dark");
  }
};

const menuItems = [
  { label: "HOME", href: "#", highlight: true },
  { label: "ABOUT", href: "#" },
  { label: "PROJECTS", href: "#" },
  { label: "EXPERIENCE", href: "#" },
  { label: "EDUCATION", href: "#" },
  { label: "WRITING", href: "#" },
  { label: "CONTACT", href: "#" },
];

return (

```

```

<div
  className="min-h-screen text-foreground transition-colors"
  style={{
    backgroundColor: isDark ? "hsl(0 0% 0%)" : "hsl(0 0% 98%)",
    color: isDark ? "hsl(0 0% 100%)" : "hsl(0 0% 10%)",
  }}
>
  { /* Header */ }
  <header className="fixed top-0 left-0 right-0 z-50 px-6 py-6">
    <nav className="flex items-center justify-between max-w-screen-2xl
mx-auto">
      { /* Menu Button */ }
      <div className="relative">
        <button
          ref={buttonRef}
          type="button"
          className="p-2 transition-colors duration-300 z-50 text-neutral-500
hover:text-black dark: hover:text-white"
          aria-label={isMenuOpen ? "Close menu" : "Open menu"}
          onClick={() => setIsMenuOpen(!isMenuOpen)}
        >
          {isMenuOpen ? (
            <X className="w-8 h-8 transition-colors duration-300"
strokeWidth={2} />
          ) : (
            <Menu className="w-8 h-8 transition-colors duration-300"
strokeWidth={2} />
          )}
        </button>

        {isMenuOpen && (
          <div
            ref={menuRef}
            className="absolute top-full left-0 w-[200px] md:w-[240px]
border-none shadow-2xl mt-2 ml-4 p-4 rounded-lg z-[100]"
            style={{
              backgroundColor: isDark ? "hsl(0 0% 0%)" : "hsl(0 0% 98%)",
            }}
          >
            {menuItems.map((item) => (
              <a
                key={item.label}
                href={item.href}
                className="block text-lg md:text-xl font-bold tracking-tight py-1.5
px-2 cursor-pointer transition-colors duration-300"
                style={{
                  color: item.highlight ? "#C3E41D" : isDark ? "hsl(0 0% 100%)" :

```

```

"hsl(0 0% 10%)",
    }}
    onMouseEnter={(e) => {
        excurrentTargetxstylexcolor = "#C3E41D";
    }}
    onMouseLeave={(e) => {
        excurrentTargetxstylexcolor = item.highlight ? "#C3E41D" :
(isDark ? "hsl(0 0% 100%)" : "hsl(0 0% 10%)");
    }}
    onClick={() => setIsMenuOpen(false)}
  >
    {item.label}
  </a>
)}}
</div>
)}
</div>

{/* Signature */}
<div className="text-4xl" style={{ color: isDark ? "hsl(0 0% 100%)" :
"hsl(0 0% 10%)", fontFamily: "'Brush Script MT', 'Lucida Handwriting',
cursive" }}>
  A
</div>

{/* Theme Toggle */}
<button
  type="button"
  onClick={toggleTheme}
  className="relative w-16 h-8 rounded-full hover:opacity-80 transition-
opacity"
  style={{ backgroundColor: isDark ? "hsl(0 0% 15%)" : "hsl(0 0%
90%)" }}
  aria-label="Toggle theme"
>
  <div
    className="absolute top-1 left-1 w-6 h-6 rounded-full transition-
transform duration-300"
    style={{
      backgroundColor: isDark ? "hsl(0 0% 100%)" : "hsl(0 0% 10%)",
      transform: isDark ? "translateX(2rem)" : "translateX(0)",
    }}
  />
</button>
</nav>
</header>

```

```

    { /* Hero Section */ }
    <main className="relative min-h-screen flex flex-col">
      { /* Centered Main Name - Always Perfectly Centered */ }
      <div className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2
w-full px-4">
        <div className="relative text-center">
          <div>
            <BlurText
              text="ALEX"
              delay={100}
              animateBy="letters"
              direction="top"
              className="font-bold text-[100px] sm:text-[140px] md:text-[180px]
lg:text-[210px] leading-[0.75] tracking-tighter uppercase justify-center
whitespace-nowrap"
              style={{ color: "#C3E41D", fontFamily: "'Fira Code', monospace" }}
            />
          </div>
          <div>
            <BlurText
              text="KANE"
              delay={100}
              animateBy="letters"
              direction="top"
              className="font-bold text-[100px] sm:text-[140px] md:text-[180px]
lg:text-[210px] leading-[0.75] tracking-tighter uppercase justify-center
whitespace-nowrap"
              style={{ color: "#C3E41D", fontFamily: "'Fira Code', monospace" }}
            />
          </div>

          { /* Profile Picture */ }
          <div className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-
y-1/2 z-10">
            <div className="w-[65px] h-[110px] sm:w-[90px] sm:h-[152px]
md:w-[110px] md:h-[185px] lg:w-[129px] lg:h-[218px] rounded-full overflow-
hidden shadow-2xl transition-transform duration-300 hover:scale-110 cursor-
pointer">
              
            </div>
          </div>

```

```

    </div>
  </div>

  {/* Tagline - Proper Distance Below Hero */}
  <div className="absolute bottom-16 sm:bottom-20 md:bottom-24
lg:bottom-32 xl:bottom-36 left-1/2 -translate-x-1/2 w-full px-6">
    <div className="flex justify-center">
      <BlurText
        text="Designing human experiences in code."
        delay={150}
        animateBy="words"
        direction="top"
        className="text-[15px] sm:text-[18px] md:text-[20px] lg:text-[22px]
text-center transition-colors duration-300 text-neutral-500 hover:text-black
dark:hover:text-white"
        style={{ fontFamily: "'Antic', sans-serif" }}
      />
    </div>
  </div>

  {/* Scroll Indicator */}
  <button
    type="button"
    className="absolute bottom-6 md:bottom-10 left-1/2 -translate-x-1/2
transition-colors duration-300"
    aria-label="Scroll down"
  >
    <ChevronDown className="w-5 h-5 md:w-8 md:h-8 text-neutral-500
hover:text-black dark:hover:text-white transition-colors duration-300" />
  </button>
</main>
</div>
);
}

```