Build this as my initial prototype

Copy-paste this component to /components/ui folder:

```tsx

beams-background.tsx

"use client";

import { useEffect, useRef } from "react";

import { motion } from "motion/react";

import { cn } from "@/lib/utils";

interface AnimatedGradientBackgroundProps {

className?: string;

children?: React.ReactNode;

intensity?: "subtle" | "medium" | "strong";

}

interface Beam {

x: number;

y: number;

width: number;

length: number;

angle: number;

speed: number;

opacity: number;

hue: number;

pulse: number;

pulseSpeed: number;

}

function createBeam(width: number, height: number): Beam {

const angle = -35 + Math.random() \* 10;

return {

x: Math.random() \* width \* 1.5 - width \* 0.25,

y: Math.random() \* height \* 1.5 - height \* 0.25,

width: 30 + Math.random() \* 60,

length: height \* 2.5,

angle: angle,

speed: 0.6 + Math.random() \* 1.2,

opacity: 0.12 + Math.random() \* 0.16,

hue: 190 + Math.random() \* 70,

pulse: Math.random() \* Math.PI \* 2,

pulseSpeed: 0.02 + Math.random() \* 0.03,

};

}

export function BeamsBackground({

className,

intensity = "strong",

}: AnimatedGradientBackgroundProps) {

const canvasRef = useRef<HTMLCanvasElement>(null);

const beamsRef = useRef<Beam[]>([]);

const animationFrameRef = useRef<number>(0);

const MINIMUM\_BEAMS = 20;

const opacityMap = {

subtle: 0.7,

medium: 0.85,

strong: 1,

};

useEffect(() => {

const canvas = canvasRef.current;

if (!canvas) return;

const ctx = canvas.getContext("2d");

if (!ctx) return;

const updateCanvasSize = () => {

const dpr = window.devicePixelRatio || 1;

canvas.width = window.innerWidth \* dpr;

canvas.height = window.innerHeight \* dpr;

canvas.style.width = `${window.innerWidth}px`;

canvas.style.height = `${window.innerHeight}px`;

ctx.scale(dpr, dpr);

const totalBeams = MINIMUM\_BEAMS \* 1.5;

beamsRef.current = Array.from({ length: totalBeams }, () =>

createBeam(canvas.width, canvas.height)

);

};

updateCanvasSize();

window.addEventListener("resize", updateCanvasSize);

function resetBeam(beam: Beam, index: number, totalBeams: number) {

if (!canvas) return beam;

const column = index % 3;

const spacing = canvas.width / 3;

beam.y = canvas.height + 100;

beam.x =

column \* spacing +

spacing / 2 +

(Math.random() - 0.5) \* spacing \* 0.5;

beam.width = 100 + Math.random() \* 100;

beam.speed = 0.5 + Math.random() \* 0.4;

beam.hue = 190 + (index \* 70) / totalBeams;

beam.opacity = 0.2 + Math.random() \* 0.1;

return beam;

}

function drawBeam(ctx: CanvasRenderingContext2D, beam: Beam) {

ctx.save();

ctx.translate(beam.x, beam.y);

ctx.rotate((beam.angle \* Math.PI) / 180);

// Calculate pulsing opacity

const pulsingOpacity =

beam.opacity \*

(0.8 + Math.sin(beam.pulse) \* 0.2) \*

opacityMap[intensity];

const gradient = ctx.createLinearGradient(0, 0, 0, beam.length);

// Enhanced gradient with multiple color stops

gradient.addColorStop(0, `hsla(${beam.hue}, 85%, 65%, 0)`);

gradient.addColorStop(

0.1,

`hsla(${beam.hue}, 85%, 65%, ${pulsingOpacity \* 0.5})`

);

gradient.addColorStop(

0.4,

`hsla(${beam.hue}, 85%, 65%, ${pulsingOpacity})`

);

gradient.addColorStop(

0.6,

`hsla(${beam.hue}, 85%, 65%, ${pulsingOpacity})`

);

gradient.addColorStop(

0.9,

`hsla(${beam.hue}, 85%, 65%, ${pulsingOpacity \* 0.5})`

);

gradient.addColorStop(1, `hsla(${beam.hue}, 85%, 65%, 0)`);

ctx.fillStyle = gradient;

ctx.fillRect(-beam.width / 2, 0, beam.width, beam.length);

ctx.restore();

}

function animate() {

if (!canvas || !ctx) return;

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.filter = "blur(35px)";

const totalBeams = beamsRef.current.length;

beamsRef.current.forEach((beam, index) => {

beam.y -= beam.speed;

beam.pulse += beam.pulseSpeed;

// Reset beam when it goes off screen

if (beam.y + beam.length < -100) {

resetBeam(beam, index, totalBeams);

}

drawBeam(ctx, beam);

});

animationFrameRef.current = requestAnimationFrame(animate);

}

animate();

return () => {

window.removeEventListener("resize", updateCanvasSize);

if (animationFrameRef.current) {

cancelAnimationFrame(animationFrameRef.current);

}

};

}, [intensity]);

return (

<div

className={cn(

"relative min-h-screen w-full overflow-hidden bg-neutral-950",

className

)}

>

<canvas

ref={canvasRef}

className="absolute inset-0"

style={{ filter: "blur(15px)" }}

/>

<motion.div

className="absolute inset-0 bg-neutral-950/5"

animate={{

opacity: [0.05, 0.15, 0.05],

}}

transition={{

duration: 10,

ease: "easeInOut",

repeat: Number.POSITIVE\_INFINITY,

}}

style={{

backdropFilter: "blur(50px)",

}}

/>

<div className="relative z-10 flex h-screen w-full items-center justify-center">

<div className="flex flex-col items-center justify-center gap-6 px-4 text-center">

<motion.h1

className="text-6xl md:text-7xl lg:text-8xl font-semibold text-white tracking-tighter"

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

transition={{ duration: 0.8 }}

>

Beams

<br />

Background

</motion.h1>

<motion.p

className="text-lg md:text-2xl lg:text-3xl text-white/70 tracking-tighter"

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

transition={{ duration: 0.8 }}

>

For your pleasure

</motion.p>

</div>

</div>

</div>

);

}

demo.tsx

import { BeamsBackground } from "@/components/ui/beams-background"

export function BeamsBackgroundDemo() {

return < BeamsBackground />

}

```

Install these NPM dependencies:

```bash

motion

```