🎬 Complete Animation Library Guide for Next.js 15 (2025)

Executive Summary: Your Best Choices for La Nuit de l'Info

Priority	Library	Use Case	Dev Time	Bundle Size
🥇 #1	**Motion**	UI animations, page transitions, gestures	⚡ Fast	~32KB
🥈 #2	**GSAP + ScrollTrigger**	Scroll-driven storytelling, timelines	Medium	~23KB core
🥉 #3	**Lottie (.lottie)**	Pre-made complex animations	⚡ Fast	Variable
🏅 #4	**Spline**	3D scenes without coding	⚡ Fast	~500KB+
⭐ Bonus	**Rive**	Interactive state-based animations	Medium	~200KB

## 🏆 #1: MOTION (formerly Framer Motion)

Why It's #1 for Your Project

- **2.5x faster** than GSAP at animating from unknown values
- **6x faster** at animating between different value types
- **Native React integration** — feels like writing React, not learning a new API
- **MIT License** — fully open source, no restrictions
- **Version 4.0** just released (December 2025)

Installation

bash

```bash
npm install motion
# or
pnpm add motion
```

Key Features Perfect for Your Project

1. Basic Entrance Animations

tsx

```tsx
```

```
import { motion } from "motion/react"

export function HeroSection() {
  return (
    <motion.div
      initial={{ opacity: 0, y: 50 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.8, ease: "easeOut" }}
    >
      <h1>Le Village Numérique Résistant</h1>
    </motion.div>
  )
}
```

2. Scroll-Triggered Animations

```tsx
import { motion, useScroll, useTransform } from "motion/react"

export function ScrollReveal({ children }: { children: React.ReactNode }) {
  const { scrollYProgress } = useScroll()
  const opacity = useTransform(scrollYProgress, [0, 0.5], [0, 1])
  const y = useTransform(scrollYProgress, [0, 0.5], [100, 0])

  return (
    <motion.div style={{ opacity, y }}>
      {children}
    </motion.div>
  )
}
```

3. Page Transitions (AnimatePresence)

```tsx
```

```
// layout.tsx
import { AnimatePresence, motion } from "motion/react"

export default function Template({ children }: { children: React.ReactNode }) {
  return (
    <AnimatePresence mode="wait">
      <motion.div
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        exit={{ opacity: 0, x: 20 }}
        transition={{ duration: 0.3 }}
      >
        {children}
      </motion.div>
    </AnimatePresence>
  )
}
```

4. Staggered Children (for lists, cards)

```
tsx
```

```
const containerVariants = {
  hidden: { opacity: 0 },
  visible: {
    opacity: 1,
    transition: {
      staggerChildren: 0.1, // Delay between each child
      delayChildren: 0.2
    }
  }
}

const itemVariants = {
  hidden: { opacity: 0, y: 20 },
  visible: { opacity: 1, y: 0 }
}

export function PillarCards() {
  const pillars = ["Inclusif", "Responsable", "Durable"]

  return (
    <motion.div
      variants={containerVariants}
      initial="hidden"
      whileInView="visible"
      viewport={{ once: true }}
    >
      {pillars.map((pillar) => (
        <motion.div key={pillar} variants={itemVariants}>
          {pillar}
        </motion.div>
      ))}
    </motion.div>
  )
}
```

5. Interactive Hover & Tap

```tsx
```

```
<motion.button
  whileHover={{
    scale: 1.05,
    boxShadow: "0 10px 30px rgba(0,0,0,0.2)"
  }}
  whileTap={{ scale: 0.95 }}
  transition={{ type: "spring", stiffness: 400, damping: 17 }}
>
  Rejoindre le Village
</motion.button>
```

6. Layout Animations (for role selection)

tsx

```
<motion.div layout layoutId="selected-role">
  {/* This will animate smoothly when it moves between positions */}
</motion.div>
```

🥈 #2: GSAP + ScrollTrigger

Why Use GSAP

- **Industry standard** for complex timeline animations
- **ScrollTrigger plugin** is unmatched for scroll-based storytelling
- **Precise control** over every aspect of animation
- **Works with any framework**

When to Combine with Motion

Use GSAP specifically for:

- Complex scroll-driven narratives (your scrollytelling)
- Timeline-based sequences with precise control
- Pinning sections while scrolling

Installation

bash

```
npm install gsap
```

Key Examples

1. ScrollTrigger for Scrollytelling Sections

```
tsx
```

```
"use client"
import { useEffect, useRef } from "react"
import { gsap } from "gsap"
import { ScrollTrigger } from "gsap/ScrollTrigger"

gsap.registerPlugin(ScrollTrigger)

export function ScrollytellingSection() {
  const sectionRef = useRef<HTMLDivElement>(null)
  const textRef = useRef<HTMLDivElement>(null)

  useEffect(() => {
    const ctx = gsap.context(() => {
      // Pin the section while scrolling through content
      gsap.to(textRef.current, {
        y: -200,
        ease: "none",
        scrollTrigger: {
          trigger: sectionRef.current,
          start: "top top",
          end: "bottom top",
          scrub: true, // Links animation to scroll position
          pin: true,   // Pins the section
        }
      })
    }, sectionRef)

    return () => ctx.revert() // Cleanup
  }, [])

  return (
    <section ref={sectionRef} className="h-[200vh]">
      <div ref={textRef}>
        <h2>La Crise Windows 10</h2>
        <p>240 millions d'ordinateurs...</p>
      </div>
    </section>
  )
}
```

2. Timeline Animation

```tsx
useEffect(() => {
  const tl = gsap.timeline({
    scrollTrigger: {
      trigger: ".hero",
      start: "top center",
      end: "bottom center",
      scrub: 1
    }
  })

  tl.from(".hero-title", { opacity: 0, y: 100 })
    .from(".hero-subtitle", { opacity: 0, y: 50 }, "-=0.5")
    .from(".hero-cta", { opacity: 0, scale: 0.8 }, "-=0.3")

  return () => tl.kill()
}, [])
```

3. Parallax Effect

```tsx
gsap.to(".background-layer", {
  y: -100,
  ease: "none",
  scrollTrigger: {
    trigger: ".section",
    start: "top bottom",
    end: "bottom top",
    scrub: true
  }
})
```

🥉 #3: LOTTIE

Why Lottie

- **Pre-made animations** from LottieFiles.com — huge time saver!
- **Vector-based** — crisp at any size

- **Small file sizes** — especially with .lottie format (up to 90% smaller)
- **Easy to integrate**

Best Use Cases for Your Project

- ✅ Loading animations
- ✅ Success/completion celebrations
- ✅ Character animations (Asterix-style mascot)
- ✅ Icon animations (Linux penguin, shield, etc.)
- ✅ Progress indicators

Installation

bash

```bash
npm install lottie-react
# For better performance, use dotLottie:
npm install @lottiefiles/dotlottie-react
```

Usage Examples

Basic Lottie Animation

tsx

```tsx
import Lottie from "lottie-react"
import celebrationAnimation from "./animations/celebration.json"

export function SuccessAnimation() {
  return (
    <Lottie
      animationData={celebrationAnimation}
      loop={false}
      style={{ width: 300, height: 300 }}
    />
  )
}
```

dotLottie (Recommended - 90% smaller files)

tsx

```
import { DotLottieReact } from "@lottiefiles/dotlottie-react"

export function LoadingSpinner() {
  return (
    <DotLottieReact
      src="/animations/loading.lottie"
      loop
      autoplay
      style={{ width: 100, height: 100 }}
    />
  )
}
```

Controlled Animation (Play on Scroll)

```
tsx
```

```
import { useRef, useEffect } from "react"
import Lottie, { LottieRefCurrentProps } from "lottie-react"

export function ScrollControlledAnimation() {
  const lottieRef = useRef<LottieRefCurrentProps>(null)

  useEffect(() => {
    const handleScroll = () => {
      if (!lottieRef.current) return
      const scrollPercent = window.scrollY / (document.body.scrollHeight - window.innerHeight)
      const frame = Math.floor(scrollPercent * lottieRef.current.getDuration(true))
      lottieRef.current.goToAndStop(frame, true)
    }

    window.addEventListener("scroll", handleScroll)
    return () => window.removeEventListener("scroll", handleScroll)
  }, [])

  return (
    <Lottie
      lottieRef={lottieRef}
      animationData={scrollAnimation}
      autoplay={false}
    />
  )
}
```

Where to Find Free Lottie Animations

- **LottieFiles.com** — Huge library, many free options
- Search for: "celebration", "success", "loading", "penguin", "shield", "computer"

🏅 #4: SPLINE (3D Without Code)

Why Spline

- **Browser-based 3D editor** — no coding for 3D design
- **Direct React/Next.js export**
- **Built-in interactions** — hover, click, scroll

- **Fast to learn** — intuitive Figma-like interface

When to Use

- ✅ Hero section 3D element (floating village, computer model)
- ✅ Interactive 3D icons
- ✅ Background 3D scenes
- ⚠️ Caution: Can be heavy, use sparingly

Installation

bash

```bash
npm install @splinetool/react-spline @splinetool/runtime
```

Basic Integration

tsx

```tsx
"use client"
import Spline from "@splinetool/react-spline"

export function Hero3D() {
  return (
    <div className="h-screen">
      <Spline
        scene="https://prod.spline.design/YOUR-SCENE-ID/scene.splinecode"
      />
    </div>
  )
}
```

Next.js Optimized (with placeholder)

tsx

```
import Spline from "@splinetool/react-spline/next"

export function Hero3D() {
  return (
    <Spline
      scene="https://prod.spline.design/YOUR-SCENE-ID/scene.splinecode"
    />
  )
}
```

Interactive Spline (Responding to Events)

```tsx
import { useRef } from "react"
import Spline from "@splinetool/react-spline"

export function Interactive3D() {
  const splineRef = useRef()

  const onLoad = (spline) => {
    splineRef.current = spline
  }

  const triggerAnimation = () => {
    if (splineRef.current) {
      // Trigger a Spline event by object name
      splineRef.current.emitEvent("mouseDown", "Button")
    }
  }

  return (
    <>
      <Spline scene="..." onLoad={onLoad} />
      <button onClick={triggerAnimation}>Trigger</button>
    </>
  )
}
```

Performance Tips

```tsx
```

```
import dynamic from "next/dynamic"
import { Suspense } from "react"

// Lazy load Spline to reduce initial bundle
const Spline = dynamic(() => import("@splinetool/react-spline"), {
  ssr: false,
  loading: () => <div className="animate-pulse bg-gray-200 h-full" />
})
```

⭐ BONUS: RIVE

Why Consider Rive

- **State Machine** — animations that respond to user input
- **10-15x smaller** file sizes than Lottie
- **60 FPS performance** vs Lottie's ~17 FPS on complex animations
- **Interactive by design** — buttons, toggles, hover states

Best For Your Project

- Interactive UI elements (buttons that animate on state)
- Character animations that react to user actions
- Game-like interactions

Installation

```bash
npm install @rive-app/react-canvas
```

Basic Usage

```tsx
```

```
import { useRive } from "@rive-app/react-canvas"

export function InteractiveButton() {
  const { RiveComponent, rive } = useRive({
    src: "/animations/button.riv",
    stateMachines: "ButtonState",
    autoplay: true,
  })

  return <RiveComponent />
}
```

Interactive State Machine

```tsx
import { useRive, useStateMachineInput } from "@rive-app/react-canvas"

export function ToggleAnimation() {
  const { rive, RiveComponent } = useRive({
    src: "/toggle.riv",
    stateMachines: "Toggle",
    autoplay: true,
  })

  const isOnInput = useStateMachineInput(rive, "Toggle", "isOn")

  const handleToggle = () => {
    if (isOnInput) {
      isOnInput.value = !isOnInput.value
    }
  }

  return (
    <div onClick={handleToggle}>
      <RiveComponent />
    </div>
  )
}
```

📊 COMPARISON TABLE

Feature	Motion	GSAP	Lottie	Spline	Rive
Learning Curve	Easy	Medium	Easy	Easy	Medium
React Integration	Native	Refs/hooks	Wrapper	Wrapper	Wrapper
Bundle Size	~32KB	~23KB	Variable	~500KB+	~200KB
Scroll Animations	✅ Good	✅ Best	⚠️ Manual	⚠️ Limited	✅ Good
Page Transitions	✅ Best	✅ Good	❌ No	❌ No	❌ No
3D Support	❌ No	❌ No	❌ No	✅ Best	❌ No
Pre-made Library	❌ No	❌ No	✅ Huge	✅ Community	✅ Growing
Interactivity	✅ Great	✅ Great	⚠️ Basic	✅ Good	✅ Best
Performance	✅ Best	✅ Great	⚠️ Variable	⚠️ Heavy	✅ Great
License	MIT	Free*	MIT	Free*	Free*

*Free with restrictions on commercial tools

🎯 RECOMMENDED STACK FOR YOUR HACKATHON

Primary Animation Stack

```bash
npm install motion gsap lottie-react
```

Why This Combination?

1. **Motion** → All UI animations, page transitions, hover/tap interactions
2. **GSAP ScrollTrigger** → Scroll-driven storytelling sections
3. **Lottie** → Pre-made celebration, loading, and character animations

Optional Additions

```bash
```

```
# If you want 3D hero element (use sparingly)
npm install @splinetool/react-spline @splinetool/runtime

# For zero-config list animations
npm install @formkit/auto-animate
```

🚀 QUICK START TEMPLATE

```tsx
// app/layout.tsx
import { AnimatePresence, motion } from "motion/react"
import "./globals.css"

export default function RootLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <html lang="fr">
      <body>
        <AnimatePresence mode="wait">
          {children}
        </AnimatePresence>
      </body>
    </html>
  )
}
```

```tsx
```

```tsx
// app/page.tsx
"use client"
import { motion } from "motion/react"
import { useEffect } from "react"
import gsap from "gsap"
import { ScrollTrigger } from "gsap/ScrollTrigger"
import Lottie from "lottie-react"
import heroAnimation from "@/animations/hero.json"

gsap.registerPlugin(ScrollTrigger)

export default function Home() {
  useEffect(() => {
    // GSAP scroll animations
    gsap.from(".scroll-reveal", {
      opacity: 0,
      y: 100,
      stagger: 0.2,
      scrollTrigger: {
        trigger: ".scroll-section",
        start: "top 80%",
      }
    })
  }, [])

  return (
    <main>
      {/* Hero with Motion entrance */}
      <motion.section
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        transition={{ duration: 1 }}
        className="h-screen flex items-center justify-center"
      >
        <motion.h1
          initial={{ y: 50, opacity: 0 }}
          animate={{ y: 0, opacity: 1 }}
          transition={{ delay: 0.3, duration: 0.8 }}
          className="text-6xl font-bold"
        >
          Le Village Numérique Résistant
        </motion.h1>
        {/* Lottie animation */}
        <Lottie
          animationData={heroAnimation}
          className="w-64 h-64"
        />
      </motion.section>

      {/* Scroll section with GSAP */}
      <section className="scroll-section min-h-screen">
        <div className="scroll-reveal">Content 1</div>
        <div className="scroll-reveal">Content 2</div>
        <div className="scroll-reveal">Content 3</div>
      </section>
    </main>
  )
}
```

🎨 ANIMATION TIMING CHEATSHEET

Recommended Durations

```
Micro-interactions (buttons, hover): 0.1s - 0.2s
Page transitions: 0.3s - 0.5s
Section reveals: 0.5s - 0.8s
Hero animations: 0.8s - 1.2s
Complex sequences: 1s - 2s
```

Best Easing Functions

```
tsx
```

```
// Motion/Framer Motion
transition={{ ease: "easeOut" }}        // Most natural for entrances
transition={{ ease: "easeInOut" }}      // For transitions
transition={{ type: "spring", stiffness: 300, damping: 30 }} // Bouncy

// GSAP
ease: "power2.out"    // Smooth deceleration
ease: "power3.inOut"  // Dramatic
ease: "back.out(1.7)" // Slight overshoot
ease: "elastic.out(1, 0.3)" // Bouncy
```

⚡ PERFORMANCE BEST PRACTICES

1. **Only animate `transform` and `opacity`** — GPU accelerated
2. **Use `will-change` sparingly** — browser hint for optimization
3. **Lazy load heavy animations** — especially Lottie and Spline
4. **Reduce motion for accessibility**:

```tsx
const prefersReducedMotion = window.matchMedia("(prefers-reduced-motion: reduce)").matches

// In Motion
<motion.div
  initial={{ opacity: 0, y: prefersReducedMotion ? 0 : 50 }}
  animate={{ opacity: 1, y: 0 }}
/>
```

5. **Clean up GSAP on unmount**:

```tsx
useEffect(() => {
  const ctx = gsap.context(() => {
    // All GSAP animations here
  })
  return () => ctx.revert()
}, [])
```

📚 RESOURCES

Motion (Framer Motion)

- Docs: https://motion.dev/
- Examples: https://motion.dev/examples

GSAP

- Docs: https://gsap.com/docs/
- ScrollTrigger: https://gsap.com/docs/v3/Plugins/ScrollTrigger/

Lottie

- Free animations: https://lottiefiles.com/
- dotLottie docs: https://lottiefiles.github.io/dotlottie-web/

Spline

- Editor: https://spline.design/
- Community: https://spline.design/community

Rive

- Editor: https://rive.app/
- Community: https://rive.app/community/

Good luck with La Nuit de l'Info! 🚀🎉