Documentación

“Interactive Multi-Client System for Real-Time Visuals — Aerials”

Título: Sistema interactivo multi-cliente para visuales en tiempo real — *Aerials*
Autores: Stella Pérez · José Ignacio Trujillo
Curso: Sistemas Físicos Interactivos 2 — UPB Facultad de Ingenierías (Ago–Oct 2025)

Analisis funcional

Propósito del sistema

Proveer un sistema interactivo multi-cliente en tiempo real para controlar visuales generativas en TouchDesigner (TD), donde:

- **Control** define parámetros globales de la escena.
- **Desktop** fija el contexto (tiempo y estación).
- **Mobile** aporta colores de usuario (una antena por participante).
- **Servidor Socket.IO** orquesta datos y distribuye estado hacia TD y los demás clientes.

Alcance (Scope)

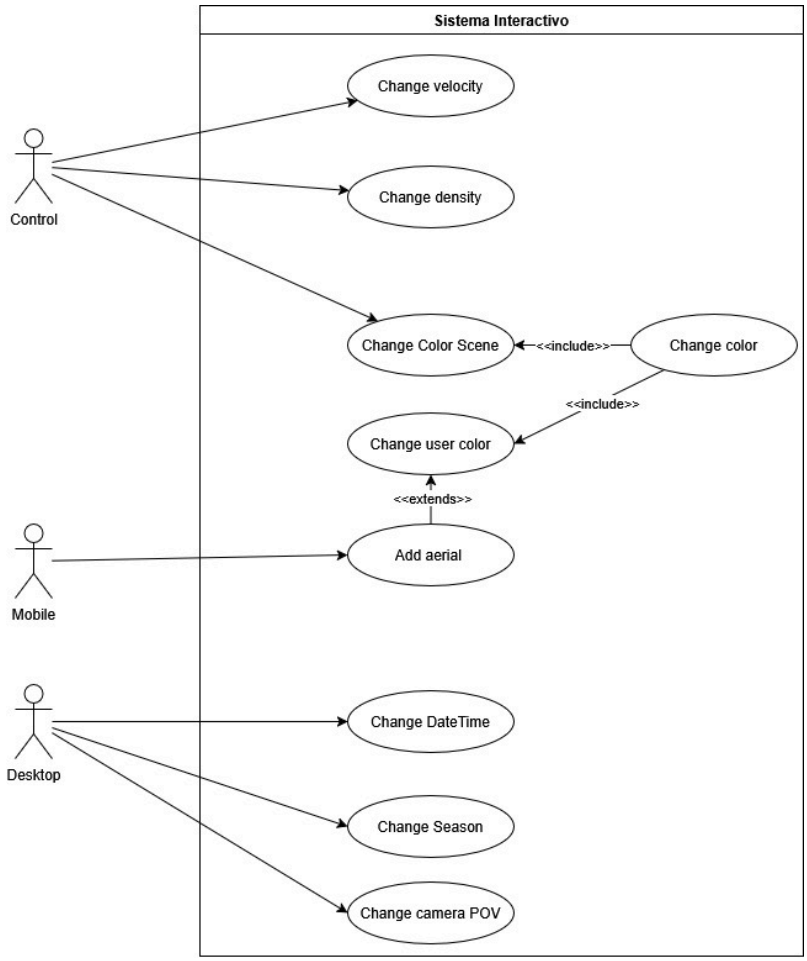
Incluye:

- Recolección de inputs (sliders, toggles, color pickers) desde clientes web.
- Normalización y difusión del estado global y eventos incrementales.
- Consumo en TD para render de visuales.
- No incluye:
- Persistencia en BD, autenticación de usuarios, ni panel de administración.

Actores

- **Participante (MobileClient):** selecciona su color (HEX/RGB).
- **Operador de escena (Control):** ajusta speed, density y color maestro.
- **Operador de contexto (DesktopClient):** define time (day/night) y season (1–4).
- **Render Engine (TouchDesigner):** consume estado/eventos para generar visual.
- **Servidor (Socket.IO):** actor técnico que intermedia y consolida.

Diagrama de casos de uso



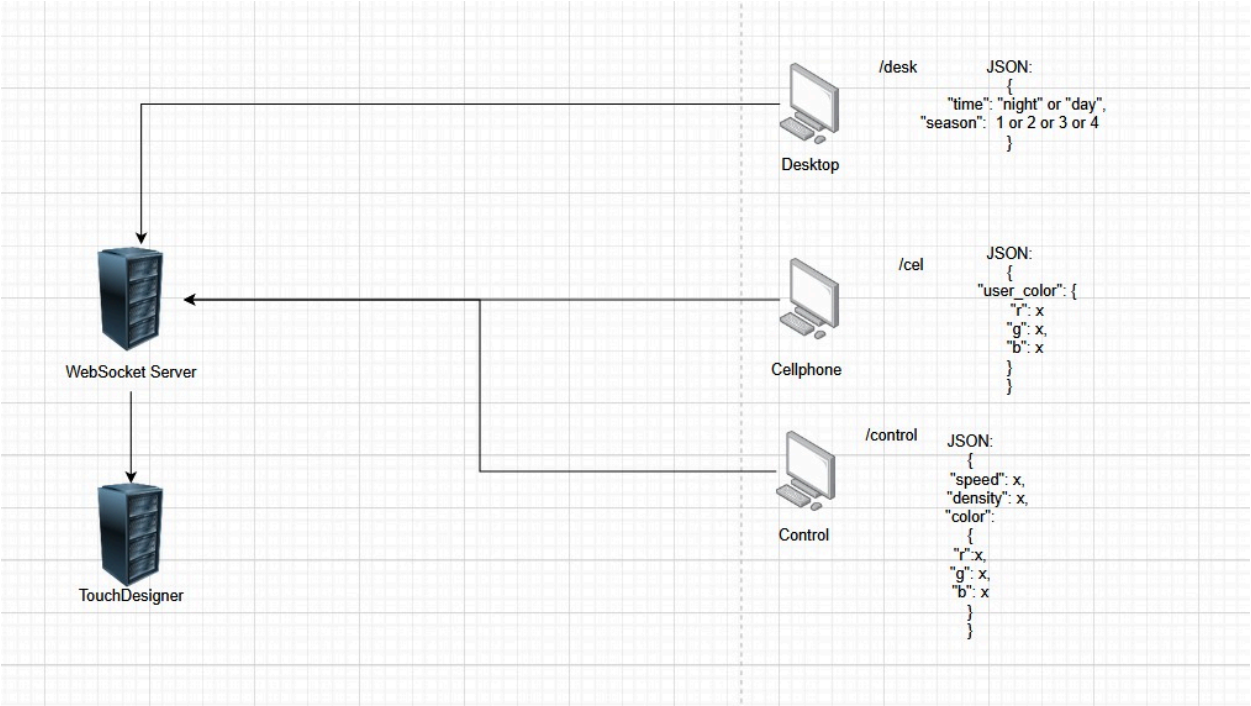
Casos de uso

ID	Nombre	Actor principal	Objetivo	Disparador	Precondiciones	Resultado esperado
CU-1	Ajustar parámetros globales	Control	Cambiar `speed`, `density` y `color` maestro de la escena	Usuario mueve sliders/selector en Control	Cliente Control conectado al servidor	Server emite `slider_changed` (a Visuales room) y `state`; TD actualiza visual en < 100 ms
CU-2	Definir contexto de la escena	Desktop	Cambiar `time` (day/night) y `season` (1–4)	Usuario alterna opciones en Desktop	Cliente Desktop conectado	Server actualiza `context` y emite `state`; los clientes quedan sincronizados

CU-3	Aportar color de usuario (antena)	Mobile	Enviar color del participante (`mobile:colorHex`/ `mobile:colorRgb`)	Usuario mueve el color picker	Cliente Mobile conectado	Server crea/actualiza la antena en `aerials`; emite `state`; TD reescribe tabla de antenas
CU-4	Sincronizar estado inicial	Cualquiera (todos)	Recibir el estado completo al conectarse	Nueva conexión de socket	Servidor operativo	Server envía `whoami` y `state:init`; cliente ajusta UI y valores locales
CU-5	Consumir estado en TouchDesigner	TouchDesigner (TD)	Actualizar tablas/CHOPs y parámetros visuales	Recepción de `state`/`state:init`/eventos	TD conectado y unido a “Visuales room” (si aplica)	TD ejecuta callbacks: reescribe `aerials_table`, ajusta parámetros y refleja cambios en la visual
CU-6	Unirse a canal de visuales	TouchDesigner (TD)	Recibir solo eventos relevantes a visualización	TD emite `messageClienteVisuales`	Conexión de TD activa	Server añade el socket a "Visuales room"; TD recibe `slider_changed` y `state` específicos
CU-7	Desconexión de participante	Mobile	Remover antena asociada al socket	Cierre/caída de conexión	Antena registrada en `aerials`	Server elimina la entrada del mapa y emite `state`; TD refleja decremento del conteo/tabla

Analisis Técnico

Diagrama de arquitectura de redes



El sistema implementa una **arquitectura cliente–servidor en tiempo real** basada en WebSockets (Socket.IO). El **WebSocket Server** actúa como **hub** que recibe entradas desde tres tipos de clientes (Desktop, Control y Cellphone) y **difunde** el estado hacia **TouchDesigner (TD)** y hacia los demás clientes que lo requieran. Así se sincronizan, en milisegundos, los parámetros de escena y color que alimentan las visuales generativas.

Componentes y roles:

Desktop (/DesktopClient):

Define el contexto de escena. Envía JSON con: { "time": "night|day", "season": 1|2|3|4 }
Estos valores cambian el modo visual (p. ej., paletas, densidad base, presets).

Control (/Control):

Es el panel maestro de parámetros globales. Publica continuamente:

```
{
  "speed": x,
  "density": x,
  "color": { "r": x, "g": x, "b": x }
```

}
Además emite eventos incrementales slider_changed para respuesta inmediata en TD.

Cellphone (/MobileClient):

Representa a cada participante. Envía su color de usuario: { "user_color": { "r": x, "g": x, "b": x } }
El servidor agrega estas entradas (mapa/tabla de antenas) y las redistribuye.

TouchDesigner (TD):

Se suscribe a los eventos del servidor y consume el estado consolidado para renderizar las visuales. Lee:
Estado completo (state / state:init): control + lista de usuarios/antenas.
Eventos incrementales (slider_changed, color, etc.) para cambios fluidos.

WebSocket Server (Socket.IO)

Orquesta las conexiones y mantiene un estado en memoria con:
Control (speed, density, color global)
MobileClient (colores por socket/cliente móvil)
Reemite a “rooms” específicos (p. ej., Visuales room) para no saturar a todos los clientes.
Expone endpoints estáticos para servir cada cliente web.

Flujo de datos (resumen)

1. DesktopClient y Control emiten cambios → Servidor actualiza estado → difunde a TD y a los clientes que escuchan.
2. Cada Cellphone publica su color → Servidor agrega y transmite una tabla/array de usuarios → TD actualiza las antenas/partículas.
3. TD no envía control; solo recibe y visualiza (en este demo).

Estados clientes:

Mobile: selector de color (HEX/RGB); muestra su ID y estado.
Control: sliders speed, density, color maestro; depurador JSON.
Visuals: monitor de estado;UI mínima para verificar la activación de escenas.
Desktop: vista del estado/escena para operador.

Ventajas de la topología

- Baja latencia y sincronía entre múltiples clientes.
- Se pueden añadir más tipos de clientes o eventos sin alterar TD.
- Aislamiento por rooms para optimizar tráfico.
- Extensible a red local o internet.

Contrato de eventos (Socket.IO)

Origen → Destino	Evento	Payload (forma)	Descripción / Uso principal
Servidor → Cliente/TD	`whoami`	`{ id: string }`	Asigna/retorna el `socket.id` del cliente al conectar.
Servidor → Cliente/TD	`state:init`	`{ control: { speed:number, density:number, color:{r,g,b} }, aerials: Array<Aerial> }`	Estado inicial completo al conectar.
Servidor → Cliente/TD	`state`	`{ state: { control: {...}, aerials: Array<Aerial> } }`	Broadcast de estado actualizado (control y móviles).
Servidor → Cliente/TD	`color`	`{ type: "color", x: number /* rojo */ }` *Opcional:* `{ id, r, g, b }`	Evento incremental de color para compatibilidad con TD.
Control → Servidor	`slider_changed`	`{ label: "speed" | "density" | ..., value: number }`	Cambio de un slider; el servidor lo reenvía a “Visuales room”.
Control → Servidor	`update`	`{ speed:number, density:number, color:{ r:number, g:number, b:number } }`	Envía el estado completo del panel de control.
Mobile → Servidor	`mobile:colorHex`	`"#rrggbb"`	Cambia color del móvil usando HEX.
Mobile → Servidor	`mobile:colorRgb`	`{ r:number, g:number, b:number }`	Cambia color del móvil usando RGB.
TD → Servidor	`messageClienteVisuales`	*(sin payload)*	Hace que el socket de TD se una a la sala "Visuales room".

Troubleshooting

- **TD no recibe nada:** URL en SocketIO DAT debe ser http://localhost:3000 y Active: On.
- **No aparecen móviles:** revisa consola del server (creación de aerial); verifica que entraste por /mobile.
- **Sliders no llegan a Visuals:** ¿socket.emit('slider_changed', {...})? ¿TD unido a “Visuales room” en onOpen()?
- **Latencia alta:** cerrar pestañas innecesarias, usar red local en lo posible 5GHz, limitar FPS en TD si el CPU está al 100%.