Martín Emiliano Lombardo
mlombardo9@gmail.com
+54 9 11 2170-2812

KeepSmiling Visor 3D

18 de abril del 2020

FUNCIONAMIENTO

El proyecto tiene como fin permitir la subida de un conjunto de modelos 3D que forman una animación de cada maxilar junto con información relevante del paciente. El sistema generará un enlace público no deducible donde se podrá acceder al contenido.

El sistema va a contar con un par de páginas de administración, una para subir casos nuevos y otra para listar los casos ya existentes. En esta lista se podrá filtrar por nombre y ID. También habrá un botón para eliminar casos.

ESPECIFICACIONES

El stack del sistema será Node de back y front será web plana (esto es, sin React o similares). El sistema tendrá su propia base de datos SQL, estará hosteado en DigitalOcean y utilizará Spaces (de DO). El visor estará hecho en JavaScript y utilizará Three.JS.

Las páginas de administración estarán bajo un formulario de login simple HTTP.

La página del visor según lo hablado con ustedes proveerán el diseño. Las páginas de administración tendrán un diseño simple hecho por mi.

El procesamiento de los archivos ocurrirá en el servidor y no se almacenarán los STL/VRML/JPG. Una vez subidos los archivos se procesarán (ver anexo PROCESAMIENTO DE ARCHIVOS) y se almacenarán los archivos crudos en el CDN. Esta es la misma técnica que usa el sistema que tienen actualmente, ya que reduce **drásticamente** el peso de los archivos que se tienen que descargar, no obstante la página va a seguir siendo pesada. (No hice los números pero podría decir que hacer esto reduce el tamaño en más de un 80%). Una vez procesado los archivos originales STL/VRML/JPG se eliminan del servidor (si los quieren conservar por alguna razón me avisan).

OBJETIVOS

1. Hacer el script para procesar archivos. Idealmente va a ser standalone para poder probarlo sin tener que depender de la subida.
2. Hacer el servidor mínimo para que levante el visor (sin DB en este punto) usando un mock.
3. Hacer el visor mínimamente funcional, que se pueda mover la cámara y se puedan avanzar las vistas (animación).

   (esto sería el mínimo viable)

4. Agregar las posiciones de los maxilares (abierto, cerrado), la grilla, los botones, el diseño en general.
5. Crear la base de datos y hacer los endpoints para subir archivos y levantar casos.
6. Maquetar la página de administración.
7. Hacer que el front suba los archivos a la API.

   (acá ya estaría prácticamente funcional)

8. Hacer la página de administración para listar casos, agregar los endpoints para eliminar y filtrar.
9. Retoques que puedan surgir.

TIEMPOS

Producto mínimo viable: que a partir de todos los archivos se generen los datos crudos y se visualicen en el visor (sin CDN, quizás sin iluminación): 1-2 semanas

Proyecto totalmente funcional: 4-6 semanas.

REQUISITOS

1. Acceso SSH a un droplet vacío.
2. Access key de Spaces (**debe ser creado en modo privado**, usen el domain custom si quieren).
3. El dominio que utilizará la página (y los DNS apuntados correctamente).
4. El diseño de la página del visor.
5. OPCIONAL: si que tienen un servidor Git y quieren que trabaje con un repositorio ahí enviarme unas credenciales, sinó utilizaré GitHub y les puedo transferir el repositorio cuando ya esté finalizado.
6. OPCIONAL: formato de preferencia para los enlaces/páginas/ID (letras / números / UUID)

7. OPCIONAL: códigos de seguimiento como Google Analytics o similares (igualmente una vez entregado van a poder modificarlo como quieran)

OTROS TEMAS

El tema de Stripping habría que verse, si me pasan el formato con el que se carga lo vemos.

MODALIDAD DE PAGO

Presupuesto

Desarrollo completo: $ 150.000,00.

Forma de pago

A definir.

ANEXO: PROCESAMIENTO DE ARCHIVOS

En principio hay que detectar qué archivo es cual de la lista de nombres que viene. Como entendí no se pueden mezclar archivos VRML y STL en el mismo caso, lo que hace mas facil separarlos y no mezcla las cosas (con/sin colores reales).

Entiendo que el programa los exporta con este formato o similar:

XXXXX_1era etapa_mover_d-m-YYYY__hh-mm-ss_subsetupX_mandibulary.vrml/jpg

XXXXX_1era etapa_MOVER_d-m-YYYY__hh-mm-ss_SubsetupX_Mandibular.stl

Estos son los formatos que me enviaron, voy a asumir que subsetup**X**_mandibulary identifica el modelo X-iésimo de la animación. Y de ahí asocio los JPGs etc.

Decidir cómo conviene divir los archivos crudos, si por frame de animación o por maxilar (o quizás en grupos). El sistema que ya tenían los dividía por frame de animación. Esto significa que hay un archivo crudo por cada frame de animación. Si se hace por maxilar sería descargar un solo archivo y correrse mediante índices.

Con la división definida arriba efectivamente leer los archivos y separar los datos en canales.

Quizás hay mas pero por los datos que ví debería haber como mínimo en cada vértice:

XYZ (coords en 3D) NXNYNZ (normales, para la luz) RGB (lo rojo / verde) RGB (el color real)

Incluyo el color real directamente en la información de los vértices para ahorrar enviar cada imagen al cargar la página, se hace un sample mientras se procesa y se guarda el color directamente.

Además de esto hay que incluir los índices.

También hay que decidir si van a estar intercalados (los vertices intercalados) en un mismo archivo o un archivo separado por cada canal. En el que ya tienen los separa por maxilar y por canal, me parece mejor idea intercalar los canales y juntarlos en un mismo archivo.

Una vez listo los archivos crudos se suben a DO y se eliminan los archivos fuente.

Nota: el formato con el que se generan los archivos crudos va a estar documentado en el código