**Modelado**:

1) Se siguieron varios tutoriales para el modelado de una persona. <http://cgi.tutsplus.com/es/categories/modeling> Exponer modelos seguidos para el cuerpo y para la cara.

2) El pelo se descargó de una web libre de derechos de autor Licencia: CC-BY-SA <http://www.blendswap.com/blends/view/63151>

3) Para texturizar se utilizó UV Mapping y la herramienta Mark Seam para que al hacer Unwrap las partes de la malla se viesen correctamente. El unwrap se exportó y con Photoshop se rellenó la imagen con las texturas deseadas.

**Animación**

1) Se intentó varias veces con resultados negativos el generar un riggin. Finalmente se creó un esqueleto (pero no se generó riggin) y en cada hueso (del layer adecuado) se copió la rotación de cada hueso respectivo del esqueleto animado. El banco de animaciones utilizado fue: <https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/daz-friendly-release>

Y se utilizaron las animaciones: 127\_07 para correr, 127\_26 para salto, 132\_22 para andar, 137\_28 para esperar.

2) Las animaciones fueron editadas en bvhacker donde se centraron y se editó el loop (recortándolo y cosiéndolo) para obtener solo la parte deseada de la animación y poder reproducirla en bucle.

3) Para la exportación se utilizó el exportador blendert->Json desarrollado por el equipo de Three.js versión r77. Dio varios problemas al estar en desarrollo y se tuvo que cambiar una línea de código del exportador porque las coordenadas Y de la malla en blender se exportaban como coordenadas Z. Por lo que la malla salía fuera del esqueleto y se deformaba al animar.

**Aplicación web cliente**

1) Para la realización de la aplicación web del lado del cliente se ha de manejar el modelo 3d respondiendo a las teclas del usuario. Para ello se ha utilizado el ejemplo de personajes y animaciones que viene en Three.js llamado “*webgl\_animation\_skinning\_blending*” y se ha empezado a partir de él. Se cambió el modelo por el propio y se establecieron todas las funciones de control (cambio de un estado a otro, comprobación de posición, ayuda al centrado, etc) se amplía la clase BlendCharacter.js para acceder a más funciones de las animaciones de manera ordenada.

2) Se crea un plano que tiene como textura el video entrante de la cámara web.

3) La comunicación con arduino es Ajax->PHP->Python->Arduino

4) Para el video se crea un plano en el fondo el cual tiene como textura una imagen. Cada pasada de render esta imagen se cambia por la siguiente. Las imágenes se van sustituyendo a medida que son tomadas por el script raspiMJPEG.

5) Cuando estemos con un móvil aparecerán controles alternativos(flechas vsg). Para saber si estamos con un móvil se usará media query de CSS.

**Raspberry**:

1) Se ha instalado VNC para su control desde internet y otro ordenador. Para ello <https://www.raspberrypi.org/documentation/remote-access/vnc/README.md> Se ha creado el script para lanzar el vnc al enchufarla.

1.2) Establecer IP estática para que siempre sea 192.168.1.137

2) Instalado servidor web nginx con PHP: <https://www.raspberrypi.org/documentation/remote-access/web-server/nginx.md>

3) Se instala el IDE de arduino y se configura

4) Hay que darle permisos para el usuario www-data en el directorio del index y añadir el usuario www-data al grupo ‘dialout’ para que este pueda controlar los puertos seriales. Reiniciar después de añadir le usuario al grupo.

5) Se consigue encender y apagar un led mediante una web.

6) Descargar el proyecto en raspberry mediante Git. Trabajo en el portátil y subo archivos con fileZilla.

7) Hay que instalar raspiMJPEG. Este comando toma fotografías cada x tiempo y las guarda en ram para que se pueda acceder a ellas rápidamente. Estas se usarán para crear el video. Para empezar a tomar las fotografías habrá que poner el comando: *raspimjpeg* \*<https://www.raspberrypi.org/forums/viewtopic.php?t=61771>   
\*<http://elinux.org/RPi-Cam-Web-Interface#RaspiMJPEG>