Escuela Politécnica Superior

Control remoto de realidad aumentada mediante un robot

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Miguel Soriano Sanz

Tutor/es:
Mireia Luisa Sempere Tortosa

Septiembre 2016

Universitat d'Alacant
Universidad de Alicante

Resumen

Prueba.

Justificación y objetivos

Prueba.

Agradecimientos

Prueba.

Citas

Prueba.

Índice de contenidos

1. Resumen
2. Justificación y objetivos
3. Agradecimientos
4. Citas
5. Índice de contenidos
6. Índice de figuras
7. Introducción
8. Marco teórico
    - 8.1. Lenguajes de programación web
        - 8.1.1. HTML 5
        - 8.1.2. Javascript
        - 8.1.3. PHP
    - 8.2. Realidad aumentada
        - 8.2.1. RA en web y WebGL
    - 8.3. Modelado(Blender)
    - 8.4. Arduino
    - 8.5. Raspberri Pi
        - 8.5.1. Raspbian como SO
        - 8.5.2. Python
        - 8.5.3. Servidores web para raspberry
            - 8.5.3.1. Apache
            - 8.5.3.2. lighttpd
            - 8.5.3.3. Nginx
    - 8.6. Estado de arte
9. Objetivos
    - 9.1. Desarrollo del personaje
    - 9.2. Desarrollo de la aplicación web
    - 9.3. Desarrollo del robot
    - 9.4. Realidad aumentada
10. Metodología
    - 10.1. Metodología del desarrollo software
    - 10.2. Gestión del proyecto
        - 10.2.1. Repositorios y GitHub
        - 10.2.2. Trello
        - 10.2.3. Toggle
11. Desarrollo
    - 11.1. Desarrollo del personaje
        - 11.1.1. Bocetos
        - 11.1.2. Modelado
        - 11.1.3. Animación
    - 11.2. Desarrollo de la aplicación web
        - 11.2.1. Cliente
        - 11.2.2. Servidor

11.3. Desarrollo del robot
11.3.1. Raspberry
11.3.2. Arduino
11.4. Realidad aumentada
12. Trabajo futuro
13. Conclusiones y valoración personal
14. Bibliografía

Índice de figuras

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. Una petición POST a PHP se ejecuta para cargar la nueva imagen. Esto pasa cada 4 pasadas del render, es decir 60/4 = 15 veces por segundo

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

6) Se completa la API con el robot (controlRobot.php y Robot.js) con las instrucciones de acelerar, parar, girarIz, girarDe y retroceder. Estas son llamadas mediante Ajax según el estado del personaje.

7) Se añade la sombra del personaje. Para ello se utiliza un plano con ShadowMaterial. Además hay que: activar sombras en render, poner plano como recibidor de sombras y poner personaje como creador de sombras, además de crear una luz y posicionarla.

8) Se llama a la función de comprobar distancia 60 veces por segundo (cada vuelta de la animación) Pero solo se realizará la función cada dos segundos gracias a un contador que comprueba que ha sido llamada 120 veces. También se puede forzar la llamada. Cuando se llama se actualiza la distancia y la posibilidad de andar del vehículo. Si la llamada sale mal se fuerza otra llamada.

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

Se eligió nginx entre apache y lighttpd por su velocidad:
https://www.jeremymorgan.com/blog/programming/raspberry-pi-web-server-comparison/

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

Arduino:

1) Se monta el chasis del robot

2) Se crea el programa en arduino para controlar el robot. Según el carácter que le entre por el puerto serial ejecutará una de las acciones (avanzar, girar, retroceder o parar). La comunicación con los motores se hace mediante una shield de motores(L298N) conectada a pilas. Para que el robot tenga fuerza, cada vez que se mueven los motores arrancan a máxima velocidad para el primer impulso y después bajan de velocidad para un mayor control del robot.

Fuente shield: https://www.bananarobotics.com/shop/How-to-use-the-L298N-Dual-H-Bridge-Motor-Driver

3) Conexiones en el robot y arduino:

4) Comprobación de distancia: El arduino ejecuta el sensor ultrasónico para saber la distancia cuando el cliente se lo pide al servidor.