API de TrackerPod
(movimiento de la cámara web)

Realizada por:

JESÚS MANUEL RODRÍGUEZ SÁNCHEZ

Proyecto VIGIA

Índice

- Software actual de control para TrackerPod
  - PTZdriver
  - TrackerCam
  - Comparativa entre ambos programas
- ¿Qué se puede hacer con la API?
- Métodos de comunicación con TrackerPod
  - Peticiones/Respuestas al servidor web
  - Usando la librería C++ / C#
  - Usando el componente COM
- Descripción del control a través de peticiones HTTP POST
  - Formato de las peticiones POST
  - Ejemplo de petición POST de control

Software actual para TrackerPod

- PTZdriver
- Software de control básico para la base móvil TrackerPod

Software actual para TrackerPod

- TrackerCam
- Software de control avanzado que trabaja conjuntamente con la base móvil TrackerPod y con la cámara web usada

Software actual para TrackerPod

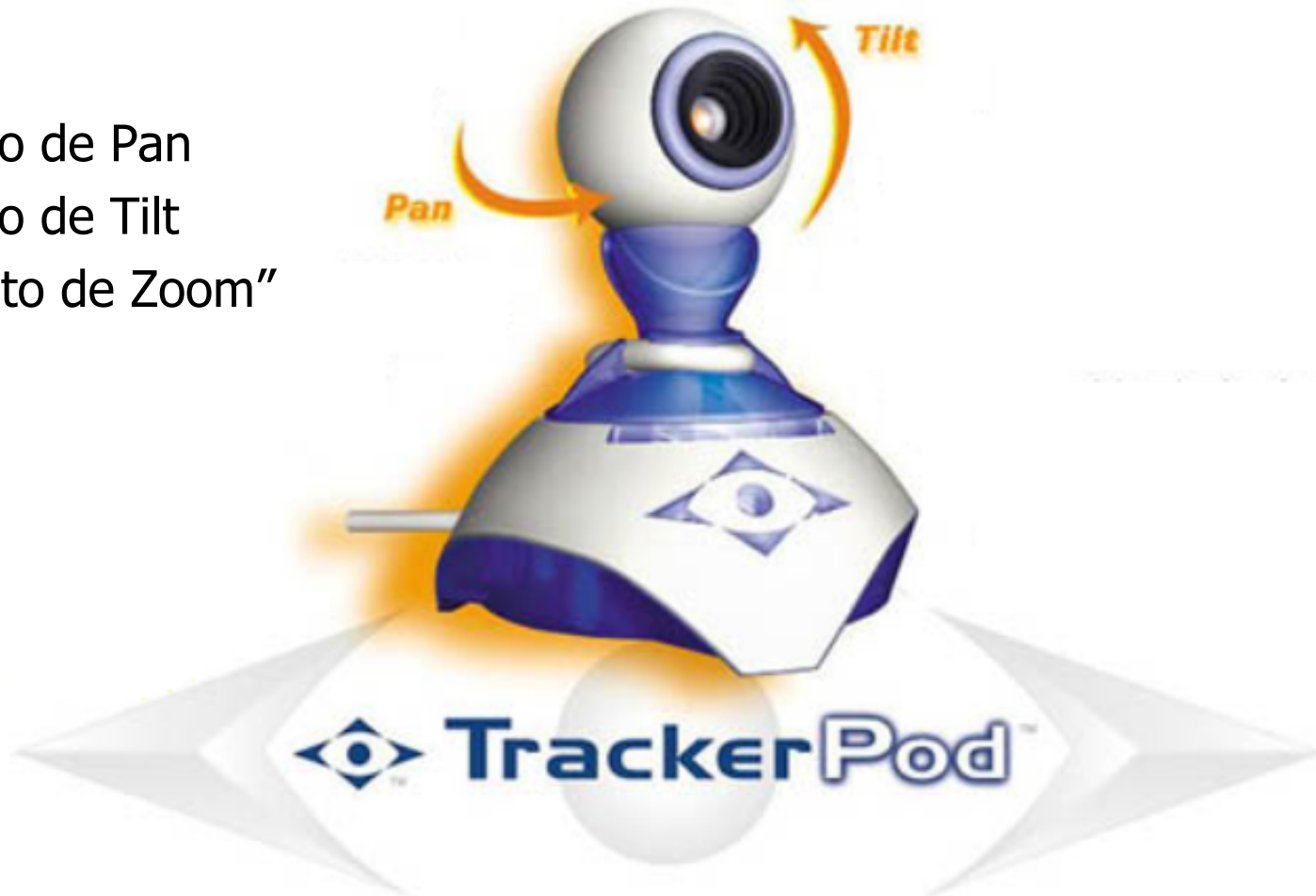
- Comparativa entre PTZdriver y TrackerCam

Funciones	TrackerCam	PTZdriver
Funciona con cámaras web/cámaras de video	Sí/Sí	No/Sí
Control Pan/Tilt/Zoom	Sí/Sí/Sí	Sí/Sí/Sí
Seguimiento de Movimiento/Personas	Sí/Sí	No/No
Servidor de video en página Web	Sí	No
Captura de vídeo/imágenes	Sí/Sí	No/Sí
Subida de imágenes a servidores FTP	Sí	No

¿Qué se puede hacer con la API?

- Funcionalidad para controlar movimientos de TrackerPod

  - Movimiento de Pan
  - Movimiento de Tilt
  - "Movimiento de Zoom"



Métodos comunicación TrackerPod

- Peticiones/Respuestas HTTP POST:
- Método de control más simple para manejar la base TrackerPod
- Este método de control se puede realizar desde cualquier lenguaje de programación
- Las peticiones POST desencadenan la ejecución de funciones PHP por parte del servidor integrado en TrackerPod

Métodos comunicación TrackerPod

- Peticiones HTTP POST para el manejo de imágenes:

```
<form name="Stills" action="RemoteProcessN.trackercamimage"
method="post" target="Invisible">
<input type="hidden" name="UserID">
<input type="hidden" name="Busy">
<input type="hidden" name="RefreshFlag">
<input type="hidden" name="Frame">
<input type="hidden" name="ConsoleType" value="Full">
<input type="hidden" name="ConsoleSize" value="Standard">
<input type="hidden" name="RefreshRate">
<input type="hidden" name="WhereIsTheFunctions" value="top.Main">
</form>
```

Métodos comunicación TrackerPod

- Peticiones HTTP POST para control PTZ:

```
<form name="Controls" action="RemoteProcessN.trackercamcontrol"
method="post" target="Invisible">
<input type="hidden" name="XPos">
<input type="hidden" name="YPos">
<input type="hidden" name="UserID">
<input type="hidden" name="RequestType">
<input type="hidden" name="StepSize">
<input type="hidden" name="WhereIsTheFunctions" value="top.Main">
</form>
```

Métodos comunicación TrackerPod

- Peticiones HTTP POST para control PTZ:

- Posibles valores para las variables

```
XPos={H(0) o un número entero}
YPos={H(0) o un número entero}
ZPos=[10..500] -> Normal 100
UserID=<nombre_usuario>
RequestType={move, zoompos}
StepSize=[1..7]
WhereIsTheFunctions={unknown}
```

Métodos comunicación TrackerPod

- Usando la librería C++ / C#:

- Existe una versión simple de la librería (una sola hebra) y una versión multihebra

- ¿Cómo usar la librería?
  - Ejecutar el programa "`trackerpod_server.exe`"
  - Incluir el fichero de cabeceras "`TrackerPod.h`"
  - Enlazar el proyecto con el fichero "`TrackerPod_ml.lib`" / "`TrackerPod_mt.lib`"

- Internamente la librería funciona a través de la ejecución del programa llamado "Administrador de TrackerPod"

Métodos comunicación TrackerPod

- Usando la librería C++ / C#:

```
class CTrackerPod
{
public:
CTrackerPod() { }
virtual ~CTrackerPod() { }
virtual bool initialize(char *client_id, _dev_change_notify pfn,
unsigned long user_data, bool b_show_on_event)
virtual bool begin_enum_device()
virtual bool enum_reset()
virtual long enum_next(char *ver)
virtual bool use_device(long devid)
virtual bool move_to(float x, float y)
virtual bool move_by(float x, float y)
virtual long control(char cmd[1024])
virtual bool get_pos(float *x, float *y)
virtual bool get_info(char info[1024])
virtual void show_on_event(bool show)
virtual void show_manager(bool show)
virtual bool is_manager_visible()
virtual void ReleaseTrackerPod()
```

Métodos comunicación TrackerPod

- Usando el componente COM de TrackerPod:
- Disponible para proyectos C++ / Visual Basic
- ¿Cómo usar la librería?
  - Ejecutar el programa "`compodsrv_setup.exe`" para instalar el componente "`trackerpod_com.dll`"
  - Crear una instancia del componenete
  - Usar la interfaz `ITrackerPodC` del componente
- Internamente el componenete funciona a través de la ejecución del programa llamado "Administrador de TrackerPod"

Métodos comunicación TrackerPod

- Usando el componente COM de TrackerPod:

```
interface ITrackerPodC : IUnknown
{
virtual HRESULT __stdcall initialize(BSTR app_name) = 0;
virtual HRESULT __stdcall move_to(short x, short y) = 0;
virtual HRESULT __stdcall move_by(short x, short y) = 0;
virtual HRESULT __stdcall get_pos(short *x, short *y) = 0;
virtual HRESULT __stdcall devcontrol(BSTR cmd, long
*result) = 0;
virtual HRESULT __stdcall get_info(BSTR *pVal) = 0;
};
```

Descripción de control HTTP POST

- Peticiones HTTP POST para control PTZ:

- Formato de una petición HTTP POST

```
POST <nombre_accion> HTTP/1.0
User-Agent: <nombre_agente_usuario>
Accept: <formato_archivos_acceptados>
Content-type: <tipo_datos>
Content-length: <longitud_datos>

variableX=<valorX>&variableY=<valorY>&…
```

Descripción de control HTTP POST

- Peticiones HTTP POST para control PTZ:

- Ejemplo de una petición HTTP POST para TrackerPod

```
POST /RemoteProcessN.trackercamcontrol HTTP/1.0
User-Agent: TrackerCamHttp/1.0
Accept: www/source; text/html; image/gif; */*
Content-type: application/x-www-form-urlencoded
Content-length: 89

XPos=0&YPos=0&ZPos=100&UserID=Guest&RequestType=mov
e&StepSize=1&WhereIsTheFuncions=unknown
```

Descripción de control HTTP POST

- Peticiones HTTP POST para control PTZ:
- Proceso de comunicación