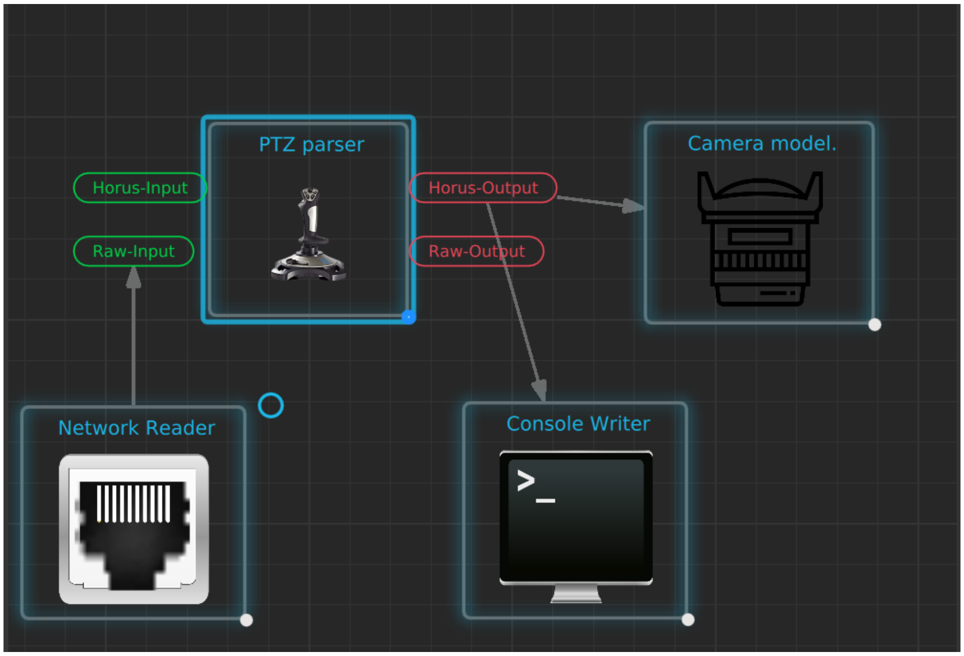
HORUS VIEW & EXPLORE

TECHNICAL DOCUMENTION

ASCII PTZ Protocol

Horus Embed
embed@horus.nu

Horus View & Explore
info@horus.nu



January 30, 2020

Contents

1 Introduction: ASCII PTZ Protocol

This relatively small document describes the **ASCII** ptz protocol which can be used with Horus View & Explore applications and services.

2 Horus View & Explore ASCII PTZ Protocol

2.1 Introduction: ASCII PTZ Protocol

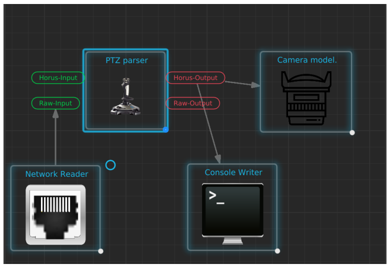
The ASCII protocol was designed to help software engineers in testing PTZ controllable domes.
Often when working on site or at home, one needed to generate joystick or button commands in order to engage a certain behavior. In most cases no physical device was available, and since few people can write binary protcols from the top of their heads a human readible format was needed. Thus we created a small and easy to remember protocol allowing simple commands such as:

```
LEFT 0.1
OFFSET ABSOLUTE_OFFSET UNITS RADIANS
YAW 10
UNITS NORMALIZED
UP −1
OFFSET ABSOLUTE_OFFSET UNITS RADIANS PITCH 1
```

2.1.1 In practice: connectivity

There are several ways to insert and extract PTZ commands within the Horus environment. The two that we will focus on are serial and network connectivity.

2.1.2 Network connectivity



Connecting a **Network Reader** { Protocol: TCP, Behavior: Read } to a **Ptz parser** {incomming protocol: ASCII} allows for ASCII/PTZ over Network communication.

Using simple commandline utilities such as netcat allows messages to be send to the Network Reader.

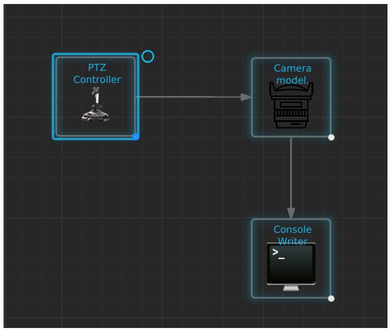
```
$>> nc -4 localhost 5567
OFFSET ABSOLUTE_OFFSET UNITS RADIANS
YAW 10
PITCH 3
```

2.1.3 Serial connectivity

In order to use serial communication as a testing input, virtual serial port pairs needs to be configured.
On windows platforms, the *http://com0com.sourceforge.net* can be used.
For Linux users we recommend the *socat* utility (make sure you have the correct rights or add the user to the "dialout" group).

```
$>> socat PTY,link=/dev/ttyS98 PTY,link=/dev/ttyS99
```

Setting up a **PTZ controller** { incoming protocol: ASCII, outgoing protocol: ASCII } allows for ASCII/PTZ over serial communication.



2.2 The protocol parser.

As we have seen in the connectivity section, the ASCII protocol parser is embedded in several different components. The ASCII protocol parser is implemented as a state-machine, meaning that the following statement has no direct effect:

```
OFFSET ABSOLUTE_OFFSET UNITS RADIANS
OFFSET ABSOLUTE_OFFSET UNITS RADIANS
```

It does make sense however to start your session by providing:

```
OFFSET ABSOLUTE_OFFSET UNITS RADIANS
```

after which all orientation based commands will be of the later configuration. After which commands such as:

```
YAW 180
PITCH 10
ZOOM 45
```

can be issued. The latter statements will be interpreted as:

```
OFFSET ABSOLUTE_OFFSET UNITS RADIANS YAW 0.2
OFFSET ABSOLUTE_OFFSET UNITS RADIANS PITCH 0.8
OFFSET ABSOLUTE_OFFSET UNITS RADIANS ZOOM 0.25
```

2.3 The protocol definition.

Setting the camera offset.		
OFFSET	RELATIVE_OFFSET	ABSOLUTE_OFFSET

Setting the units.			
UNITS	RADIANS	DEGREES	NORMALIZED

Orientation commands, example LEFT 0.1			
LEFT	RIGHT	UP	DOWN
YAW	PITCH	ROLL	ZOOM_TELE
ZOOM_WIDE	ZOOM	FOCUS_FAR	FOCUS_NEAR
FOCUS			

The stop command
STOP

Best practice combinations		
TOKEN	OFFSET	UNITS
LEFT	RELATIVE_OFFSET	RADIANS
RIGHT	RELATIVE_OFFSET	RADIANS
UP	RELATIVE_OFFSET	RADIANS
DOWN	RELATIVE_OFFSET	RADIANS
YAW	ABSOLUTE_OFFSET	RADIANS
PITCH	ABSOLUTE_OFFSET	RADIANS
ROLL	ABSOLUTE_OFFSET	RADIANS
ZOOM_TELE	RELATIVE_OFFSET	RADIANS
ZOOM_WIDE	RELATIVE_OFFSET	RADIANS
ZOOM	ABSOLUTE_OFFSET	RADIANS
FOCUS_FAR	RELATIVE_OFFSET	RADIANS
FOCUS_NEAR	RELATIVE_OFFSET	RADIANS
FOCUS	ABSOLUTE_OFFSET	RADIANS