# picoFlamingo Alpha 0.4. User Manual

Codename: **"The Cracking Egg"**

Edition 0 Rev 1 DRAFT by DMO

June 4, 2011


picoflamingo@papermint-designs.com

http://www.papermint-designs.com/picoflamingo

http://www.papermint-designs.com/community/?q=blog/3

.

# Contents

# 1 Introduction

picoFlamingo is a lightweight presentation system intended to be deployed on embedded systems and therefore be really portable. The currently chosen hardware platform for the development is the BeagleBoard (http://beagleboard.org), as processing unit, and the Texas Instruments picoDLP projector.

The small power fingerprint and size of booth system makes them a very good tandem for the picoFlamingo project.

The intended audience of the system are professional that need to present their works in an attractive way to small audiences, anywhere at any time. Specially the ones that need 3D capabilities to show, for instance, their product concepts (no physically available) or products too huge for a meeting room.

This document covers the release of the second alpha version of the system (code name "The Cracking Egg"), including system installation, configuration and operation.

## 1.1 Features of this Release

The features of picoFlaming alpha 0.4, "The Cracking Egg", release are:

- The following slide items are allowed:
  - Text. Several different True-Type fonts can be used to add text to the slides.
  - Image. Several different image formats can be used to add images to the slide
  - 3D Models in 3DS file format.
  - Live video streams
- All item in slide are visualised in a 3D space. Position and Orientation can be changed for any item
- In Slide animation of any item
- Predefined slides transitions
- Slide show mode
- Different Remote Control options
- OpenGL ES 2.0. Frame buffer and X11 versions.
  - Remote control through Bluetooth or TCP/IP (wired or wifi depending on configured hardware).
  - Simple remote control application for Symbian Devices using Python S60, allowing the navigation throughout the presentation
  - Simple remote control application for FreeRunner OpenMoko and Linux boxes with X11/SDL graphics and Bluetooth or Networking capabilities. This application allows rotation of the selected item in the presentation.
  - Simple Android remote control version. Not included in distribution.
- Additional tools and utilities
  - Simple Voice commanding solution for navigating presentations
  - Simple Video streamer for v4l2 devices
  - Basic button agent for building simple interfaces

Figure 1: The “Cracking Egg” system at Papermint Designs Lab

# 2 System Overview

This section describes the hardware and software environment required by the system as well as the installation process for the "The Cracking Egg" different components.

## 2.1 Software Components

The picoFlamingo "Cracking Egg" is composed of the following minimal software components:

- **The NetKitty communications tool**. This is an small program that allows Bluetooth and TCP connectivity. The picoFlamingo application uses it and the redirection of the standard input to get remote commands. The extended remote control application, uses it and the redirection of standard output to send control messages to the presentation.
- **The picoFlamingo application**. This is the main application in charge of controlling the presentation. The picoFlamingo application renders the slides and exposes remote commands to control the navigation through the presentation.

  This application needs the Netkitty communication tool for remote operation.
- **The simplified remote control application for Python S60**. This is a simple python script that can be executed in Symbian S60 devices and uses the Bluetooth interface for navigating a picoFlamingo "Cracking Egg" presentation remotely. This application only supports moving to the next and previous slide.
- **The extended remote control application for OpenMoko FR and Linux Boxes**. This is an SDL application providing extended functionalities for remote controlling the picoFlamingo "Cracking Egg". In addition to forward and backward slide navigation, this application allows to rotate a given object (this is defined in the presentation itself) within the presentation in real-time.

  This application needs the Netkitty communication tool for remote operation.

## 2.2 Tested Hardware

The "Cracking Egg" has been tested using the following hardware:

- The picoFlamingo "Cracking Egg" has been tested on the following hardware
  - BeagleBoard Revisions B6 and C3.
  - x86 PC compatible computers.
  - Omap Zoom II
  - OMAP4 Blaze
- The extended Remote Control application has been tested on the following hardware
  - x86 PC compatible computers.
  - Omap Zoom II
  - OpenMoko FreeRunner
- Nokia N70 Symbian device

  Runs the simplified remote control application for Python S60.
- picoDLP projector

  Used for portable projection of the presentation from the BeagleBoard.

- USB Bluetooth dongle.

  Used for remote wireless Bluetooth connectivity.

As an alpha release no extensive testing has been done, however the system should work with other configurations without major problems.

## 2.3 Software Requirements

The "Cracking Egg" needs the following additional software components to run.

- OpenGL ES 2.0 support. The OpenGL ES 2.0 user libraries (and hardware drivers) for the Beagle board are required. To run the application in a x86 PC an OpenGL ES 2.0 emulation layer, as for instance the Imagination Technologies SDK, is also required.
- The freetype library.
- The SDL library for PC and OpenMoko to run the extended remote control application on these platforms.
- The Python S60 Interpreter to use the Remote Symbian remote control.
- The OM.2008 OpenMoko distribution. For using the extended remote contrl application with the OpenMoko FR.

The "Cracking Egg" has been tested with the following BeagleBoard software environment in order to get the different components (USB, SGX, etc...) working all together:

- X-Loader 1.4.2
- U-Boot 2009.01-dirty
- Linux Kernel 2.6.28

For supporting on installation check the picoFlamingo forums in the Papermint Designs Community (http://community.papermint-designs.com).

The U-Boot configuration tested is shown below:

```
bootcmd=if mmcinit; then if run loadbootscript; then run bootscript; else if run loaduimage; then if run loadramdisk; then r
bootdelay=10
baudrate=115200
loadaddr=0x80200000
rdaddr=0x81600000
console=ttyS2,115200n8
mmcargs=setenv bootargs console=${console} ${optargs} root=/dev/mmcblk0p2 rw rootfstype=ext3 rootwait
ramargs=setenv bootargs console=${console} ${optargs} root=/dev/ram0 rw ramdisk_size=32768 initrd=${rdaddr},32M
ubifsargs=setenv bootargs console=${console} ${optargs} root=ubi0:beagleroot ubi.mtd=4 rw rootfstype=ubifs
jffs2args=setenv bootargs console=${console} ${optargs} root=/dev/mtdblock4 rw rootfstype=jffs2
loadbootscript=fatload mmc 0 ${loadaddr} boot.scr
bootscript=echo Running bootscript from mmc ...; autoscr ${loadaddr}
loaduimage=fatload mmc 0 ${loadaddr} uImage.bin
loadramdisk=fatload mmc 0 ${rdaddr} ramdisk.gz
ramboot=echo Booting from ramdisk.gz ...; run ramargs; bootm ${loadaddr}
mmcboot=echo Booting from mmc ...; run mmcargs; bootm ${loadaddr}
nandboot=echo Booting from nand ...; run jffs2args; nand read ${loadaddr} 280000 400000; bootm ${loadaddr}
usbtty=cdc_acm
serial=5ac400030000000004013f8901001001
optargs=video=omapfb:vram:4M omapfb.video_mode=640x480MR-16@60
stdin=serial
stdout=serial
stderr=serial

Environment size: 1333/131068 bytes
```

## 2.4 Angstrom Images

The picoFlamingo "Cracking Egg" compiles and runs out of the box with the latest Angstrom images, including the images created by the Narcissus on-line image generator.

Those images has been used for the BeagleBoard and the Omap Zoom II

## 2.5 OMAP 4

The picoFlamingo "Cracking Egg" compiles and runs out of the box on OMAP 4 platform running Ubuntu 10.10 once the software dependencies listed above are installed.

# 3 Installation

The installation processes for the different system components are described below:

## 3.1 Installation on Beagle Board/OMAP Zoom II

In order to install "The Cracking Egg" in a BeagleBoard or OMAP Zoom II device a fully bootable system is required, including the software dependencies indicated on section 2.3.

The simpler way to build that base system is to generate an Angstrom image using Narcisus. Angstrom images already include the OpenGL ES drivers and libraries required to run picoFlamingo "Cracking Egg"

Once the system is ready, you only need to compile the "Cracking Egg" sources. Please refer to section "Compiling"

Note that for using picoFlamingo "The Cracking Egg" with X11, the X11 development package is required.

## 3.2 Installation in an x86 PC

- Install of OpenGL ES 2.0 development platform. The latest version of MESA can be used.

  http://www.mesa3d.org/

  Alternatively other solutions like the Imagination Technologies POWERVR SDK. It can be downloaded from here:

  http://www.imgtec.com/powervr/insider/sdk/KhronosOpenGLES2xSGX.asp

  Requires registration.

  Make sure the libraries can be found by the dynamic linker (add the path to `LD_LIBRARY_PATH` or update your `/etc/ld.so.conf`.

- Untar the "Cracking Egg" source code tarball in your preferred location. It can be downloaded from:

  http://papermint-designs.com/picoflamingo/download.html

- Make sure that the `libfreetype.so` library is available in your system

## 3.3 Installation of the Symbian Remote Control application

- Install Pyhton for S60 in your Symbian Device
- Edit the `000-pf.py` script to update the Bluetooth address you want to connect to (yes, this is an early alpha release). This file is available for downloading from here:

  http://papermint-designs.com/picoflamingo/download.html

- Install the script in your Symbian Device.

## 3.4 Installation of the Extended Remote Control Application

This application uses SDL and needs the NetKitty communication tool to run.

- Make sure that the SDL libraries are available for your platform (OpenMoko, Pc, etc...)
- Copy the right binary for the chosen platform from:

  http://papermint-designs.com/picoflamingo/download.html

This application is also distributed as source code and, therefore can be recompiled for the users if the provided binary files does not work for their platforms.

- Install the SDL development packages for your chosen platform
- The source code is included in the "Cracking Egg" source code distribution under apps/pf-control.
- Recompile the application issuing the `make` command.

## 3.5 picoFlamingo Compilation

picoFlamingo the "Cracking Egg" includes a `Makefile` that expect the environment variable `PLATFORM`, to be set to one of the following values:

- fb. This will compile picoFlamingo to run on frame buffer
- x11. This will compile picoFlamingo to run on X11

In the picoFlamingo source distribution there are two subdirectories called, respectively, `fb` and `x11`. They contains the code for the frame buffer and the X11 initialisation.

Additionally, if your EGL and OpenGL ES libraries and includes are not accessible on the standard locations (/usr/include and /usr/lib), they have to be added in the respective subdirectories under `fb` or `x11`, depending on the desired solution.

For BeagleBoard and OMAP Zoom II Angstrom images, the development files should be accessible and no further action is required for compiling. If special version of the libraries and headers are copied under `beagle` or `pc` subdurectories, then the `LD_LIBRARY_PATH` environment variable has to be properly set to allow the system to locate the libraries.

The previous paragraph also applies to OMAP4 Ubuntu distributions.

## 3.6 User Feedback

Please, let us know the details of your hardware platform and if you could successfully run picoFlamingo 0.0 the "Cracking Egg", or any problem or troubleshooting action you performed to make the system work. You can use the forums in http://community.papermint-designs.com/ to report any issue.

# 4 Running The Pink Egg

This section explains how to use the "Cracking Egg" and how to produce your own presentation.

## 4.1 Command Line Interface Magic

This alpha release of picoFlamingo supports the following command line flags:

- `--w`. Specifies the width of the screen by default it is set to 640.
- `--h`. Specifies the height of the screen by default it is set to 480.
- `--slideshow`. Activates the slideshow mode. In this mode, the "Pink Egg" will cycle through all the presentation slides automatically, waiting arounf 10 secs on each one.
- `--dir dirname`. Specifies the directory where the slide files can be found. Normally these directories will hang from the "Cracking Egg" installation directory.
- `--reload slide_num`. Activates the autoload mode. In this mode the slide slide_num will be reloaded each 4 seconds. This mode is intended for basic editing of slides.**This feature is currently broken. Please do not use**
- `--goto n`. Specifies the slide to load, where n is the slide number
- `--identity`. Specifies a "name" for the current picoFlamingo instance. These names allows to filter commands when several picoFlamingo instances are controlled from a single NetKitty hub
- `--postproc`. Specifies a post processing shader to be applied as the latest rendering step. This post processing shader can be used, for instance, for geometric distortion corrections.

The "Cracking Egg" doesn't includes networking code. The different commands used to control the presentation are read from the standard input. Therefore, to support remote operation throughout the network, the "Cracking Egg" uses netkitty to take care of network connections. The typical execution commands for the "Cracking Egg" are:

- Unattended slide show

  ```
  ./picoFlamingo_beagle --slideshow --dir my_presentation
  ```

- Edition of slide 5

  ```
  ./picoFlamingo_beagle --dir my_presentation --reload 5
  ```

- Simple stdin control.

  ```
  ./picoFlamingo_beagle --dir my_presentation
  ```

- Remote BlueTooth RFComm Connections on channel 1

  ```
  ./nk --server B,1 | ./picoFlamingo_beagle --dir my_presentation
  ```

- Remote TCP/IP Control accepting connections on port 5000

  ```
  ./nk --server T,5000 | ./picoFlamingo_beagle --dir my_presentation
  ```

- Remote TCP/IP and Bluetooth Control together

  ```
  ./nk --server T,5000 B,4 | ./picoFlamingo_beagle --dir my_presentation
  ```

- Distortion correction using post-processing filter

  ```
  ./nk --server T,5000 B,1 | ./picoFlamingo_beagle --postproc proc02 --dir my_presentation
  ```

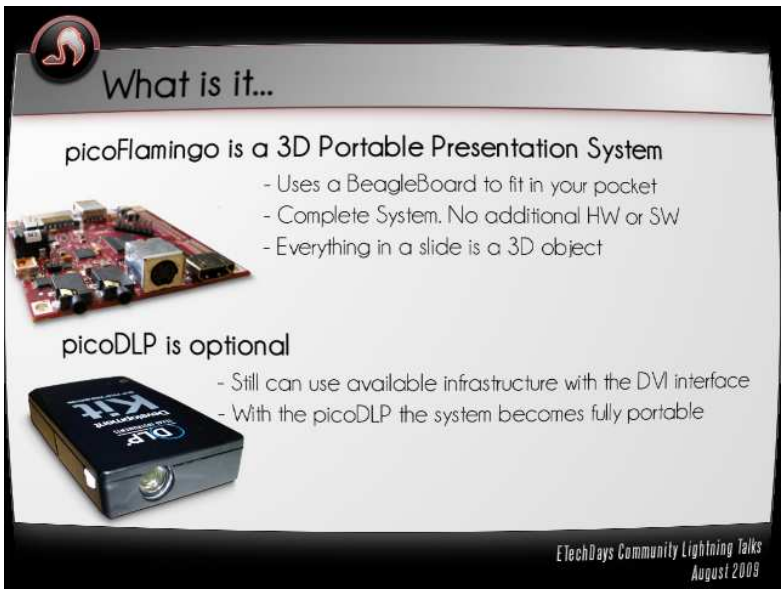


Figure 2: The "Cracking Egg" post processing shader. Distortion correction

## 4.2 Adding Text Styles

The "Cracking Egg" provides a very simplistic style management system to allow the users include their own fonts in their presentations.

Adding a new style to the presentation is a very simple process.

- Add your True-Type font file (.ttf) into the fonts directory.
- Update the `font_list.txt` file. This file is in the same directory that the "Cracking Egg" main application. The format for this file is as follows:

  `FontPointSize StyleName TruTypeFile`

  where:

  - `FontPointSize` is the size in points of the font. The bigger this number the better the quality of the text in the presentation, but also, the most memory used. A value of 32 is normally enough to achieve an affordable quality.
  - `StyleName` is the user name for this style. This name will be used in the slides to select the text style for the text items.
  - `TrueTypeFile` is the name of the ttf file in the fonts directory to use for this style.

# 5 Running the Remote Control Applications

This section explains how to run the different remote control solutions for picoFlamingo "Cracking Egg".

## 5.1 Command Line Control

The picoFlamingo "Cracking Egg" can be controlled from the command line using the commands described in section 8.12. These commands can be issued in the following ways.

- Using a communication tool like Netkitty or Netcat and connecting to one of the external interfaces provided by the system. These interfaces can be a TCP (wired or wifi) connection or a bluetooth RFComm connection (this only using NetKitty).
- Directly typing in the console where the application was launched if available.

## 5.2 Simple Remote Control for Python S60

This application connects directly to the Bluetooth address hard coded in the script. It read the device cursor keys. The right cursor key sends a next slide command, and the left cursor key sends a previous slide command.

## 5.3 Extended Remote Control Application

This application can be executed in different platforms once installed as explained in 3.4. This application needs a communication tool to remotely connect to the picoFlamingo "Cracking Egg" system.

To achieve this goal, the application should be launched using a command similar to the ones below:

**Connection using TCP networking**

```
pf-control | nk -client T:pink_egg_ip:port
```

**Connection using BlueTooth**

```
pf-control | nk -client B:pink_egg_bt_addr:channel
```

The specific TCP port or Bluetooth channel will depend on the way the picoFlamingo "Cracking Egg" is started.

Once started the application will show an screen like the one on the left of Figure 3.

The screen is divided in three main zones.

- The upper part of the screen shows some descriptive information about the tool. Clicking or touching is part of the screen will close it.
- The lower part of the screen provides two navigations buttons to move forward and backwards through the presentation.
- Finally, the central part of the screen is used to interact with the slide item having the focus. Pressing in this area will send a rotation command to the picoFlamingo "Cracking Egg" and the current focused item will be rotated accordingly.

  The interface only sends X and Y rotation angles. This angles are calculated proportionally to the distance in the X and Y axis between the center of the screen and the point the user clicked or touched the screen.

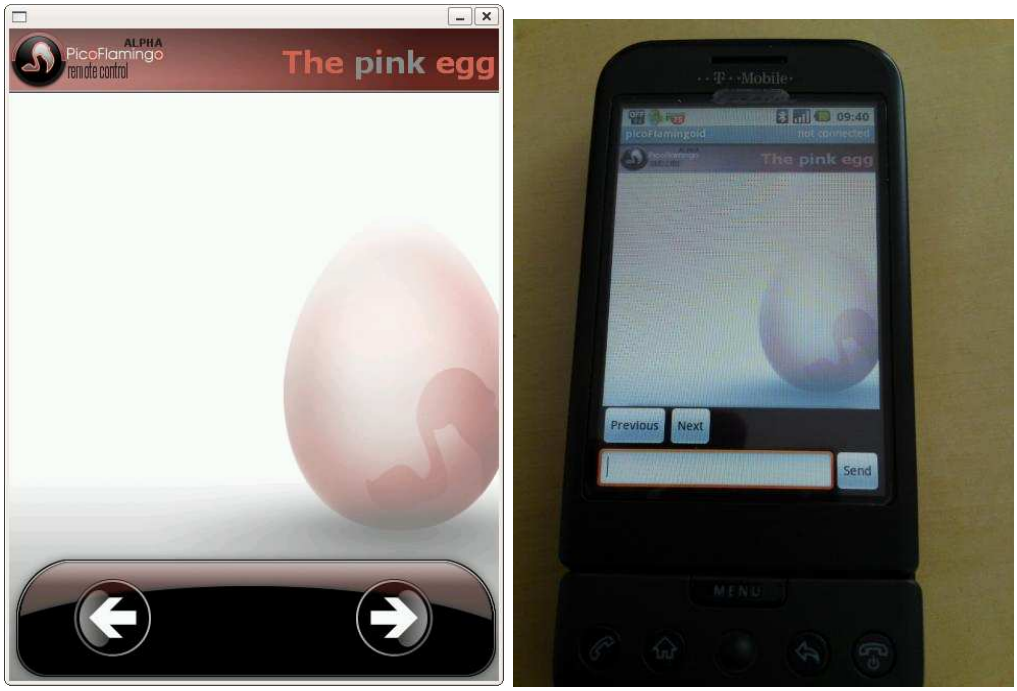


Figure 3: Extended Remote Control Application (left) and early Android prototype -picoFlamingoid- (right)

This interface is being extended to support several new features that are being added and will change in future, including more different operational models.

This interface has been successfully tested with the picoFlamingo "Cracking Egg" from an x86 PC and from a Free Runner OpenMoko phone.

There is also a preliminary version of an Android application for remote controlling picoFlamingo. The prototype is shown on the right side of Figure 3. This application can also be downloaded from the picoFlamingo website, but it is not included in the source code distribution.

# 6 Video Streamer Applications

picoFlamingo the "Cracking Egg" includes a simple video streamer application to feed real-time live video (from a webcam for instance) into a picoFlamingo image item in a slide.

This video streamer uses libjpeg to compress the captured images and works with v4l2 devices.

## 6.1 Installation

picoFlamingo video streamer is distributed as source code. In order to compile the application the v4l2 headers are needed and the development package of libjpeg needs to be installed in the system.

Once these requirements are fulfilled simply move to `app/video_server` and execute `make`. Then the produced executable needs to be copied to some directory included in the system `PATH` variable.

## 6.2 Command line arguments

The video streamer application distributed with "The Cracking Egg" accepts the following command line arguments:

- `--device name` specifies the v4l2 device to be use for acquiring images. Default value is `/dev/video0`
- `--width w` specifies the width of the images to be captured
- `--height h` specifies the height of the images to be captured
- `--port p` specifies the TCP port the server will accept connections to transmit the captured images
- `--format n` specifies the image format delivered by the v4l2 device. A value of 0 means that the images are delivered as compressed JPEG streams. A value of 1 means that the images are delviered as YUYV. The video streamer application will convert, in any case, the acquired images from the v4l2 devices into a JPEG coded image.

## 6.3 Usage

Please refer to section 8.10 for details on how to feed images from the video streamer into a picoFlamingo "The Cracking Egg" image item.

## 6.4 Hardware Supported

The video streamer application distributed with picoFlamingo "The Cracking Egg" should work with any v4l2 compliant device. The following devices have been successfully used with the application:

- Logitech Webcam Pro 9000 (uvc driver) (YUYV format)
- OMAP Zoom II built-in camera (YUYV format)

# 7 Voice Commanding Applications

picoFlamingo "The Cracking Egg" includes a voice commanding application based on Julius voice recogniser. This application allows to navigate a presentation using the voice commands `NEXT` and `PREVIOUS`.

## 7.1 Installation

picoFlamingo voice commanding application is distributed as source code. In order to compile the application the Julius speech recogniser and as suitable sound device for the target platform needs to be already configured.

Once these requirements are fulfilled simply move to app/speech directory and execute `make`. Then the produced executable needs to be copied to some directory included in the system `PATH` environmental variable

## 7.2 Extra data files

The voice commanding tool included with picoFlamingo needs an acoustic model and grammar to effectively recognise the speaker voice. These files can be obtained from the quick start package distributed by voxforge.org:

http://www.voxforge.org/home/downloads#QuickStart%20Anchor

Then follow the instructions below:

- Download the quickstart package and uncompress in your system.
- Copy the grammar file distributed with picoFlamingo into the `grammar` directory
- Run the command `./bin/mkdfa.pl grammar/sample`. This will process the picoFlamingo grammar and produce the files required by Julius to interpret the navigation commands

In order to execute the application run the following command in the quick start package directory:

```
$ ./julius-simple -input mic -C julian.jconf
```

## 7.3 Usage

The voice commanding tool relies on the NetKitty application to connect to picoFlamingo. The usual way to execute the tools is with the following command line:

```
$ ./julius-simple -input mic -C julian.jconf | nk -client localhost 5000
```

This command starts the voice commanding application using the local microphone and the grammar produced in the previous section, and forwards the recognised commands to post 5000 in the local machine.

Then picoFlamingo should be launch together with NetKitty listening in port 5000.

For instructions on how to setup this tool to get the audio from a different machine, please check this site:

http://papermint-designs.com/community/?q=node/61

For Julius/audio related troubleshooting please refer to the appropriated forums.

# 8 Creating Presentations

In this release, a presentation is a sequence of slides. Each slide is stored in a separated file named as '`slideN.pfs`, where N is the slide number. In order to create a presentation, the user has to create a subdirectory and put on it one of these slideN.pfs files for each slide s/he wants to show.

As explained above, the `--reload` command line flag is very useful when creating a presentation as it will show the changes made in the slide file as they are made.

**NOTE. The –reload flag is broken on current release**

Therefore, in order to create a presentation the format of the slides (the .pfs file formats) needs to be known.

The rest of this section describes this file format.

## 8.1 Comments

The .pfs file formats considers a comment any line starting with the character ';'. Note that this character HAS TO be in the first column of the line to comment out (no other character before it).

**Example**

```
; This is a .pfs comment.
```

## 8.2 Slide Items

This section describes the main items that can be added to a slide.

### 8.2.1 BACKGROUND

The command `BACKGROUND` sets up the presentation background. The background has to be an image in one of the supported formats. Slide transition effects does not apply to the item index 0. In general the `BACKGROUND` command should be the first one in order to not be affected by slide transitions.

**Syntax**

`BACKGROUND image_file`

where

- `image_file`, image file to add as a presentation background.

The background item has the additional property of not being affected for the transition effects.

**Example**

```
; Background
BACKGROUND background3.png
```

### 8.2.2 ADD_TEXT and ADD_STEXT

The `ADD_TEXT` command adds a dynamic text item to the current slide. The `ADD_STEXT` command adds a static text item to the current.

A static text item can be rendered faster but it cannot be updated once initialised. Dynamic text items can be updates at any time, changing any of their properties, including their content.

**Syntax**

`ADD_TEXT text_style`

where

- `text_style` is one of the valid text styles defined in the `font_list.lst` file for the current presentation.

To change general properties for this item check the section 8.6.

The `ADD_TEXT` command supports the following subcommands to add extra information to the item:

- `ADD_DATA`. This subcommand adds a new text line to a text item.
- `SET_TEXT_INTERLNE`. This subcommand modifies the inter line separation for the current text item.
- `SCALE`. This subcommand modifies the size of the current text item. The expected value is the divisor for the text sizes.

**Example**

```
ADD_TEXT text01
; Text Data
SET_TEXT_INTERLINE 0.6
SCALE 100.0
ADD_DATA SLIDESHOW:
ADD_DATA EDIT SLIDE 5:
; General Item Properties
POSITION -3.0 1.2 -3.5
COLOR 0.9 0.9 0.9 0.9
```

## 8.3 ADD_IMAGE and ADD_CENTERED_IMAGE

The `ADD_IMAGE` and `ADD_CENTERED_IMAGE` commands adds an image to the current slide.

**Syntax**

`ADD_IMAGE scale imagefile`

where

- `scale` is the size of the image and
- `imagefile` is the image file to add.

The `ADD_IMAGE` maps the given image as a texture into a QUAD running from coordinates (0,0) to (1,1). The `ADD_CENTERED_IMAGE` commands does the same but using a QUAD from (-0.5, -0.5) to (0.5, 0.5).

To change general properties for this item check the section 8.6.

**Example**

```
; Image
ADD_IMAGE 2.5 beagle.png
POSITION -2.8 -2.2 -3.0
```

## 8.4 ADD_CUBE

The `ADD_CUBE` command adds a 3D cube model to the current slide.

### Syntax

```
ADD_CUBE sizeX sizeY sizeZ
```

where

- `sizeX` is the cube size in the X axis.
- `sizeY` is the cube size in the Y axis.
- `sizeZ` is the cube size in the Z axis.

To change general properties for this item check the section 8.6. *Note: The color property cannot be changed for 3D Models in this version*

### Example

```
;Interactive Cube
ADD_CUBE 1.5 1.0 1.0
SET_FOCUS
POSITION -0.0 -0.2 -3.0
ROTATION 0.7 0.2 0.0
```

## 8.5 ADD_MODEL

The `ADD_MODEL` command adds a 3D model to the current slide.

### Syntax

```
ADD_CUBE model_file
```

where

- `model_file` is the file name of a 3D model on 3DS format

To change general properties for this item check the section 8.6. *Note: The color property cannot be changed for 3D Models in this version*

### Example

```
;Interactive Cube
ADD_MODEL 3D1.3d
SET_FOCUS
POSITION 0.0 0.2 -3.0
```

## 8.6 General Item Properties

This section describes the commands used to set general item properties.

## 8.7 NAME

The `NAME` command allows to provide a symbolic name to the item currently being defined. This name can be used to give the item the focus or to reference it with the FX commands

**Syntax**

`NAME name`

where

- `name` is a string.

**Example**

```
ADD_CUBE 1.5 1.0 1.0
NAME my_cube
```

### 8.7.1 POSITION

The `POSITION` command allows the user to set the position in the slide for the current item.

**Syntax**

`POSITION X Y Z`

where

- `X Y Z` are the 3D position for the current item.

**Example**

```
ADD_CUBE 1.5 1.0 1.0
POSITION -0.0 -0.2 -3.0
```

### 8.7.2 ROTATION

The `ROTATION` command allows the user to set the rotation in the slide for the current item.

**Syntax**

`ROTATION A B C`

where

- `A B C` are the 3D rotation w.r.t. the X, Y, Z axis for the current item. Angles are in radians.

**Example**

```
ADD_CUBE 1.5 1.0 1.0
POSITION -0.0 -0.2 -3.0
ROTATION 0.7 0.2 0.0
```

### 8.7.3 COLOR

The `COLOR` command allows the user to change the color of a given item. *Note: this command doesn't work on 3D models in this release.*

**Syntax**

`COLOR R G B ALPHA`

where:

- `R G B, ALPHA` is a 4D vector specifying the color to apply, including the alpha channel.

**Example**

```
ADD_TEXT text07
ADD_DATA ... much more coming soon!!!
POSITION -2.0 -2.6 -3.5
SCALE 100.0
COLOR 0.1 0.1 0.1 0.8
```

## 8.8 Other Commands

This section introduce other commands not directly related to the slide items.

### 8.8.1 INCLUDE

The `INCLUDE` command allows the inclusion of an external file containing picoFlamingo "Pink Egg" commands in the current slide.

**Syntax**

`INCLUDE template_file`

where:

- `template_file` is the file to include in the current slide.

**Example**

```
; Include common header for presentation
INCLUDE header.pfi
```

### 8.8.2 SET_FOCUS

The `SET_FOCUS` command, indicates to the system that the given item has the focus and will receive any interactive command from the remote system interfaces.

The `SET_FOCUS` command can only be applied after the slide is defined (dynamically at run-time) or between a `FX_START`, `FX_END` block within the slide

**Syntax**

`SET_FOCUS my_cube`

**Example**

```
; Remote Controlled Spining Cube
ADD_CUBE 1.5 1.0 1.0
NAME my_cube
POSITION -0.0 -0.2 -3.0
ROTATION 0.7 0.2 0.0
FX_START
SET_FOCUS my_cube
FX_END
```

### 8.8.3 GENERIC_EVENTS

The `GENERIC_EVENTS` command indicates to picoFlamingo if the mouse vents received on X11 mode should be interpreted as commands for the current item with the focus or should be forwarded to other application using `stdout`.

**Syntax**

`GENERIC_EVENTS n`

where:

- `n` can be 0 (events manipulate current slide item) or 1 (events are forwarded to stdout)

### 8.8.4 SHOW

The `SHOW` command indicates to the system if a given slide item has to be shown or hidden.

**Syntax**

`SHOW name n`

where:

- `name` represents the name of the item to show or hide
- `n` can be 0 (item is hidden) or 1 (item is shown)

**Example**

```
; Remote Controlled Spining Cube
ADD_CUBE 1.5 1.0 1.0
NAME my_cube
POSITION -0.0 -0.2 -3.0
ROTATION 0.7 0.2 0.0
FX_START
SHOW my_cube 0
FX_END
```

## 8.9 Text Manipulation Commands

The following commands can be used to manipulate the dynamic text (text items created with command `ADD_TEXT`) rendering.

- `SET_TEXT_INTERLINE name interline` .

  Changes the interline space between the lines of the `name` slide text item

- `UPDATE_TEXT_INTERLINE name interline`.

  Dynamically changes the interline space between the lines of the `name` slide text item.

- `UPDATE_FONT name font_name`.

  Dynamically updates the associated font to the `name` slide text item.

- `UPDATE_TEXT_SCALE name scale_x scale_y`.

  Dynamically changes the X and Y scale for the slide text item `name`

- `UPDATE_TEXT_WIDTH name width`.

  Dynamically updates the fixed width associated to `name` text item in the slide

- `UPDATE_TEXT name new_text`.

  Dynamically changes the content of `name` text item.

- `TXT_SET_LINE name num_line`.

  Sets the initial line for a multi-line text item.

- `TXT_SET_LINES_UP name lines_up`.

  Indicates how many lines up of the current line will be rendered.

- `TXT_SET_LINES_DOWN name lines_down`.

  Indicates how many lines down of the current line will be rendered.

## 8.10 Image Manipulation

The following command can be used to manipulate images

- `SET_IMAGE name file_name`. Allows to load a new image file into the image slide item `name`. **Current version requires the new image to be the same width and height that the current one**
- `CONNECT name ip port`. Connect to a video streamer server (Section 6) and starts rendering the video stream in the image item `name`. **Current version requires the video stream an the target image to have the same width and height**
- `DISCONNECT name`. Disconnect image `name` from any video streamer associated to it.
- `SHOT_WIDTH_NAME name file_name`. Saves current video stream frame associated to image `name` into a JPEG file with the provided name.

## 8.11 Special Effects and Animation

The following command can be used to animate slide items

- `FX_START`.

  Starts the FX definition section in a slide file.

- `FX_END`.

  Indicates that the FX section in the slide file ends

- `FX_POS name frame_0 duration X0 Y0 Z0 X1 Y1 Z1`.

  Initiates a position animation on item `name`, starting at frame `frame_0` relative to the time the comment was issued, for `duration` frames, moving the item from position `(X0,Y0,Z0)` to position `(X1,Y1,Z1)`

- `FX_MOVE_TO name frame_0 duration X1 Y1 Z1`.

  Initiates a position animation on item `name`, starting at frame `frame_0` relative to the time the comment was issued, for `duration` frames, moving the item from its current position to position `(X1,Y1,Z1)`

- `FX_ROT name frame_0 duration H0 P0 R0 H1 P1 R1`.

  Initiates a rotation animation on item `name`, starting at frame `frame_0` relative to the time the comment was issued, for `duration` frames, rotating the item from angles `(H0,P0,R0)` to angles `(H1,P1,R1)`

- `FX_ROT_TO name frame_0 duration H1 P1 R1`.

  Initiates a rotation animation on item `name`, starting at frame `frame_0` relative to the time the comment was issued, for `duration` frames, rotating the item from its current orientation to orientation `(H1,P1,R1)`

- `FX_COLOR name frame_0 duration R0 G0 B0 A0 R1 G1 B1 A1`. Initiates a color animation on item `name`, starting at frame `frame_0` relative to the time the comment was issued, for `duration` frames, changing item from color `(R0,G0,B0, A0)` to color `(R1,G1,B1, A1)`

- `FX_COLOR_TO name frame_0 duration R1 G1 B1 A1`.

  Initiates a color animation on item `name`, starting at frame `frame_0` relative to the time the comment was issued, for `duration` frames, changing the item from its current color to color `(H1,P1,R1)`

In addition to the frame based animation commands described above, a time-based version of all those commands is also available. For these time based version, the parameters `frame_0` and `duration` should be identified as initial time (in milliseconds) and duration (also in milliseconds

The time based animation commands are, respectively: `FX_TPOS`, `FX_TROT`, `FX_TCOLOR`, `FX_TMOVE_TO`, `FX_TROT_TO` and `FX_TCOLOR_TO`.

## 8.12 Controlling the Presentation

The "Cracking Egg" release only supports three control commands. This commands can be issued directly from the console (if netkitty is not used), or remotely through bluetooth or TCP/IP networking (using netkitty or netcat).

The available commands are:

- **NEXT**: Goes to next slide in presentation.
- **PREV**: Goes to previous slide in presentation.
- **SLIDESHOW**: Switches ON/OFF slideshow mode.
- **ROT X Y Z**: Rotates the current presentation item with respect to the three main axis.

# 9 Known Issues

This is a very early alpha version that is very buggy and it is only intended to give future users a feeling of what picoFlamingo can offer. This version is simply a quick integration of the different tests carried out so far and, therefore should be considered as a macro test.

In any case, there are some already know issues with this version.

- There is no proper error management. If something is wrong (a very big image, a wrong scale value or an improper SGX configuration) the application will simply crash.
- The specific configuration for U-Boot and Linux kernel seems to be a bit tricky and we were only able to make it work with the configuration stated above.
- The `ROTATION` command has some issues with image transparency and doesn't rotates properly text items.
- The texture manager object has not yet been added so a texture is created for each single image used (even when used several times), so images should be used carefully.
- U-Boot 2009.1-dirty is required for USB Host operation on BeagleBoard C3. This version seems to no longer take the environment from NAND so a simple `saveenv` do not work, when using U-Boot from the SD card.

  In order to pass the proper command line to the kernel to initialise the graphics in 16bpp mode, the U-Boot booting process can be interrupted from the console, the proper variable set and the `boot` command used for one-time booting. This has to be done each time the system is rebooted.

  The easiest way to avoid this is to Flash U-Boot to NAND so the `saveenv` command will work properly. Otherwise a proper U-Boot script needs to be provided

# picoFlamingo project

PORTABLE 3D PRESENTATION SYSTEM

http://papermint-designs.com/picoflamingo/
http://community.papermint-designs.com/
picoflamingo [AT] papermint-designs [.] com