stages:

- build

- deploy

- run

variables:

CI\_SCRIPTS\_DIR: "./ci"

macos build:

stage: build

script:

- MATLAB\_SCRIPT=$(pwd)/build\_executables.m

- MATLAB\_EXE=/Applications/MATLAB\_R2017a.app/bin/matlab

- XCODE\_EXE=/Applications/Xcode7.3.1.app/Contents/Developer

- env DEVELOPER\_DIR=$XCODE\_EXE $CI\_SCRIPTS\_DIR/build-mac.sh $MATLAB\_EXE $MATLAB\_SCRIPT

artifacts:

paths:

- dist

expire\_in: 7 days

only:

- tags

tags:

- macOS

- matlab

linux build:

stage: build

script:

- MATLAB\_SCRIPT=$(pwd)/build\_executables.m

- MATLAB\_EXE=/usr/local/R2017a/bin/matlab

- LD\_PRELOAD\_DIR=/usr/lib/x86\_64-linux-gnu/libstdc++.so.6

- env LD\_PRELOAD=$LD\_PRELOAD\_DIR CORE\_CXX=/usr/bin/g++-4.9 $CI\_SCRIPTS\_DIR/build-linux…

function PTKAddPaths(varargin)

reset = nargin > 0 && strcmp(varargin{1}, 'reset');

force = nargin > 0 && strcmp(varargin{1}, 'force');

if reset

path(pathdef);

force = true;

end

% This version number should be incremented whenever new paths are added to

% the list

PTKAddPaths\_Version\_Number = 6;

persistent PTK\_PathsHaveBeenSet

full\_path = mfilename('fullpath');

[path\_root, ~, ~] = fileparts(full\_path);

cached\_pathname = [path\_root '.' PTKAddPaths\_Version\_Number];

% We force a class reset if the paths have changed since the last run

clear\_classes = ~isempty(PTK\_PathsHaveBeenSet) && ~strcmp(PTK\_PathsHaveBeenSet, cached\_pathname);

if force || (isempty(PTK\_P…

classdef PTKConfig

CacheFolder = ''

end

end

function updated = PTKUpdate(varargin)

% PTKUpdate. A script to update the PTK codebase via git

%

%

%

% Licence

%

persistent PTK\_LastUpdated

force\_update = nargin > 0 && strcmp(varargin{1}, 'force');

current\_date = date();

updated = false;

% We check for updates at most once per day, to avoid unnecessary

% startup delays

if force\_update || isempty(PTK\_LastUpdated) || ~strcmp(PTK\_LastUpdated, current\_date)

PTK\_LastUpdated = current\_date;

…

classdef (Sealed) PTKUtils < handle

% PTKUtils. A script to update the PTK codebase via git

%

%

%

% Licence

% -------

%

%

methods (Static)

function RunScript(script\_name, varargin)

% Runs a PTKScript from the Scripts folder

m = PTKMain();

m.RunScript(script\_name, varargin{:});

end

function Recompile()

% Forces recompilation of mex files

m = PTKMain();

m.Recompile();

…

classdef PTKViewer < GemFigure

% PTKViewer. A standalone image viewer for showing 3D images slice-by-slice

%

% PTKViewer uses MimViewerPanel to create a visualisation window for the

% supplied image, which can be a PTKImage or raw data. If a raw data

% matrix is supplied, the type argument can be supplied to ensure the

% image is displayed as expected.

%

% Syntax:

% obj = PTKViewer;

% obj = PTKViewer(image);

% obj = PTKViewer(image, image\_type);

% obj = PTKViewer(image, image\_type, reporting);

% obj = PTKViewer(image, overlay\_image);

% obj = PTKViewer(image, overlay\_image, reporting);

%

% Examples

% --------

%

…