The tikzquests package
A parametric questions' repository
Version 0.1

Alceu Frigeri*

January 2024

Abstract

This is a framework for building parametric questions' repositories, which can be further used to construct parametric questions for exams. Unlike other packages (for instance *exam*, *exam-n* or *exam-lite*) this doesn't try to enforce any pre-defined presentation format, focusing only on how to set a repository and use it.

Contents

1 Introduction

One recurring problem when typesetting exams, specially in Engineering fields, is the need to have parameterized questions, specially parameterized schematics used in exam's questions. The intent of this is to allow the creation of repositories (one or more) of 'easily' parameterized text and/or graphic (*tikz*) questions, but not enforcing any style/format whatsoever, leaving it to the end user.

The package offers

- A set of commands to create and use repositories,
- A set of commands to create and use 'questions' (code snippets) in a repository,
- Question's customization, parametrization, based on *pgfkeys* and a set of associated macros.

*https://github.com/alceu-frigeri/tikzquests

2 Package Options

`xtrakeys` This allows to expand the set of pre-defined (*pgfkeys*) keys. See 5.

`xtraidx` This allows to expand the set of pre-defined (*pgfkeys*) indexes per key. See 5.

`no defs` (default: false). This changes the parametrization strategy suppressing the creation of "keys commands".

`no alias` (default: false). This suppress the creation of Question's aliases (see 4.1), handy when one wants to reduce the 'noise' when listing all currently known Questions.

`in review` (default: false). When using a question (see 4.2), the question's associated remarks and annotations will be printed as well, if any.

`undef color` (default: red). This sets the color used to indicated the use of a non defined parameter. See 5.

3 Repositories

Questions are stored in a set of repositories. Per default there is one such repository, named `default`. A repository can have an "unlimited" set of questions.

`\defNewRepository` `\defNewRepository*` {⟨new-repository⟩}
`\selectRepository` `\SelectRepository` {⟨repository⟩}

`\defNewRepository` creates a new Repository, ⟨new-repository⟩. The starred version also switches to it, making it the "current default". `\SelectRepository` activates ⟨repository⟩ as the "current default" one.

> ***Note:*** About repository's names: It can be almost anything, the name can contain strings normally not allowed in a macro name, like spaces, dots, two-dots and so on. an important exception the backslash is still an active character, meaning that if someone typesets `\XYZ` as a repository, the value stored in `\XYZ` will be used (if `\XYZ` isn't defined this might raise a rather cryptic LaTeX 2_ε error).

> ***Note:*** When creating a new repository, a warning is raised in case ⟨new-repository⟩ already exists. When Selecting a repository, an error is raised if ⟨repository⟩ doesn't exists.

4 Questions

There are two kind of questions ("code snippets"): Text ones (which can include LaTeX 2_ε commands) and Graphical ones (which are assumed to be used inside a *tikz* environment). All questions can be parameterized (see 5 below) with a set of predefined commands, viz-à-viz, *pgfkeys*.

4.1 Defining a Question

`\defQuestion` `\defQuestion*` [⟨repository⟩] {⟨quest-name⟩} {⟨code⟩} [⟨remarks⟩]

`\defQuestion` creates a new Question in ⟨repository⟩ (per default using the "currently active" one). ⟨quest-name⟩ will be the associated question key, for later reference. ⟨code⟩ will be the corresponding Question's code. ⟨remarks⟩ is just a small note associated with the Question. It won't be normally used/typeset, except if the `in review` option is being used, or when using the command `\QuestionsList`.
The starred version `\defQuestion*` will create a "text code" one, whilst the non starred one `\defQuestion` will create a non starred question. The real main difference between the two is regarding their later use (see bellow). Please note that there are *two sets* of questions per repository. The "starred" (text ones) and the "non-starred" (graphical ones). They are fully distinct.

> ***Note:*** About question's names: It can be almost anything, the name can contain strings normally not allowed in a macro name, like spaces, dots, two-dots and so on, including backslashes, meaning that if someone typesets `\XYZ` as a question name, `\XYZ` will be it's name: a backslash isn't an active character anymore and one can't use macros when defining a question's name.

Note: An error is raised if ⟨repository⟩ doesn't exist. In case the ⟨quest-name⟩ already exists, it's code is silently replaced by the new one.

\defQuestionAlias `\defQuestionAlias*` [⟨repository⟩] {⟨quest-alias⟩} {⟨quest-name⟩}

`\defQuestionAlias` creates an alias, ⟨quest-alias⟩ for a given question, ⟨quest-name⟩. As for `\defQuestion`, the "starred one" regards text code ones and the "non starred one" regards the graphical ones.

Note: An error is raised if ⟨repository⟩ or ⟨quest-name⟩ doesn't exist.

4.2 Using a Question

\ftikzQuestion `\ftikzQuestion` (⟨scale⟩) [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩>
\tikzQuestion `\tikzQuestion` (⟨scale⟩) [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩>
\rawtikzQuestion `\rawtikzQuestion` [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩>

Those are the main commands to use a "non starred" (graphical) question. The `\tikzQuestion` will display the Question's code inside a *tikzpicture* environment. `\ftikzQuestion` will further nest the code inside a *center* environment (preparing it to be used inside a floating environment), whilst `\rawtikzQuestion` will just display the code in a local group (in case one wants to use a different graphical environment, like SVG. The ⟨annotation⟩ will only be added if, and only if, the `in review` option is being used. The ⟨scale⟩ factor, when present, is related to the current `\textwidth`, so a factor of 0.25 will scale the width of the question to 1/4 of the text width. The ⟨key=value list⟩ is a set of *pgfkeys*, see 5 below.

Note: An error is raised if ⟨repository⟩ or ⟨quest-name⟩ doesn't exist.

\textQuestion `\textQuestion` [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩>

This is the command to use a "starred" (text) question. The `\textQuestion` will just use the code in a local group. The ⟨annotation⟩ will only be added if, and only if, the `in review` option is being used. The ⟨key=value list⟩ is a set of *pgfkeys*, see 5 below.

Note: An error is raised if ⟨repository⟩ or ⟨quest-name⟩ doesn't exist.

\QuestionsList `\QuestionsList` [⟨repositories list⟩]

⟨repositories list⟩ is a comma separated list of repositories. For each repository, all Questions will be typeset in a *describe* environment. If no ⟨repositories list⟩ is supplied, all repositories will be listed.

Note: An error is raised if a repository in ⟨repositories list⟩ doesn't exist. To reduce the clutter (and number of questions listed) one should consider the use of the option `no alias`.

5 Parameters as pgfkeys and auxiliary macros.

By default a set of *pgfkeys* and macros is defined as follow: ⟨ID⟩⟨idx⟩. The predefined range of ⟨IDs⟩ being *R*, *L*, *C*, *X*, *Y*, *Z*, *K*, *T*, *Q*, *EQ*, *V* and *I*. The predefined range of ⟨idx⟩ spans from a, b, c up to z, aa, ab, ac up to az and, finally, ba, bb, bc up to bz.

Note: That means, one gets to use *pgfkeys* as, for example, Ra=200, Lca=500, Kbe=230 and so on. For each of the key's ⟨ID⟩ there is (per default) a set of 3x26 keys. If one adds some 3 other ⟨idx⟩ then one gets 6x26 keys per ⟨ID⟩.

Note: The ⟨ID⟩ can be extended with the `xtrakeys` option. For example with `xtrakeys={NN , B}`, each extra key will add 3x26 keys (per default).

Note: The ⟨idx⟩ can be extended with the `xtraidx` option. For instance with `xtraidx={f , g}` one gets fa, fb, fc up to fz, ga, gb, gc up to gz as well, a x26 set per extra idx.

Unless the `no defs` option is defined, a corresponding macro (with the same name) will also be defined. For example, there is a macro `\Ra` associated with the key *Ra*. All those macros/keys are initialized as follow: "math mode" (`\ensuremath`), ID_{idx}, so, for example, the key *Rab*, which can be accessed with the macro `\Rab`, will be predefined as R_{ab}, *Lca* (`\Lca`) will be predefined as L_{ca}, and so on.

The idea is that, when using the commands in 4.2, if one doesn't specify a ⟨key=value list⟩, the default values will be "en force", and all one has to do (to change those values) is to set said list, which doesn't have to be complete, non assigned keys will keep their default value.

Besides those default keys, one can set and use any key at will (in ⟨key=value list⟩) non-existing keys will be created "on the fly" with the given name as the default value.

5.1 Assigning a new value to pre-defined keys

In fact, with each and every key there are 3 ways to assign a value to it ⟨key⟩=*value*, ⟨key*⟩=*value* and ⟨key raw⟩=*value*. The difference being that ⟨key raw⟩ will assign whatever code/value to the key (and associated macro), ⟨key*⟩ will assign the code/value inside a math environment.

warning: ⟨key⟩=*value* (without any specifier) will be the same as ⟨key*⟩ (default). But, if the option `no defs` is used, it will be equivalent to ⟨key raw⟩.

5.2 Using a parameter key

When defining the ⟨code⟩ of a question (see 4.1) one has two options to recover a key's value:

- A macro named after the key itself (in case of the default keys, see above) and, or
- the `\QuestVal` command which allows to recover the value of both the default keys, as well as the ones defined on the fly.

Warning: If the option `no defs` is defined, the only option to recover a key's value is `\QuestVal`. One better choose which style fits better, and keep it.

`\QuestVal` `\QuestVal` {⟨IDidx⟩}
`\QuestVal` {⟨key⟩}

This will always recover the value of a key, regardless if the key is one of the pre-defined ones (in the form ⟨IDidx⟩ or a "on the fly" one. If the key didn't got (re)defined with the ⟨key=value list⟩, this will return the key/paramenter default value. In the case of a "on the fly" key, it will be the key's name in red (or the color set up with the `undef color` option, see 2).
This should be safe in most situations where `\pgfkeys` command can be used. Though, the safest, and most robust, way to use a parameter/key is using it's related macro.

6 Example of Use

6.1 Package Options

Package Options

```
\usepackage{tikzquests}
```

This is the default case, in which both `\QuestVal` and (for default keys) associated macro name can be used to retrieve a key/parameter value.

Package Options

```
\usepackage[xtrakeys={EX,N},xtraidx={f,h},undef color={blue},no defs]{tikzquests}
```

In this case, one will get the following set of keys

- Ra, Rb ... Rz, Rab, Rab ... Raz .... *Rfa, Rfb ... Rfz, Rha, Rhb ... Rhz*
- .... all other default sets of keys, plus
- EXa, EXb ... EXz, EXab, EXab ... EXaz .... *EXfa, EXfb ... EXfz, EXha, EXhb ... EXhz*
- Na, Nb ... Nz, Nab, Nab ... Naz .... *Nfa, Nfb ... Nfz, Nha, Nhb ... Nhz*

Besides that, the undefined color will be blue and no additional macro will be defined, in which case on has to use `\QuestVal` to recover the value of a key/parameter.

Package Options

```
\usepackage[no alias, in review]{tikzquests}
```

In this case, no alias will be defined (the command `\defQuestionAlias` will be ignored), and when using `\tikzQuestion` (and similar) the question's remarks (defined by `\defQuestion`) and annotations (from `\tikzQuestion`) will be printed. The `no alias` is specially useful when using the command `\QuestionsList`.

6.2 A More Complete Example

In the code below, an extra repository will be set (besides the default one) and two questions (a starred, text, and non starred, graphics) will be defined for each repository.

Questions Definitions

```
% A repository name can be just about anything.
% the star makes sure 'Repo 2' is now the active/default one.
\defNewRepository*{Repo 2}

% quest names are even more flexible than a repository one
% the star implies this is a text one.
\defQuestion*{Quest A:1}{
 In the following circuit, assuming $\beta \approx \QuestVal{Beta}$ and that $V_{be} \approx 0.65V$, find the
      value of $R_c$ such that the small signal gain is \QuestVal{Gain}.
}[That would be a question enunciate.]

%% Note the use of the macros \Ra, \Rb, \Rc, \Rd, \Vi, \Vbc and \Vo
\defQuestion[Repo 2]{Elect. 1a}{
    \draw
    (0,0) coordinate(A) to[V,invert,l=\Vi] ++(0,3) coordinate(V)
      to[R=\Ra] ++(2,0)
      to[C] ++(2,0) coordinate(B)
      -- ++(1,0) node[npn,anchor=B] (T1){}
    (A) -- (A -| B) coordinate(Ba) to[R=\Rb] (B) to[R=\Rg] ++(0,3) coordinate(C)
    (T1.E) to[R,l=\Rc] (T1.E |- A) -- (A)
    (T1.C) to[R,l_=\Rd] (T1.C |- C) -- (C -| A) -- ++(-2,0) coordinate(X) to[V,l=\Vbc] (X |- A) -- (A)
    (T1.C) -- ++(1.5,0) node[ocirc]{} coordinate(k) to[open,v=\Vo] (k |- A) node[ocirc]{} -- (A)
     ;
}[this is a CircuiTikZ example]

%switching repositories
\SelectRepository{default}

% Note that, since it is a different repository, there is no name crashing.
\defQuestion*{Quest A:1}{
 In the following circuit, assuming $\beta \approx \QuestVal{Beta}$ and that $V_{be} \approx 0.65V$, find the
      value of \Rg \ such that the DC level of \Vo \ is equal to \QuestVal{DC level}.
}[Just for the sake of it.]

%% Note the use of the macros \Ra, \Rb, \Rc, \Rd, \Vi, \Vbc and \Vo
\defQuestion{Elect. 1b}{
    \draw
    (0,0) coordinate(A) to[V,invert,l=\Vi] ++(0,3) coordinate(V)
      to[R=\Ra] ++(2,0)
      to[C] ++(2,0) coordinate(B)
      -- ++(1,0) node[pnp,anchor=B] (T1){}
    (A) -- (A -| B) coordinate(Ba) to[R=\Rb] (B) to[R=\Rg] ++(0,3) coordinate(C)
    (T1.C) to[R,l=\Rc] (T1.C |- A) -- (A)
    (T1.E) to[R,l_=\Rd] (T1.E |- C) -- (C -| A) -- ++(-2,0) coordinate(X) to[V,l=\Vbc] (X |- A) -- (A)
    (T1.C) -- ++(1.5,0) node[ocirc]{} coordinate(k) to[open,v=\Vo] (k |- A) node[ocirc]{} -- (A)
     ;
}[this is a CircuiTikZ example]
```

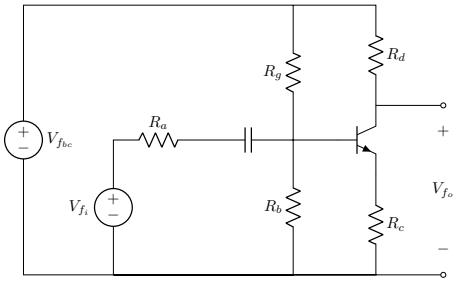
Once Questions are defined one can use them, for instance, using just the default parameter's values.

Questions Defaults

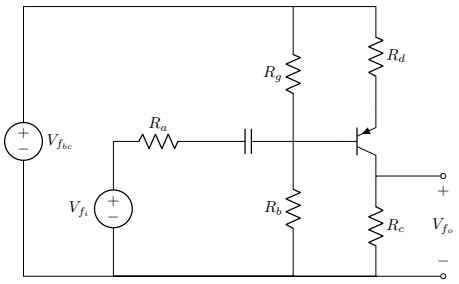
```
Choose one of the following two questions:

\begin{enumerate}
  \item \textQuestion[Repo 2]{Quest A:1}<just a last minute note about this>\par
  \ftikzQuestion(0.4)[Repo 2]{Elect. 1a}

  \item \textQuestion[default]{Quest A:1}<just for the sake of it...>\par
  \ftikzQuestion(0.4)[default]{Elect. 1b}
\end{enumerate}
```

Choose one of the following two questions:

1. In the following circuit, assuming $\beta \approx$ ***Beta*** and that $V_{be} \approx 0.65V$, find the value of R_c such that the small signal gain is ***Gain***.



2. In the following circuit, assuming $\beta \approx$ ***Beta*** and that $V_{be} \approx 0.65V$, find the value of R_g such that the DC level of V_{f_o} is equal to ***DC level***.



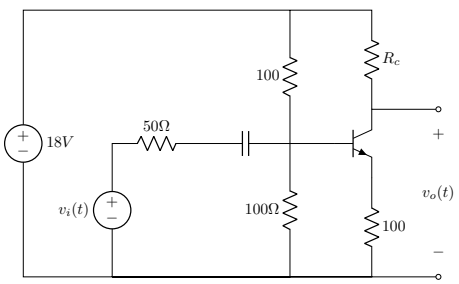
Finally, one can use those same questions, setting it's parameters:

Questions Using Parameters

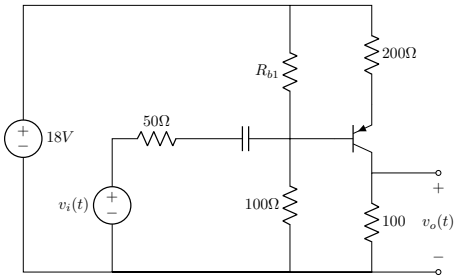
```
Choose one of the following two questions:

\begin{enumerate}
  \item \textQuestion[Repo 2]{Quest A:1}[Beta=200,Gain=4,Ra=50\Omega]<just a last minute note about this>\par
  \ftikzQuestion(0.4)[Repo 2]{Elect. 1a}[Ra=50\Omega,Rb*=100\Omega,Rg raw=100,Rc=100,Rd=R_c,Vbc=18V,Vi=v_i(t),Vo=v
      _o(t)]

  \item \textQuestion[default]{Quest A:1}[Beta=200,DC level=8V,Rg=R_{b1},Vo=v_o(t)]<just for the sake of it...>
      \par
  \ftikzQuestion(0.4)[default]{Elect. 1b}[Ra=50\Omega,Rb*=100\Omega,Rg=R_{b1},Rc=100,Rd=200\Omega,Vbc=18V,Vi=v_i(t
      ),Vo=v_o(t)]
\end{enumerate}
```

Choose one of the following two questions:

1. In the following circuit, assuming $\beta \approx 200$ and that $V_{be} \approx 0.65V$, find the value of R_c such that the small signal gain is 4.



2. In the following circuit, assuming $\beta \approx 200$ and that $V_{be} \approx 0.65V$, find the value of R_{b1} such that the DC level of $v_o(t)$ is equal to 8V.


200Ω
R_{b1}
50Ω
18V
$v_i(t)$
100Ω
100
$v_o(t)$
+
−