

SLyDIF | ドキュメント

monaqa

GitHub: <https://github.com/monaqa>

2021 年 11 月 14 日

はじめに

はじめに

SLyDIF₁とは

SLyDIF₁はSATySF₁でプレゼンテーション用のスライドを作成するためのクラスファイル。以下のような機能を備えている。

- ◆ タイトルフレーム、セクションフレームなどを含めたスライドの作成
- ◆ スライド内でのインラインマークアップ
- ◆ 任意の位置への図版の挿入、整列
- ◆ オーバーレイ
- ◆ カスタマイズ / 自作の可能なテーマ

このドキュメントもSLyDIF₁を用いて書かれており、使用テーマはArcticである。ただし紙面を十分に確保するため、フォントサイズを通常より縮めた設定を入れている。

SLyDIF₁の概要

SLyDIF₁の概要

パッケージ構成

SLyDIF₁ は以下のヘッダファイルで構成されている。

- ◆ `slydifi.satyh`: 共通処理やコマンドが書かれた `SlydifiScheme` モジュールを定義。
- ◆ `footnote.satyh`: 脚注に関するモジュール `FootnoteScheme` を定義。
- ◆ `overlay.satyh`: オーバーレイに関するモジュール `SlydifiOverlay` を定義。
- ◆ `param.satyh`: 汎用パラメータに関するモジュール `SlydifiParam` を定義。
- ◆ `graphics.satyh`: グラフィックスに関するモジュール `SlydifiGraphics` を定義。
- ◆ `theme` ディレクトリ: SLyDIF₁ のテーマを格納。内部で `slydifi.satyh` を読み込む。
 - ▶ `akasaka.satyh`
 - ▶ `arctic.satyh`
 - ▶ `hakodate.satyh`
 - ▶ `plain.satyh`

パッケージのインポート

基本的には、`theme` 内に入っているテーマファイルを読み込めばよい。

```
@require: class-slydifi/theme/plain
% "plain"の部分を変えてテーマ変更が可能

document '<
  (本文)
>
```

テーマはデフォルトで以下の4種類が用意されている。

<code>plain</code>	最も簡素なテーマ
<code>akasaka</code>	灰色のスタンダードなテーマ
<code>arctic</code>	<code>iceberg</code> の色をベースにしたテーマ (本ドキュメントで使用)
<code>hakodate</code>	<code>gruvbox</code> の色をベースにしたテーマ

SLYDIF₁の基本機能

SLYDIF₁の基本機能

フレーム：スライドの1ページ

フレームの作成には `+frame{ title }< inner >` を用いる。

```
+frame{フレーム：スライドの1ページ}<
  +p{
    フレームの作成には `+frame` を用いる。
  }
  ...
>
```

- ◆ `+frame` で囲った部分が出力される PDF の1ページになる。
- ◆ `+frame` の中にどれだけ多くの記述を入れても、自動でのページ分割は行われない。
 - ▶ 入り切らないブロックは描画領域からはみ出て見えなくなるだけ。
 - ▶ 「1枚のフレームでどの情報まで表示するか」という発表時の戦略に関わる部分は手動で制御するのが望ましいと考え、自動で分割する機能は入れていない。

章組み：タイトルフレーム

タイトルフレームを挿入したい箇所に `+make-title(| configs |)` を入れる。

```
+make-title(|
  title = {|\SLyDIFi; ドキュメント |};
  author = {
    | monaqa
    | GitHub: \link(`https://github.com/monaqa`);
  };
  date = {|2020年7月23日|};
|);
```

- ◆ 引数は `title`, `author`, `date` ラベルを持つレコード型であり、対応する値はいずれも `inline-text list` 型。
- ◆ `|` で区切ることで、複数行にわたるタイトルや著者名を表示可能

章組み：セクションフレーム

`+section{| title |}< frames >` を使うと、冒頭にセクションフレームが追加される。

```
+section{|\SLyDIFi; の基本機能 |}<

  +frame{フレーム：スライドの1ページ}< (内容) >
  +frame{章組み：タイトルフレーム}< (内容) >
  ...

>
```

- ◆ 多くのテーマでは、セクション冒頭にセクション用のフレームが追加されるだけ。
- ◆ arctic テーマではセクション用のフレームが追加されるだけでなく、中のフレームのフッターにセクション名が表示される。

フレーム内の文書構造

- ◆ `+p{ inner text }` で段落を作成できる。
- ◆ `+listing{ * items }` で番号のない箇条書きを作成できる (`enumitem` パッケージを使用)。
- ◆ `+enumerate{ * items }` で番号付き箇条書きを作成できる (同上)。

```
+frame{パッケージ構成}<
  +p{
    \SLyDIFi; は以下のヘッダファイルで構成されている。
  }
  +listing{
    * `slydifi.satyh`: 共通処理やコマンドが書かれた
      `SlydifiScheme` モジュールを定義。
    * ...
  }
>
```

インラインマークアップ

SLyDIF₁ ではいくつかのインラインマークアップを提供する。

コマンド	役割
<code>\br;</code>	強制改行
<code>\emph{ text }</code>	強調（強調の方式はテーマにより異なる）
<code>\text-color(color){ text }</code>	テキストの色の変更
<code>\uline{ text }</code>	テキストに下線を引く
<code>\strike{ text }</code>	打ち消し線
<code>\ctx[ctx-func]{ text }</code>	テキスト処理文脈の部分的な変更。
<code>\SLyDIF₁;</code>	「SLyDIF ₁ 」 と書きたいときに

スライドのテーマによっては、`\textbf` など上で述べた以外のコマンドを追加していることもある。また、`annot` や `code` ライブラリで追加されるコマンドを用いることも可能。

オーバーレイ

オーバーレイ

オーバーレイとは

1枚のフレームの中で、プレゼンターの操作によって初めはなかった文字が現れたり、文字が順々に表示されたりすることがある。このようなフレームの動きをオーバーレイと呼ぶ。元はL^AT_EXのBeamerで実装されている機能の名称だが、SLyDIF₁でも同じ名称を用いる。

「オーバーレイ」を最も単純に実現したい場合、右のように一部が共通しているフレームを複数作成する方法が考えられる。しかしこの方法には以下の問題がある：

- ◆ 同じ内容を2回以上書かなければならない
- ◆ 共通する部分を変更する際に複数箇所を編集する必要がある

SLyDIF₁には、より簡単な記法でオーバーレイを実現するための手法が備わっている。

```
+frame{タイトル}<
  +p{1枚目と2枚目で表示される段落。}
>
+frame{タイトル}<
  +p{1枚目と2枚目で表示される段落。}
  +p{2枚目のみで表示される処理。}
>
```

オーバーレイ (1/3)

SLyDIF_lでは、右のように書くことで同一の内容を複数回書くことなくオーバーレイを実現できる。具体的には、`+p?:(only 2){ ... }`と書かれている段落が2枚目のレイヤーでのみ表示されるようになる。

`+frame` は整数値のオプション引数をとることができ、明示すると指定した数だけのレイヤーが作られる。第2引数のインラインテキストにはフレームタイトルを、ブロックテキストには `+frame` と同じように本文を指定する。複数枚のレイヤーを作成することで、「同じ内容のスライドを、一部だけ変えて繰り返す」ことが可能となる。

```
+frame?:(2){タイトル}<
  +p{1枚目と2枚目で表示される段落。}
  +p?:(only 2){2枚目のみで表示される処理。}
>
```

オーバーレイ (2/3)

SLyDIF_l では、右のように書くことで同一の内容を複数回書くことなくオーバーレイを実現できる。具体的には、`+p?:(only 2){ ... }` と書かれている段落が2枚目のレイヤーでのみ表示されるようになる。

`+frame` は整数値のオプション引数をとることができ、明示すると指定した数だけのレイヤーが作られる。第2引数のインラインテキストにはフレームタイトルを、ブロックテキストには `+frame` と同じように本文を指定する。複数枚のレイヤーを作成することで、「同じ内容のスライドを、一部だけ変えて繰り返す」ことが可能となる。

2つ目の `+p` ではオプション引数が指定されており、オプション引数に表示条件を入れることで表示を制御することが出来る。

```
+frame?:(2){タイトル}<
  +p{1枚目と2枚目で表示される段落。}
  +p?:(only 2){2枚目のみで表示される処理。}
>
```


オーバーレイ (3/3)

SLyDIF_l では、右のように書くことで同一の内容を複数回書くことなくオーバーレイを実現できる。具体的には、`+p?:(only 2){ ... }` と書かれている段落が2枚目のレイヤーでのみ表示されるようになる。

```
+frame?:(2){タイトル}<
  +p{1枚目と2枚目で表示される段落。}
  +p?:(only 2){2枚目のみで表示される処理。}
>
```

`+frame` は整数値のオプション引数をとることができ、明示すると指定した数だけのレイヤーが作られる。第2引数のインラインテキストにはフレームタイトルを、ブロックテキストには `+frame` と同じように本文を指定する。複数枚のレイヤーを作成することで、「同じ内容のスライドを、一部だけ変えて繰り返す」ことが可能となる。

2つ目の `+p` ではオプション引数が指定されており、オプション引数に表示条件を入れることで表示を制御することが出来る。

オプション引数の `only 2` とは、2枚目のレイヤーのみで表示される、ということを表している。

挙動を変えるためのコマンド一覧 1

今までに紹介したコマンドのうち、`\emph` などいくつかのものはオプション引数によってオーバーレイに対応することができる。たとえば `\emph` では、オプション引数に入れた表示条件を満たした場合のみその文字が強調される。他のコマンドも同様に「表示条件を満たした場合のみ効果が発動する」ようになっている。

- ◆ `\emph?:(cond){ text }`
- ◆ `\text-color?:(cond)(color){ text }`
- ◆ `\uline?:(cond){ text }`
- ◆ `\strike?:(cond){ text }`
- ◆ `\ctx?:(cond)[ctx-funcs]{ text }`
- ◆ `+ctx?:(cond)[ctx-funcs]{ text }`

挙動を変えるためのコマンド一覧 2

特定のレイヤーで書式を変えるだけでなく、特定のレイヤーでのみ表示したい、非表示にしたいということもよくある。以下のコマンドでそういった表示そのものを制御できる。

コマンド	役割
<code>+p?:(cond){ text }</code>	条件を満たすときのみ表示される段落
<code>+ghost(cond)< block ></code>	条件を満たすときのみ表示されるブロック
<code>\ghost(cond){ text }</code>	条件を満たすときのみ表示されるテキスト
<code>+phantom(cond)< block ></code>	条件を満たすときは普通に表示され、満たさないときは同等の空白が表示されるブロック
<code>\phantom(cond){ text }</code>	<code>+phantom</code> のインラインテキスト版
<code>+select(cond)< block >< block ></code>	条件を満たすときは2番目、満たさないときは3番目の引数に置き換わるブロック
<code>\select(cond){ text }{ text }</code>	<code>+select</code> のインラインテキスト版

レイヤーの表示条件を定める関数一覧

`+p` のオプション引数などに与える `cond` は `int -> bool` 型の関数であれば何でも良いが、簡単のためエイリアス的な関数をいくつか用意している。「 m 枚目から n 枚目のレイヤーまで」のように比較的単純な条件であれば以下の関数を用いると楽。

いずれも `SlydifiOverlay` モジュール中にあるため、最初に `open SlydifiOverlay` を行うと良い。

関数	挙動
<code>only N</code>	N 番目のレイヤーでのみ成立
<code>until N</code>	N 番目のレイヤーまで成立
<code>from N</code>	N 番目のレイヤー以降のみ成立
<code>between M N</code>	M 番目から N 番目のレイヤーでのみ成立
<code>on-slides [N1; N2; ...]</code>	リストに数字が入っているレイヤーでのみ成立
<code>non (cond)</code>	<code>cond</code> の否定

図版の挿入

図版の挿入

FigBox モジュール

スライドというコンテンツでは、通常の文書以上に図版が重要な役割を持つ。図版の大きさ、配置、余白の量がスライドの見た目に大きな影響を与え、結果的に通常の文書よりも自由度の高い設定が求められることになる。

SLyDIF₁ では、カスタマイズしやすい図版の挿入を実現するために `figbox` パッケージを依存として用いる。Satyrographos を用いてインストールされた場合は自動で入っている。

以下では、プリアンブルにて `FigBox` モジュールを `open` しているものとする。

```
open FigBox
```

`figbox` パッケージの詳細な使い方は [パッケージのドキュメント](#) を参照すること。

図の配置コマンド

`FigBox.t` 型の値 `figbox` を指定して、以下のコマンドを用いることで図が挿入される。

コマンド	役割
<code>+fig-center(figbox);</code>	中央揃えで配置
<code>+fig-block?:(align)(figbox);</code>	<code>align</code> で指定した揃え方で表示 (左右揃え)
<code>+fig-on-right(figbox) < block ></code>	本文 (<code>block</code>) の横幅を <code>figbox</code> の横幅の分だけ短くし、本文の右側に図を表示。
<code>+fig-on-left(figbox) < block ></code>	本文 (<code>block</code>) の横幅を <code>figbox</code> の横幅の分だけ短くし、本文の左側に図を表示。
<code>+fig-abs-pos((x, y))(figbox);</code>	絶対座標で配置
<code>+fig-inline(figbox);</code>	インラインテキスト中に図を挿入

「図」 として配置可能なもの

FigBox.t 型として配置することのできる基本的なコンテンツは以下のとおり。

コンテンツ	関数
JPEG 画像	<code>include-image width `path/to/file.jpg`</code>
PDF (1 ページ目)	<code>include-image width `path/to/file.pdf`</code>
PDF (n ページ目)	<code>include-image ? : n-page width `path/to/file.pdf`</code>
JPEG 画像 (縦幅指定)	<code>include-image-with-height height `path/to/file.jpg`</code>
ダミーのブロック	<code>dummy-box width height</code>
インラインテキスト	<code>textbox ? : (ctx-func) { text }</code>
インラインテキスト (幅指定)	<code>textbox-with-width ? : (ctx-func) width { text }</code>
ブロックテキスト	<code>textblock ? : (ctx-func) width '< text ></code>

図の結合

複数の図を並べたくなることは多い。SLyDIF₁ では複数の図を簡単に並べることができる。

内容	関数
図を横に並べる	<code>hconcat ?:(align) [figbox-1; ...; figbox-n]</code>
図を縦に並べる	<code>vconcat ?:(align) [figbox-1; ...; figbox-n]</code>

- ◆ オプション引数 `align` に `float` 型の値を指定することで、揃え方を変更できる。
 - ▶ `0.0`: 左揃え / 下揃え
 - ▶ `0.5`: 中央揃え
 - ▶ `1.0`: 右揃え / 上揃え
- ◆ `gap` 関数を途中に挟むと、指定したぶんの余白が図と図の間に挿入される。
 - ▶ `hconcat` でも `vconcat` でも使える
 - ▶ 例: `hconcat [figbox-1; gap 10pt; figbox-2; gap 5pt; figbox-3]`

図の変換

「縦に並べる」「指定した大きさのマージンを周囲に足す」といった操作を図に加えたいこともある。SLyDIF₁では `figbox` を入力として新たな `figbox` を返す関数をいくつか用意しており、既存の図から簡単にマージンなどの要素を加えた新たな図を作成できる。

変換内容	関数
左右に余白を追加	<code>figbox > hmargin length</code>
上下に余白を追加	<code>figbox > vmargin length</code>
上下左右に余白を追加	<code>figbox > hvmargin length</code>
上下左右に異なる大きさの余白を追加	<code>figbox > margin left right top bottom</code>
フレーム（外枠）を描画	<code>figbox > frame line-width line-color</code>
図の背景色を設定	<code>figbox > bgcolor color</code>

◆ 複数の変換を組み合わせることも当然可能。

▶ 例：`include-image 100pt `file.pdf` |> hvmargin 10pt |> frame 1pt Color.black`

◆ `hconcat` や `vconcat` と組み合わせることも当然可能。

図のオーバーレイ表示 (1/4)

図もオーバーレイに対応しており、特定のレイヤーに応じて図を出し分けることが可能。表示条件を満たすとき図を表示し、それ以外では大きさを0にしたいときは `fig-ghost` 関数を用いて `figbox |> fig-ghost (cond)` とする。

表示条件を満たすとき図を表示し、それ以外では図を不可視としたい（元の図と同じ大きさの透明の箱を置きたい）ときは `fig-phantom` 関数を用いて `figbox |> fig-phantom (cond)` とする。

`select` 関数を用いて図の内容や体裁などを切り替えることも可能。

This box appears on the first and fourth layers.

The line color of this box is blue on the first and second frame, and red after that. Therefore, the line color of this box is **blue** now.

図のオーバーレイ表示 (2/4)

図もオーバーレイに対応しており、特定のレイヤーに応じて図を出し分けることが可能。表示条件を満たすとき図を表示し、それ以外では大きさを0にしたいときは `fig-ghost` 関数を用いて `figbox |> fig-ghost (cond)` とする。

表示条件を満たすとき図を表示し、それ以外では図を不可視としたい（元の図と同じ大きさの透明の箱を置きたい）ときは `fig-phantom` 関数を用いて `figbox |> fig-phantom (cond)` とする。

`select` 関数を用いて図の内容や体裁などを切り替えることも可能。

The line color of this box is blue on the first and second frame, and red after that. Therefore, the line color of this box is **blue** now.

図のオーバーレイ表示 (3/4)

図もオーバーレイに対応しており、特定のレイヤーに応じて図を出し分けることが可能。表示条件を満たすとき図を表示し、それ以外では大きさを0にしたいときは `fig-ghost` 関数を用いて `figbox |> fig-ghost (cond)` とする。

表示条件を満たすとき図を表示し、それ以外では図を不可視としたい（元の図と同じ大きさの透明の箱を置きたい）ときは `fig-phantom` 関数を用いて `figbox |> fig-phantom (cond)` とする。

`select` 関数を用いて図の内容や体裁などを切り替えることも可能。

The line color of this box is blue on the first and second frame, and red after that. Therefore, the line color of this box is **red** now.

図のオーバーレイ表示 (4/4)

図もオーバーレイに対応しており、特定のレイヤーに応じて図を出し分けることが可能。表示条件を満たすとき図を表示し、それ以外では大きさを0にしたいときは `fig-ghost` 関数を用いて `figbox |> fig-ghost (cond)` とする。

表示条件を満たすとき図を表示し、それ以外では図を不可視としたい（元の図と同じ大きさの透明の箱を置きたい）ときは `fig-phantom` 関数を用いて `figbox |> fig-phantom (cond)` とする。

`select` 関数を用いて図の内容や体裁などを切り替えることも可能。

This box appears on the first and fourth layers.

The line color of this box is blue on the first and second frame, and red after that. Therefore, the line color of this box is **red** now.