Creating a Declarative UI Library in C++ with wxWidgets

Part 1

Richard Powell, rmpowell77@me.com, v1.0

```
➜  ~ curl -sS "https://en.wikipedia.org/wiki/Declarative_programming?
action=raw" | grep -o -i declarative | wc -l
      38
```

Why the command line is overrated…

https://medium.com/nerd-for-tech/why-the-command-line-interface-is-overrated-497fd4ecd7a6

Why the command line is overrated…

- “We humans are not good at memorizing sequences of characters without visual cues.”

https://medium.com/nerd-for-tech/why-the-command-line-interface-is-overrated-497fd4ecd7a6

Why the command line is overrated…

- “We humans are not good at memorizing sequences of characters without visual cues.”
- Generally require more cognitive effort to use than their graphical counterparts.

https://medium.com/nerd-for-tech/why-the-command-line-interface-is-overrated-497fd4ecd7a6

Why the command line is overrated…

- “We humans are not good at memorizing sequences of characters without visual cues.”
- Generally require more cognitive effort to use than their graphical counterparts.
- Require the user to know the commands exactly before using the tool.

https://medium.com/nerd-for-tech/why-the-command-line-interface-is-overrated-497fd4ecd7a6

Why the command line is overrated…

- “We humans are not good at memorizing sequences of characters without visual cues.”
- Generally require more cognitive effort to use than their graphical counterparts.
- Require the user to know the commands exactly before using the tool.
- Massive assumptions about the user’s language and accessibility.

https://medium.com/nerd-for-tech/why-the-command-line-interface-is-overrated-497fd4ecd7a6

Why the command line is overrated…

- “We humans are not good at memorizing sequences of characters without visual cues.”
- Generally require more cognitive effort to use than their graphical counterparts.
- Require the user to know the commands exactly before using the tool.
- Massive assumptions about the user’s language and accessibility.
- Extremely challenging to do creative endeavors.

https://medium.com/nerd-for-tech/why-the-command-line-interface-is-overrated-497fd4ecd7a6

My journey

My journey

Pregame 2011_mod_w_cont.shw

Colors | Zoom | Grid: 2 | Tool: 2 | Ref Group: Off | Path | Ghost: Off | Instruments | Marcher

Field Thumbnails

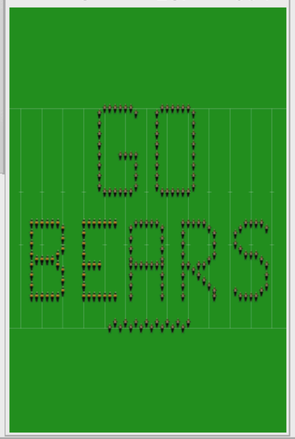
F-O-R SS6A

N-I-A SS6B

SF SS7

Animation

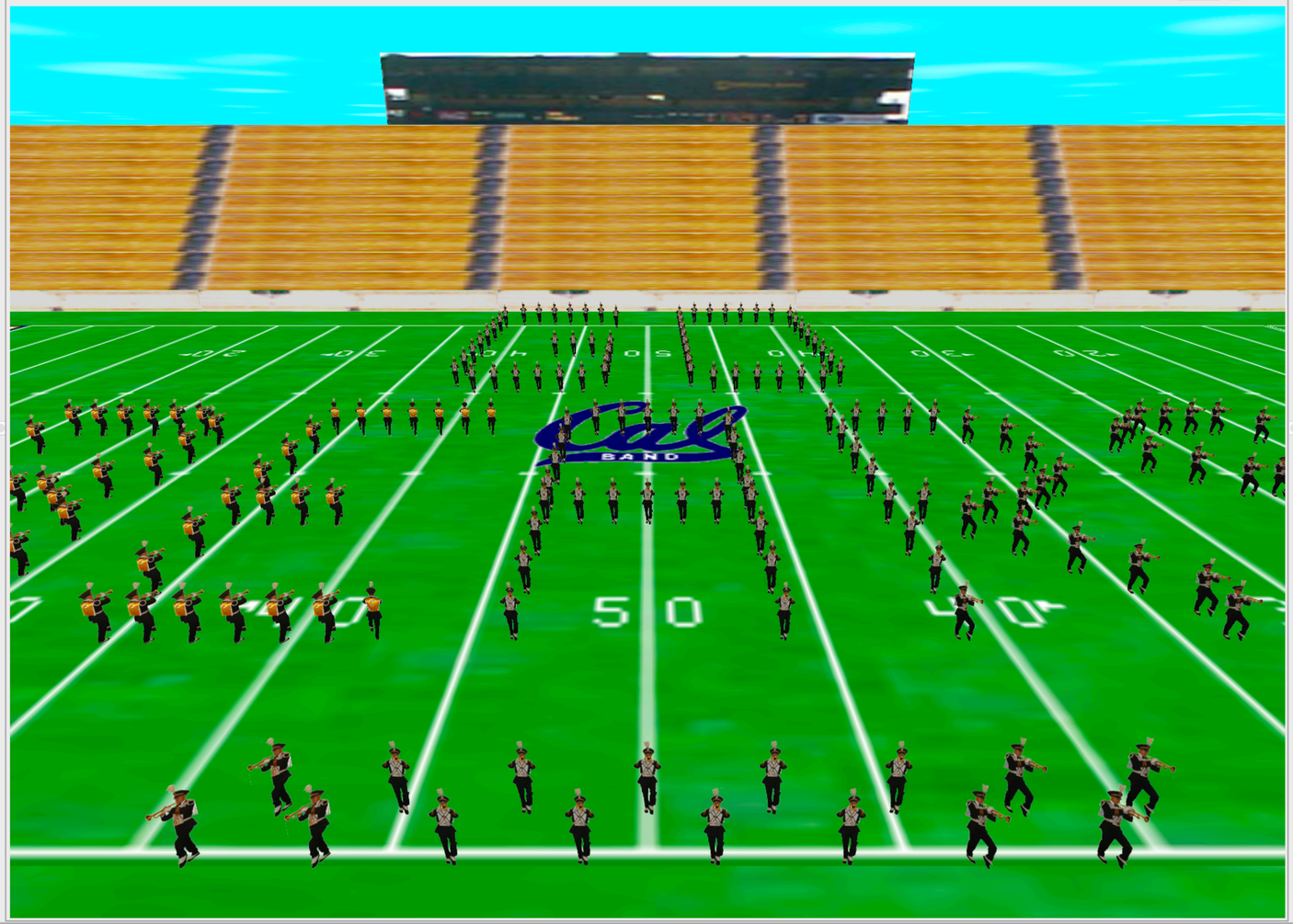
0 267 534



Animate

0 267 534

Tempo 120



Continuities

Plain

MarkTime for Remaining W

Solid

Marching Errors

Errors

Collisions

Print Continuity

Note: Maintain 8/5 until Point P

"C" line ●: **Line Leader:** Form the "C" line as per

~\bsNote: Maintain 8/5 until Point P\be

"C" line \so: \bsLine Leader:\be Form the "C" line
\po: Follow Line Leader (\so) as per arrow

Welcome to Calchart vv3.6.3 | * 13 of 29 "F-O-R SS6A" 4 beats | 0 of 199 selected

Pregame 2011_mod_w_cont.shw

Colors | Zoom | Grid: 2 | Tool: 2 | Ref Group: Off | Path | Ghost: Off | Instruments | Marcher

Field Thumbnails

F-O-R SS6A

N-I-A SS6B

SF SS7

Animation

0 267 53

Animate

Tempo

Continuities

Plain

MarkTime for Remaining W

Solid

Errors

Errors

Collisions

Print Continuity

Note: Maintain 8/5 until Point P

"C" line ●: Line Leader: Form the "C" line as per

Follow Line Leader (●) as per a

~bsNote: Maintain 8/5 until Point P\be

C" line \so: \bsLine Leader:\be Form the "C" line

\po: Follow Line Leader (\so) as per arrow

Welcome to Calchart vv3.6.3

Pregame 2011_mod_w_cont.shw

Colors | Zoom | Grid: 2 | Tool: 2 | Ref Group: Off | Path | Ghost: Off | Instruments | Marcher

Field Thumbnails

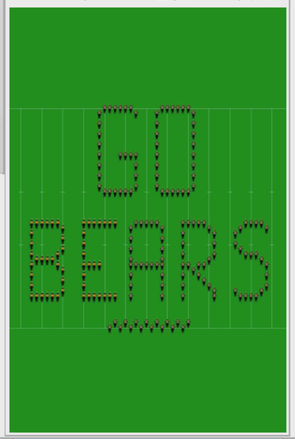
F-O-R SS6A

N-I-A SS6B

SF SS7

Animation

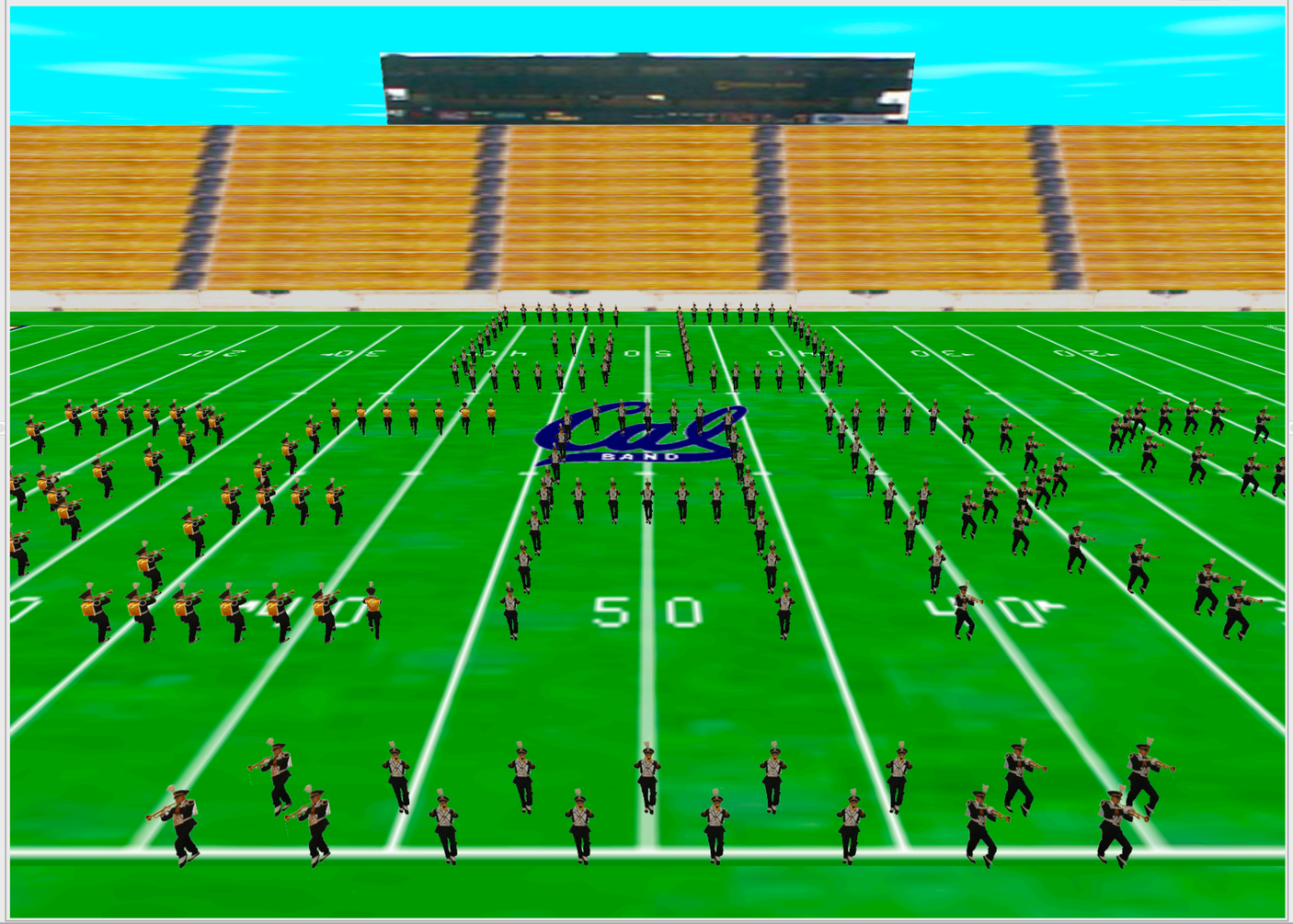
0 267 534



Animate

0 267 534

Tempo 120



Continuities

Plain

MarkTime for Remaining W

Solid

Marching Errors

Errors

Collisions

Print Continuity

Note: Maintain 8/5 until Point P

"C" line ●: **Line Leader:** Form the "C" line as per

~\bsNote: Maintain 8/5 until Point P\be

"C" line \so: \bsLine Leader:\be Form the "C" line
\po: Follow Line Leader (\so) as per arrow

Welcome to Calchart vv3.6.3 | * 13 of 29 "F-O-R SS6A" 4 beats | 0 of 199 selected

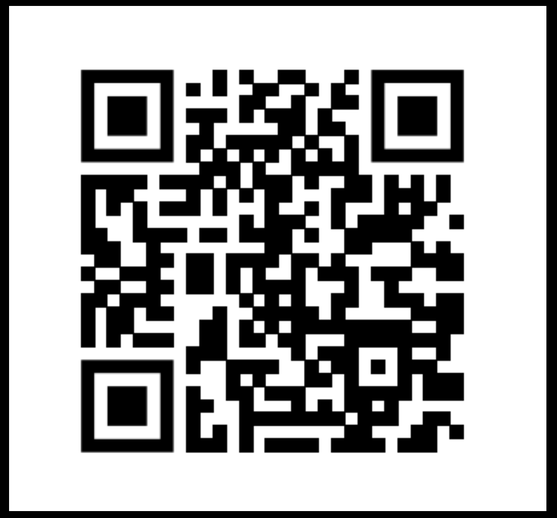
wxWidgets is a C++ library that lets developers create applications for Windows, macOS, Linux and other platforms with a single code base. It has popular language bindings for Python, Perl, Ruby and many other languages, and unlike other cross-platform toolkits, wxWidgets gives applications a truly native look and feel because it uses the platform's native API rather than emulating the GUI. It's also extensive, free, open-source and mature.

wxWidgets is a C++ library that lets developers create applications for Windows, macOS, Linux and other platforms with a single code base. It has popular language bindings for Python, Perl, Ruby and many other languages, and unlike other cross-platform toolkits, wxWidgets gives applications a truly native look and feel because it uses the platform's native API rather than emulating the GUI. It's also extensive, free, open-source and mature.

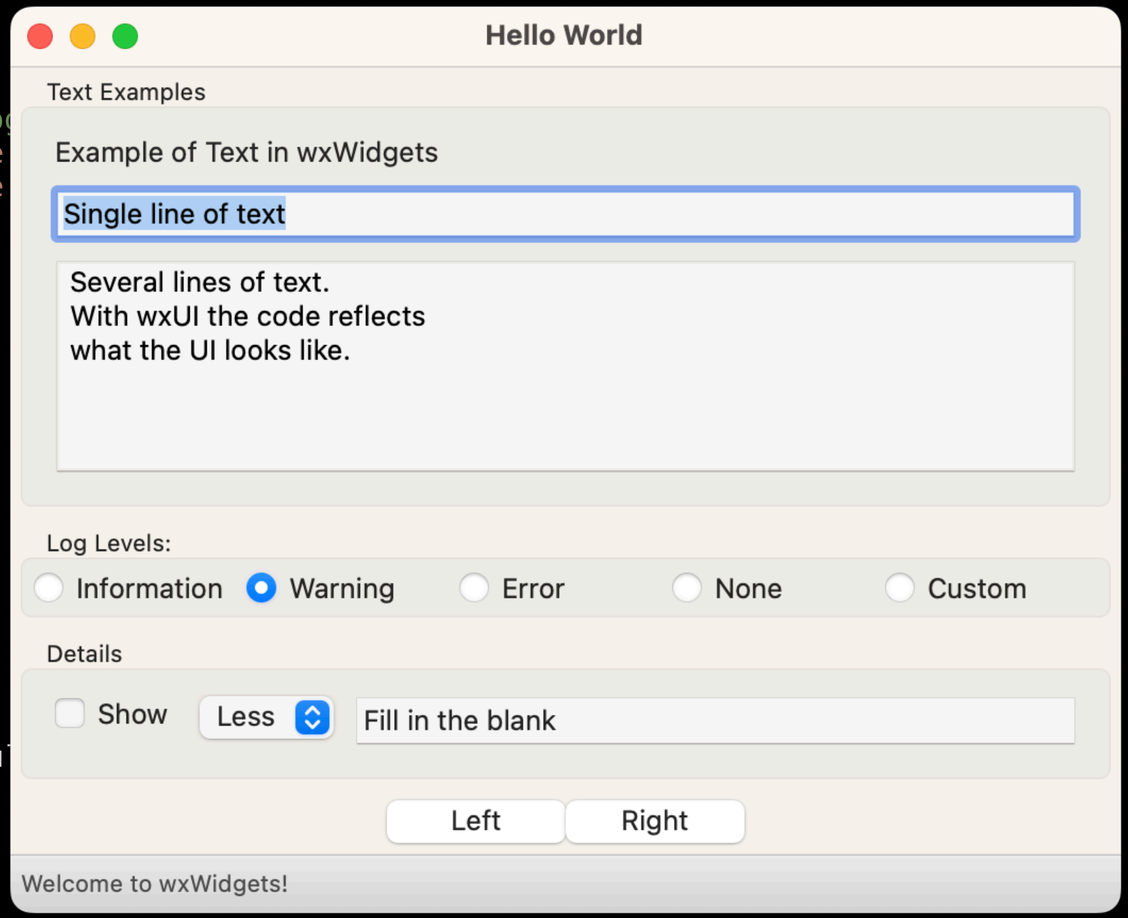
Let’s make an App!

Demo time

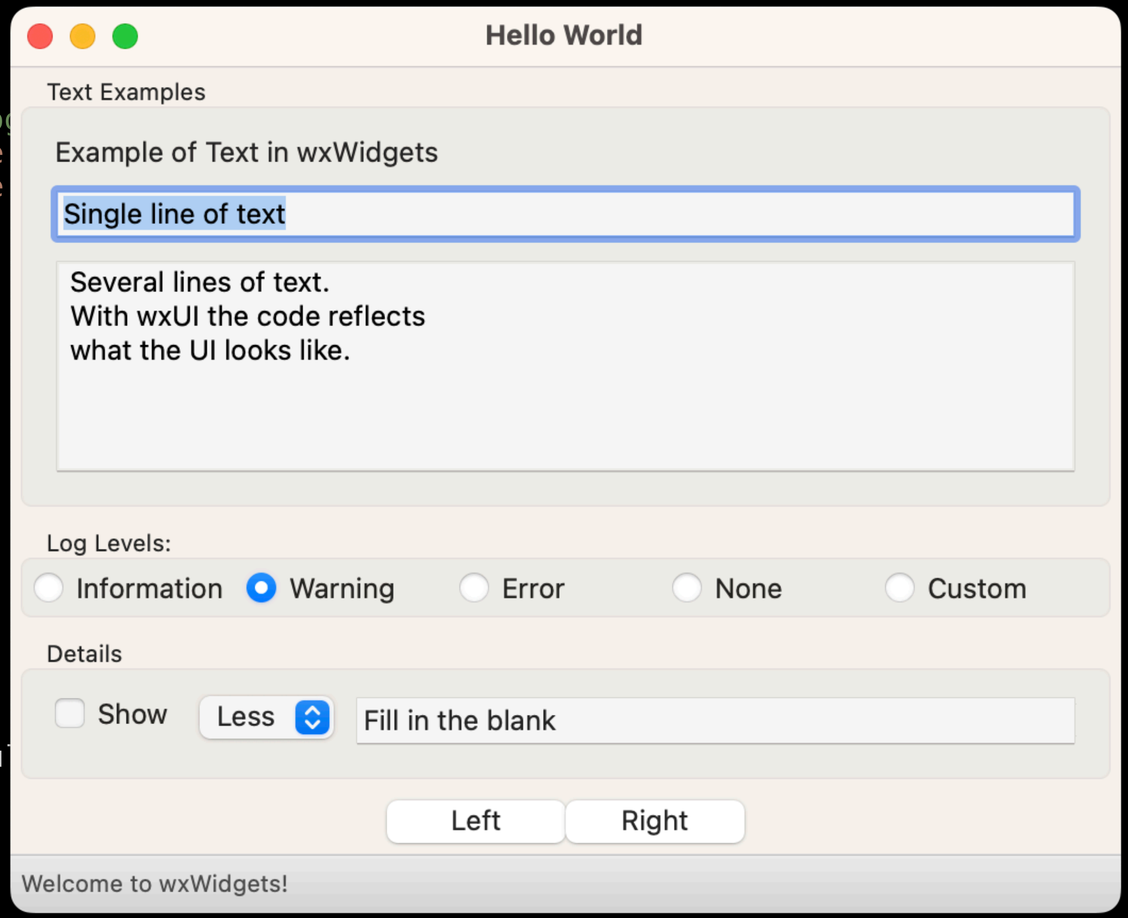
https://github.com/rmpowell77/wxHelloWorld



```
menuBar->Append(menuFile, "&File");
menuBar->Append(menuHelp, "&Help");

SetMenuBar(menuBar);

CreateStatusBar();
SetStatusText("Welcome to wxWidgets!");

// Create the controls.  This is cribbed from the RichTipDialo
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Example
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Single
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefau
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
```

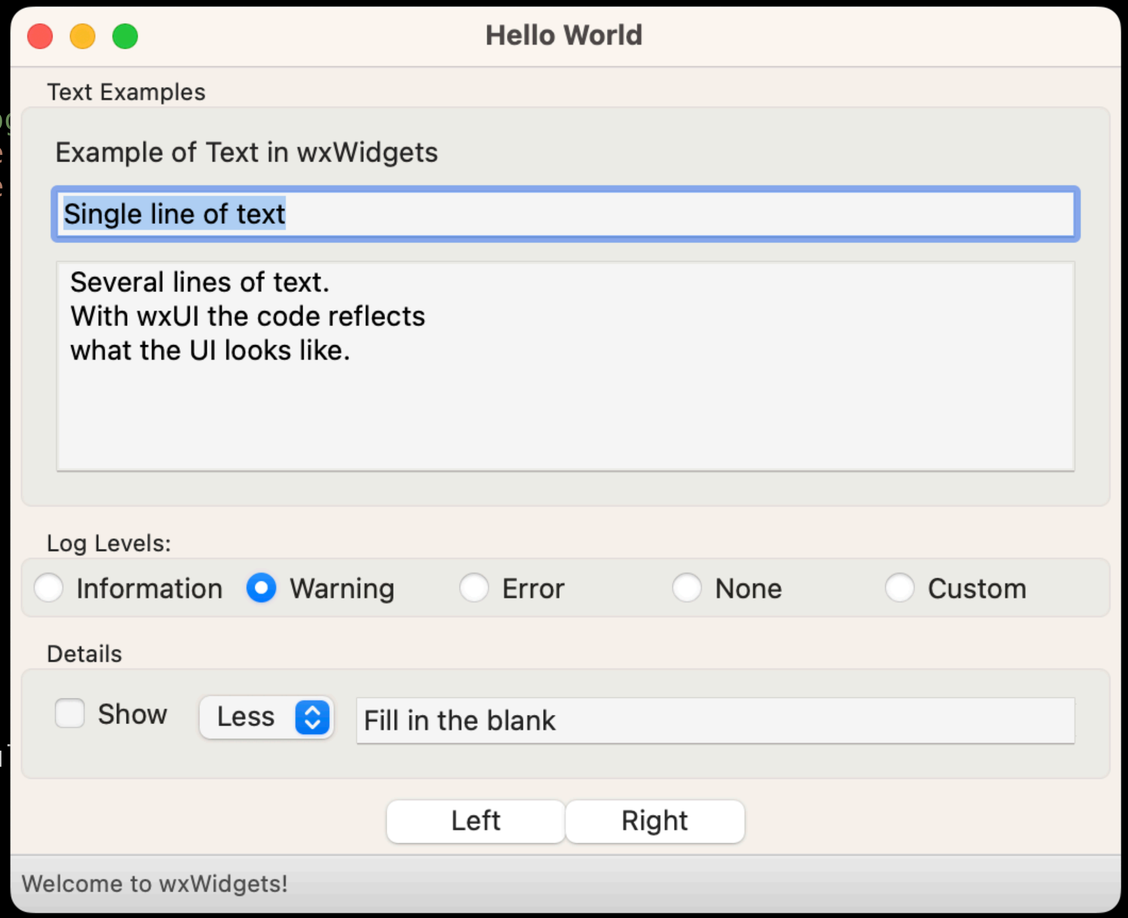


```
menuBar->Append(menuFile, "&File");
menuBar->Append(menuHelp, "&Help");

SetMenuBar(menuBar);

CreateStatusBar();
SetStatusText("Welcome to wxWidgets!");

// Create the controls.  This is cribbed from the RichTipDialo
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Example
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Single
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefau
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
```



```
menuBar->Append(menuFile, "&File");
menuBar->Append(menuHelp, "&Help");

SetMenuBar(menuBar);

CreateStatusBar();
SetStatusText("Welcome to wxWidgets!");

// Create the controls.  This is cribbed from the RichTipDialo
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Example
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Single
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefau
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
```

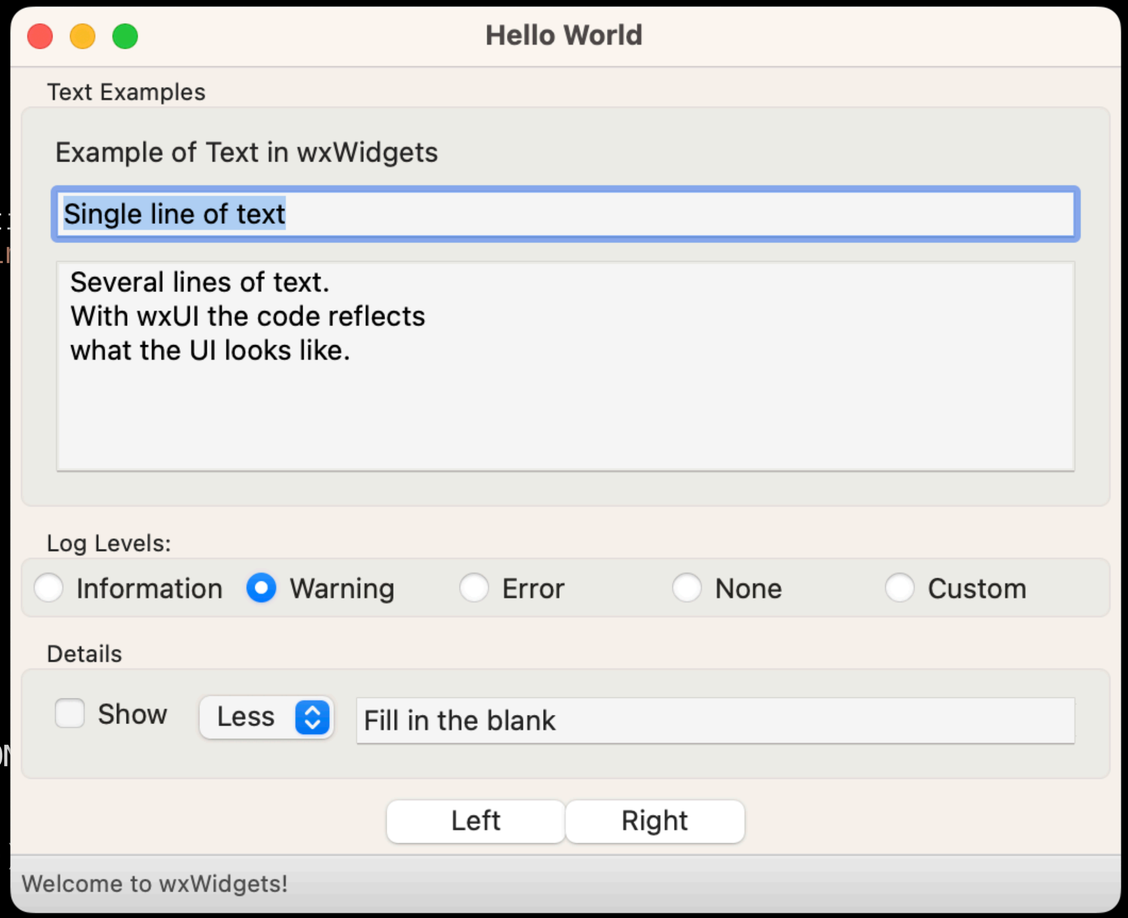
```
menuBar->Append(menuFile, "&File");
menuBar->Append(menuHelp, "&Help");

SetMenuBar(menuBar);

CreateStatusBar();
SetStatusText("Welcome to wxWidgets!");

// Create the controls.  This is cribbed from the RichTipDialo
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Example
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Single
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefau
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
```



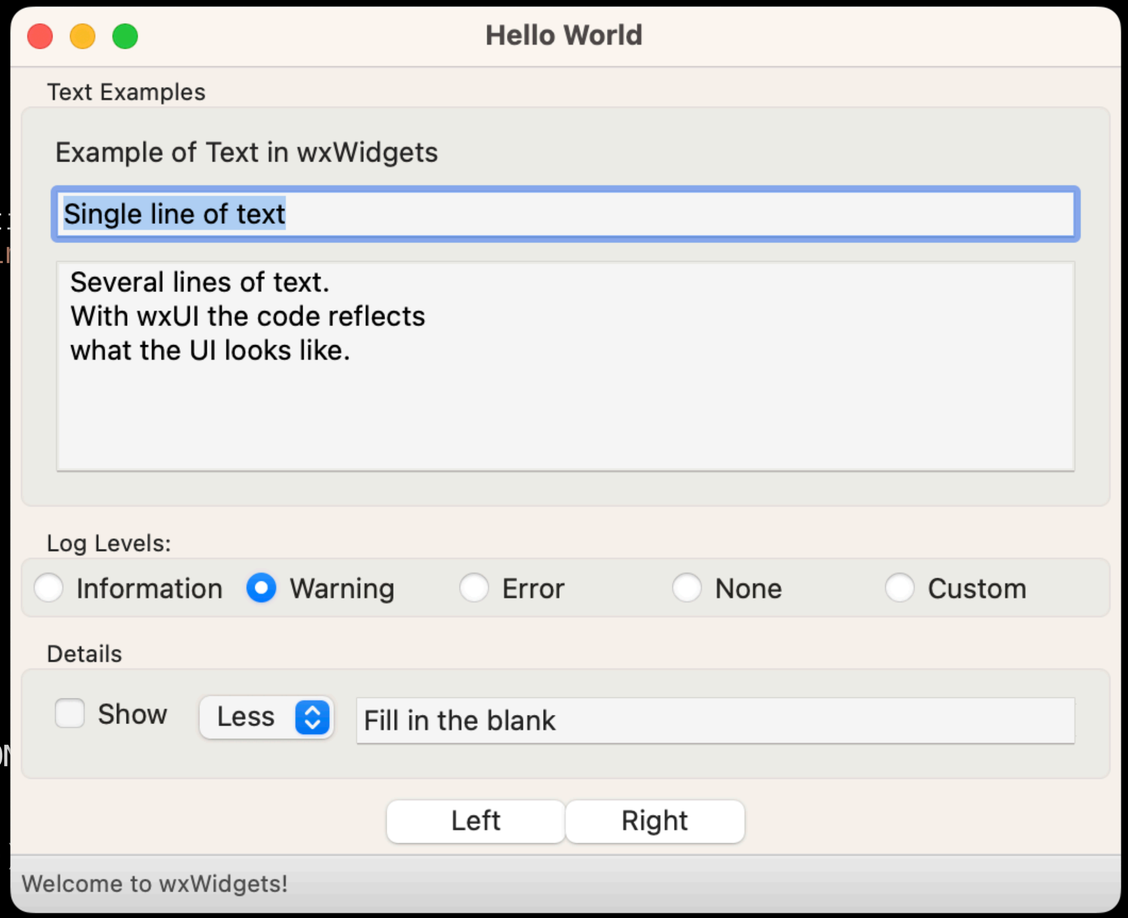
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Examples");
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "Details");
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Examples");
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "Details");
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

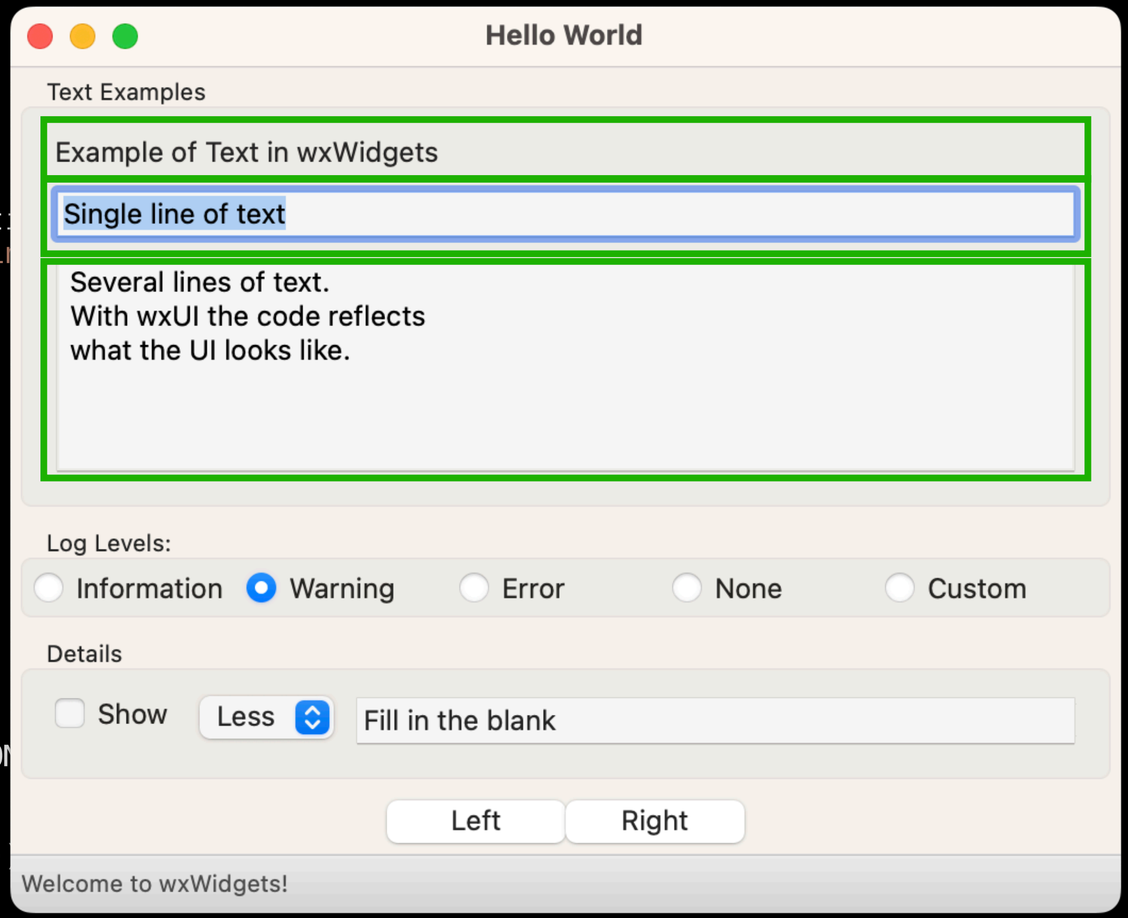
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



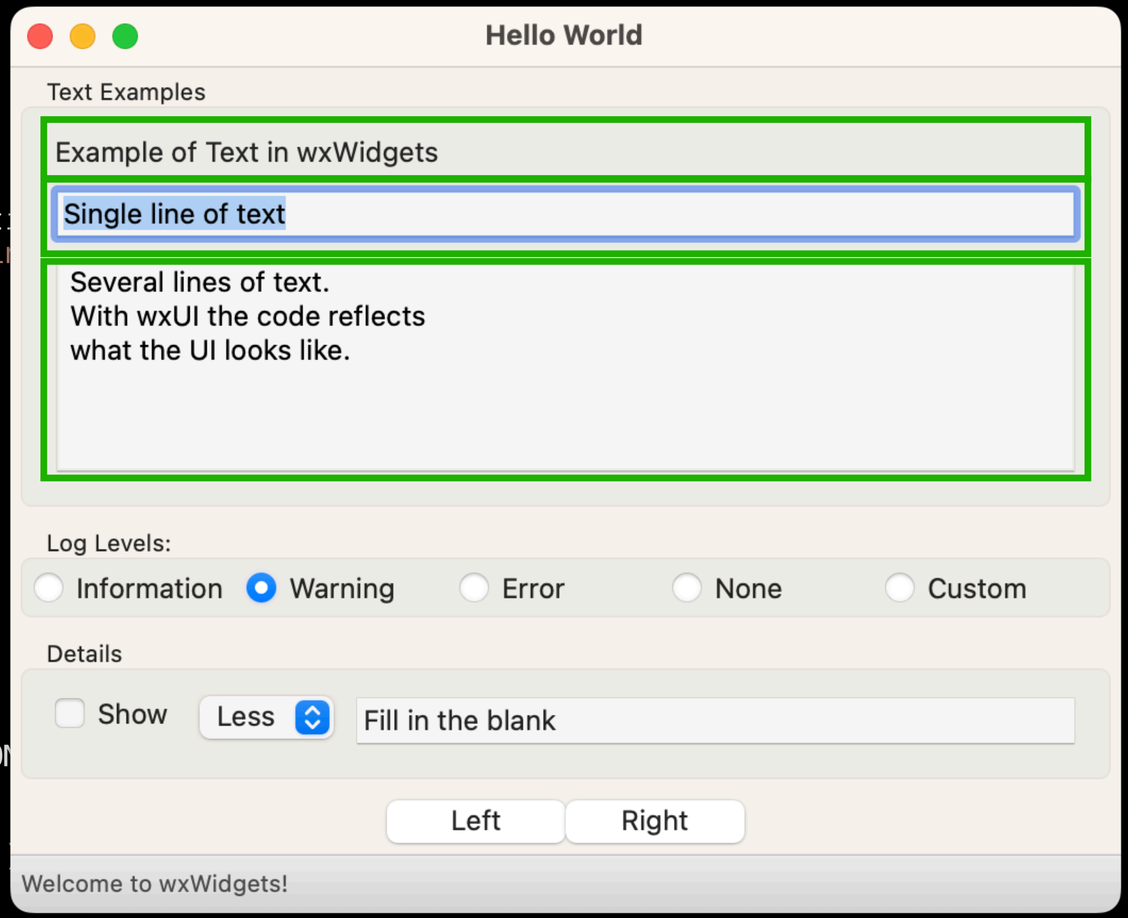
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



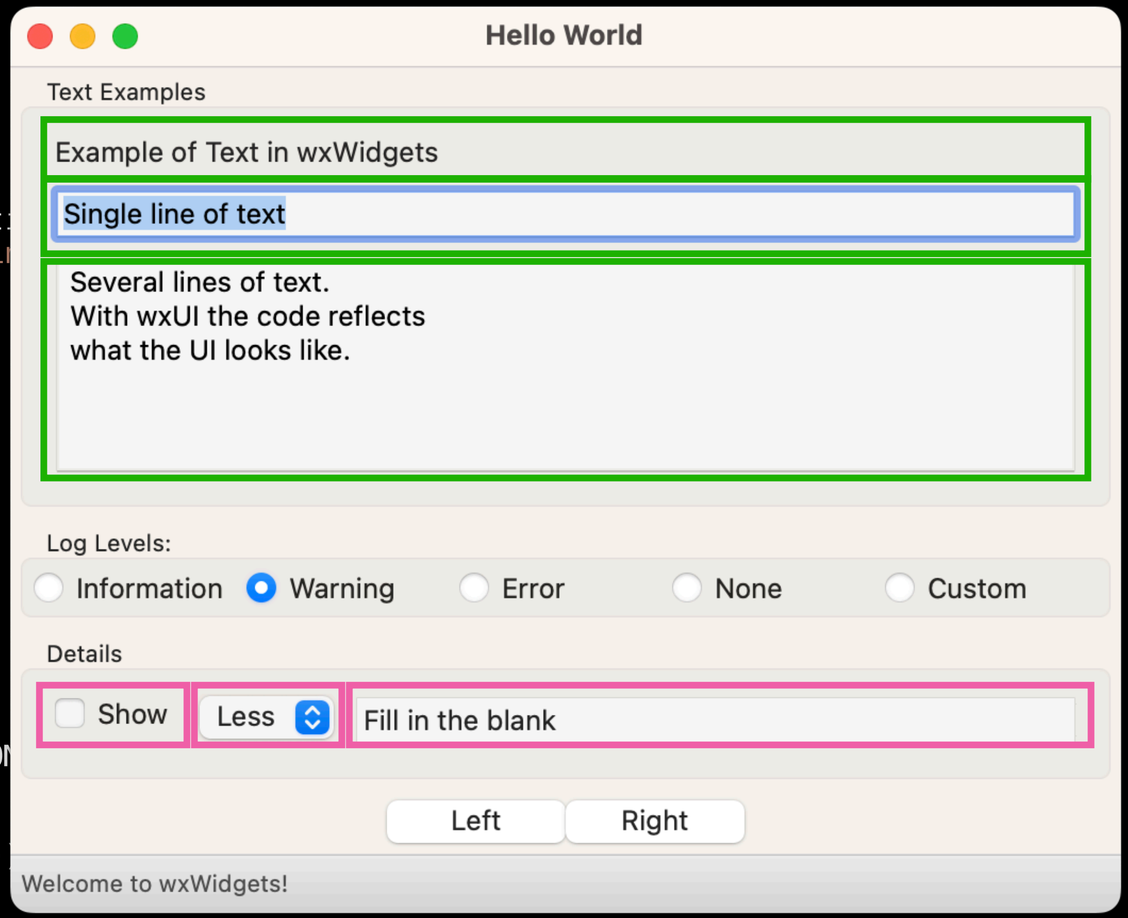
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



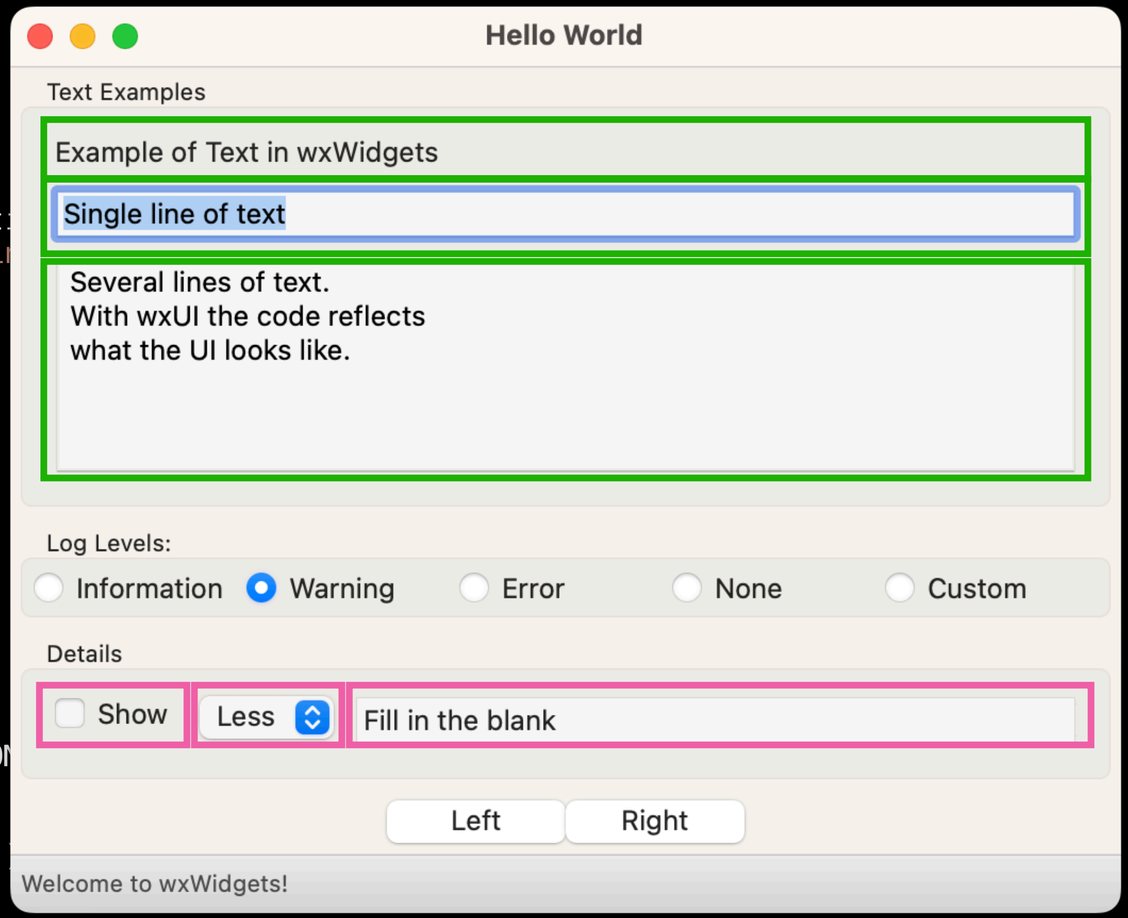
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



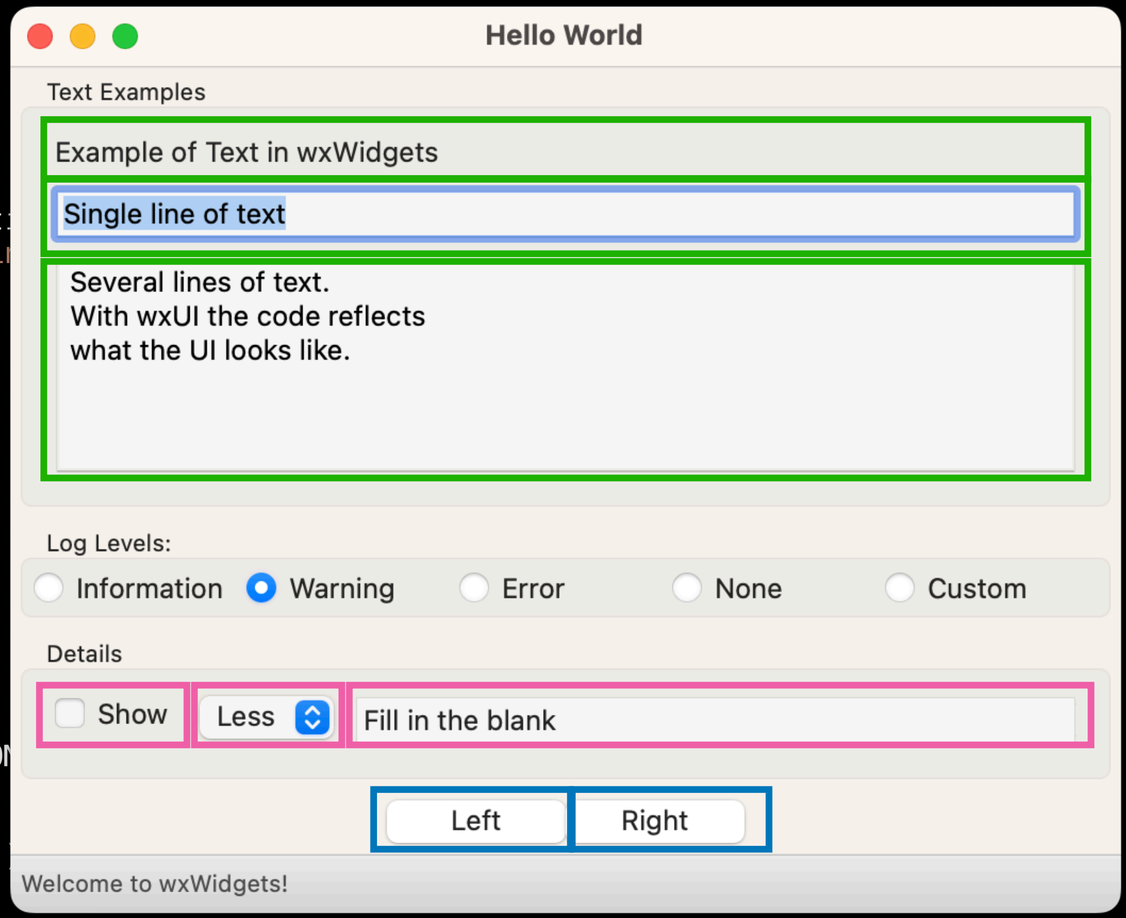
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



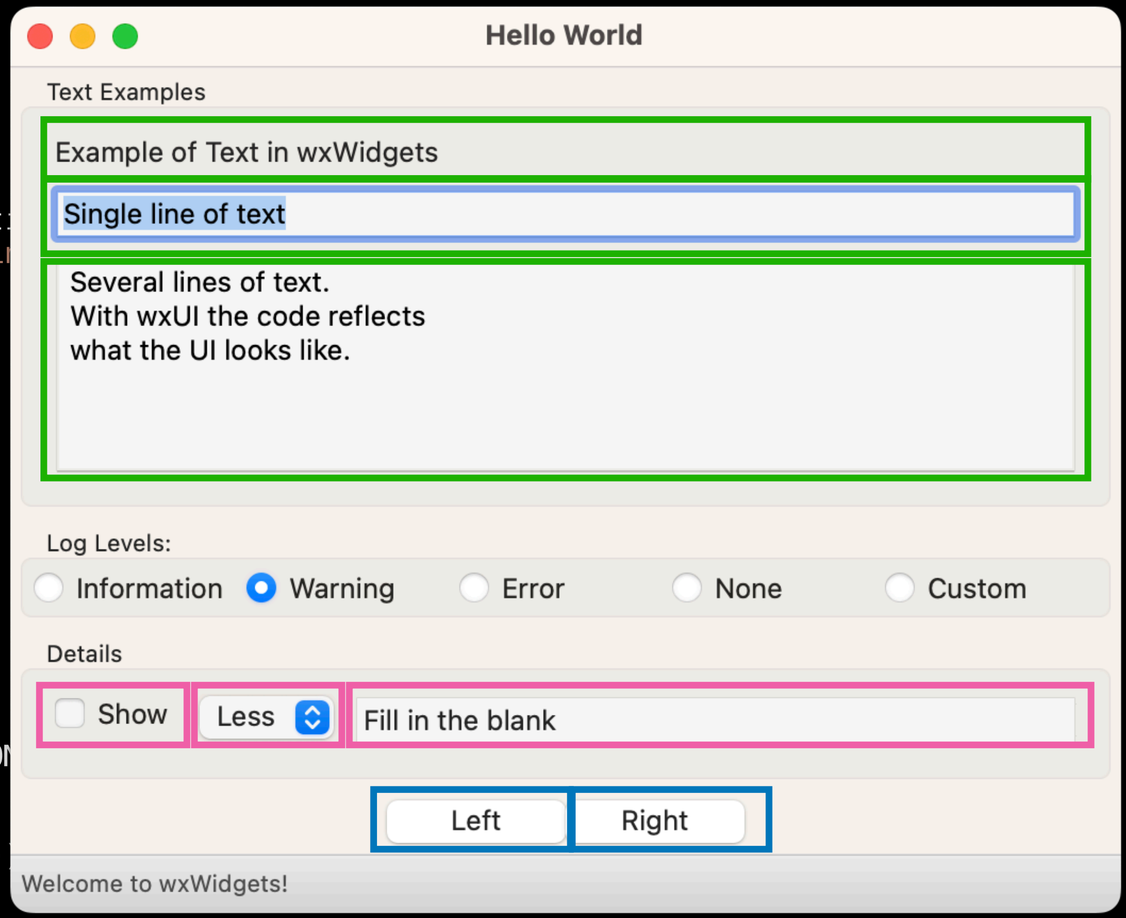
```cpp
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



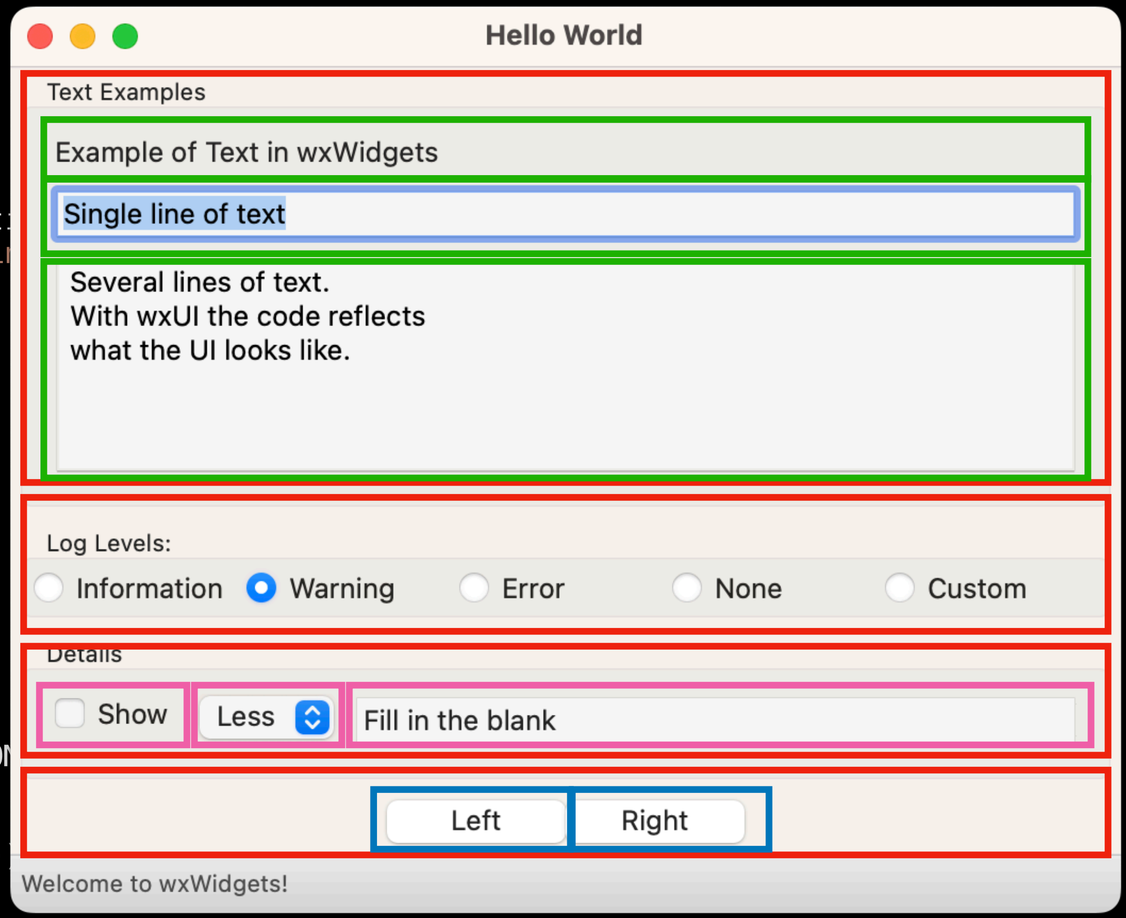
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



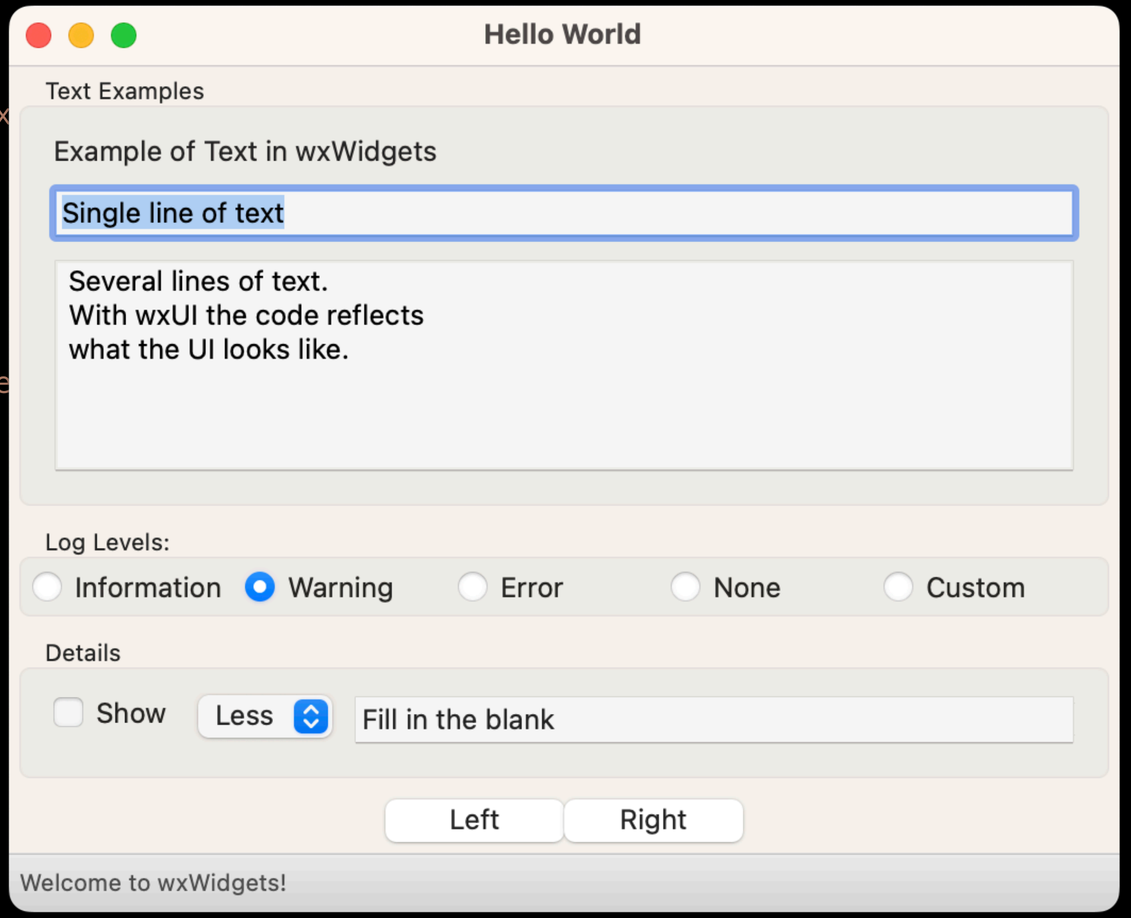
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



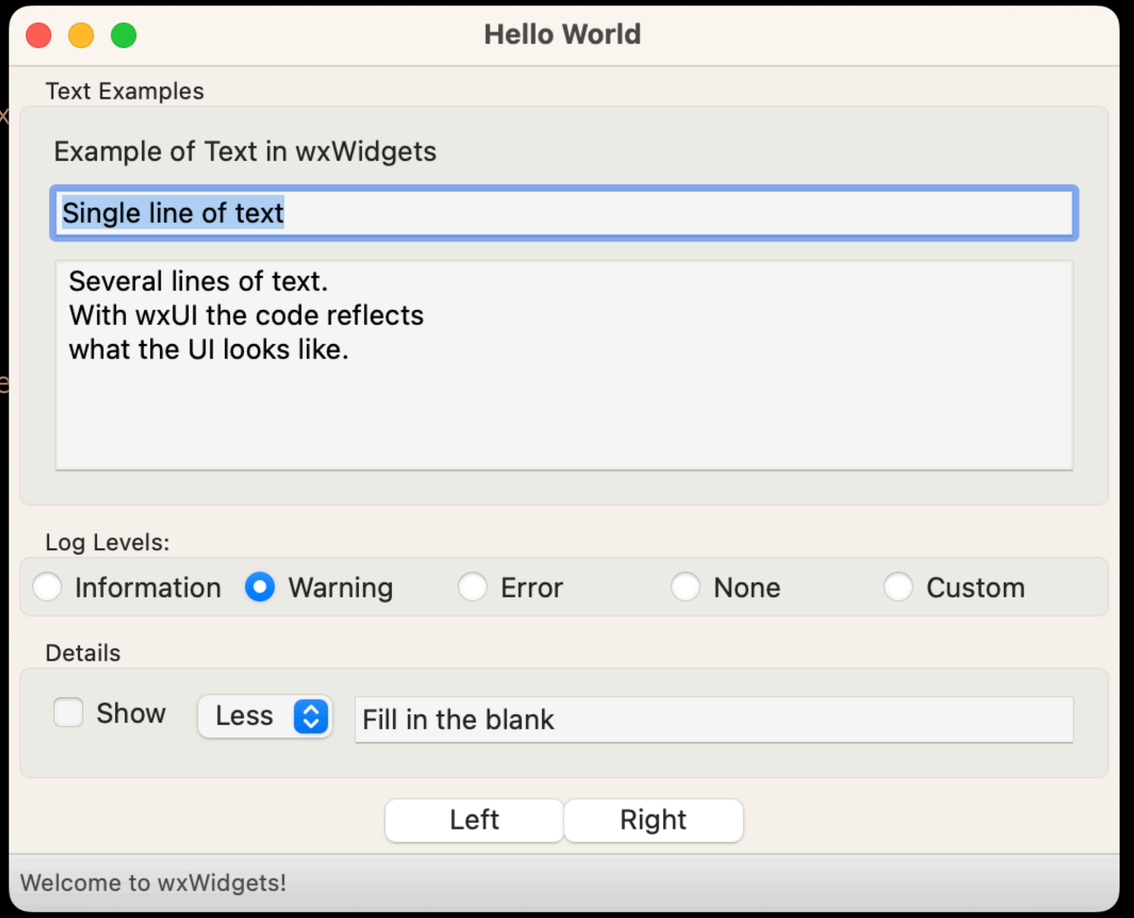
```
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPositi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill i

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border()
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



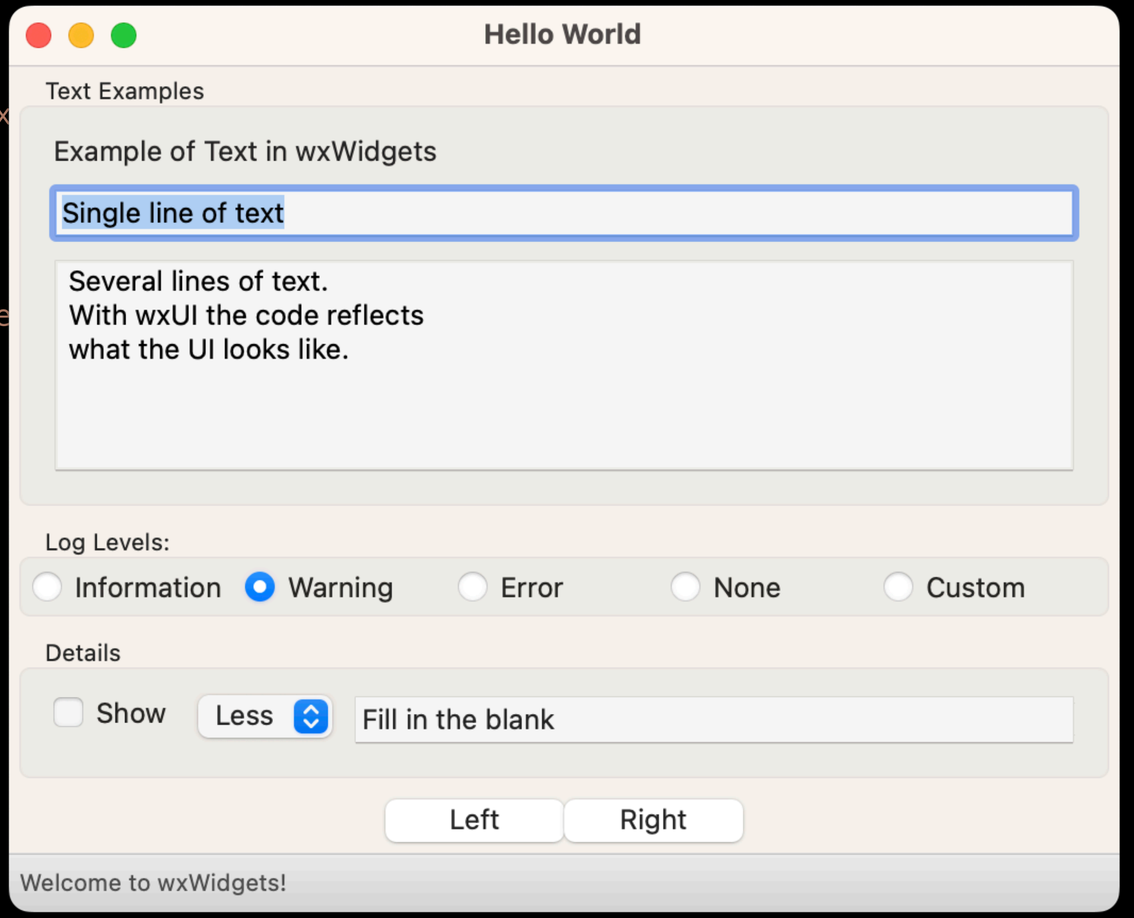
```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



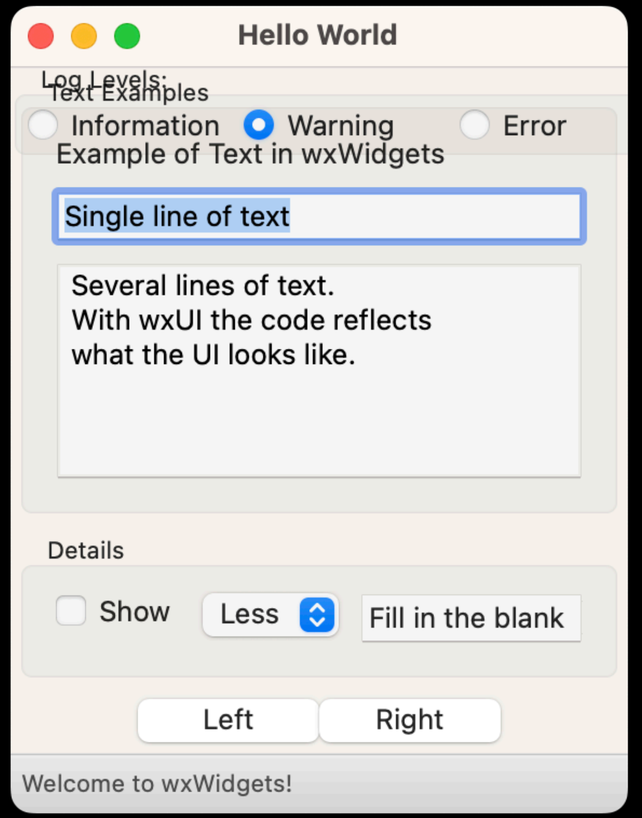
```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



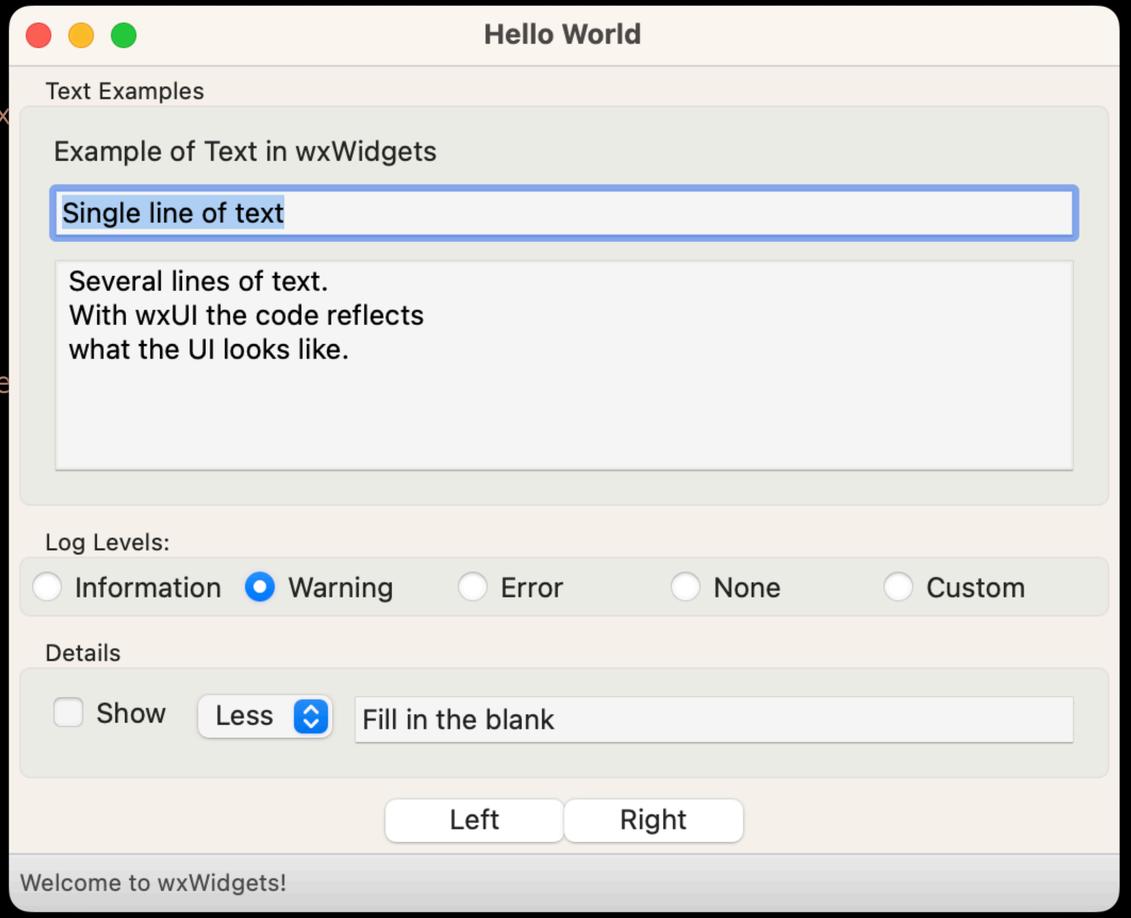
```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



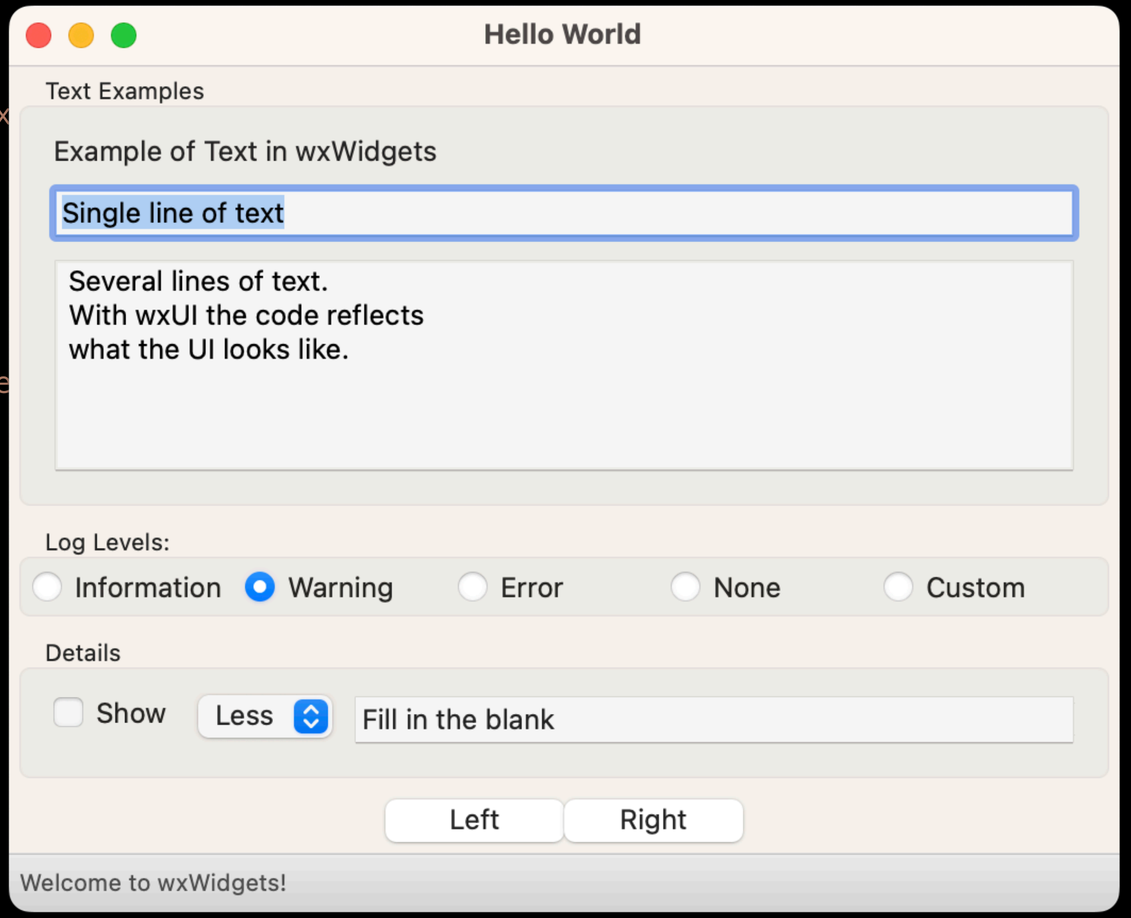
```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Examples");
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "Details");
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



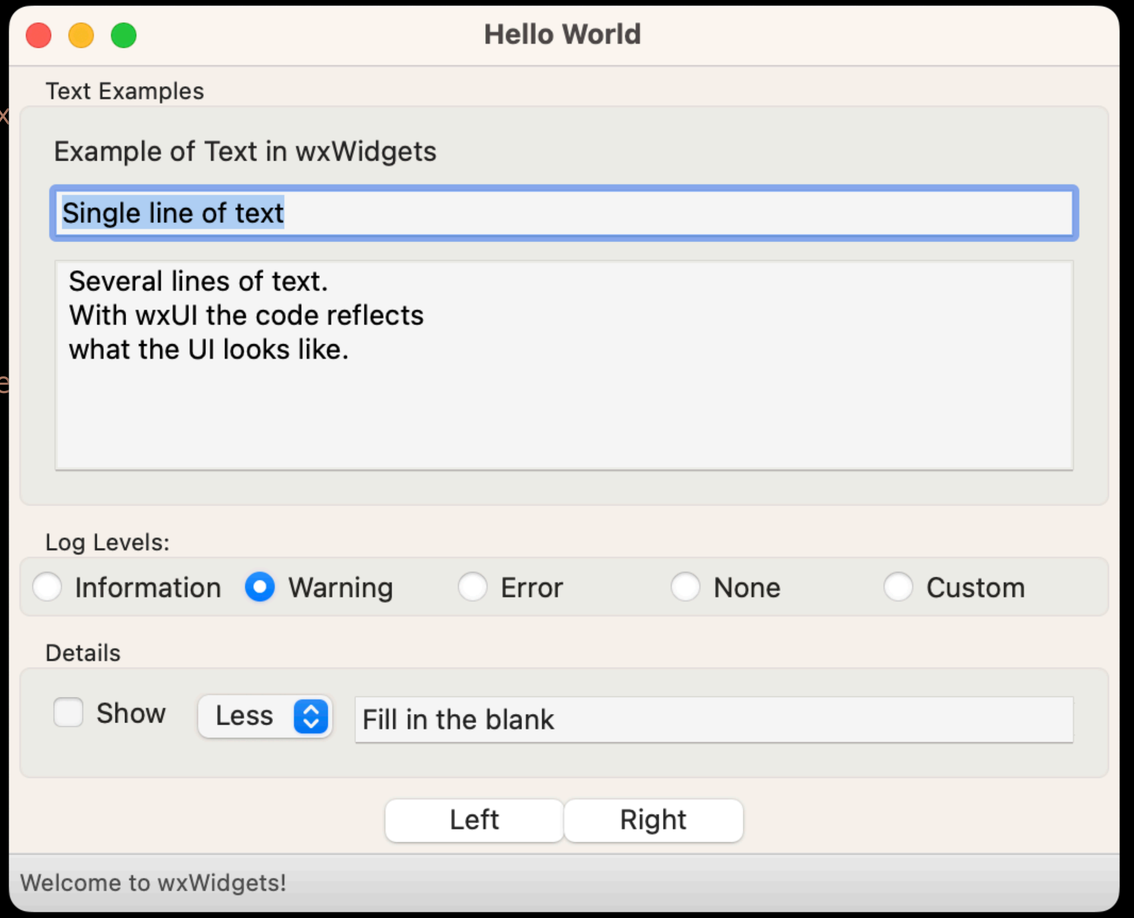
```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

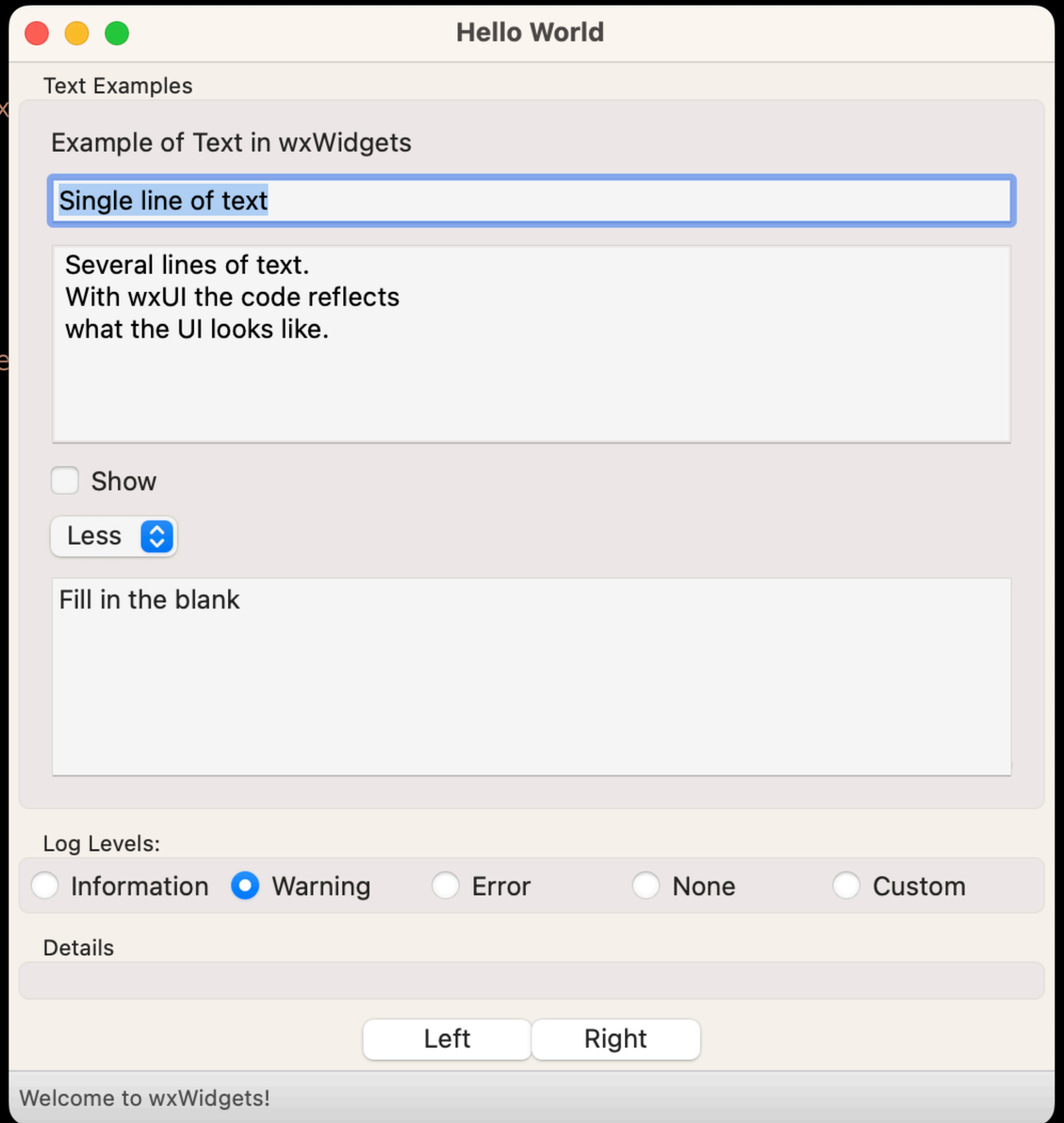


```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

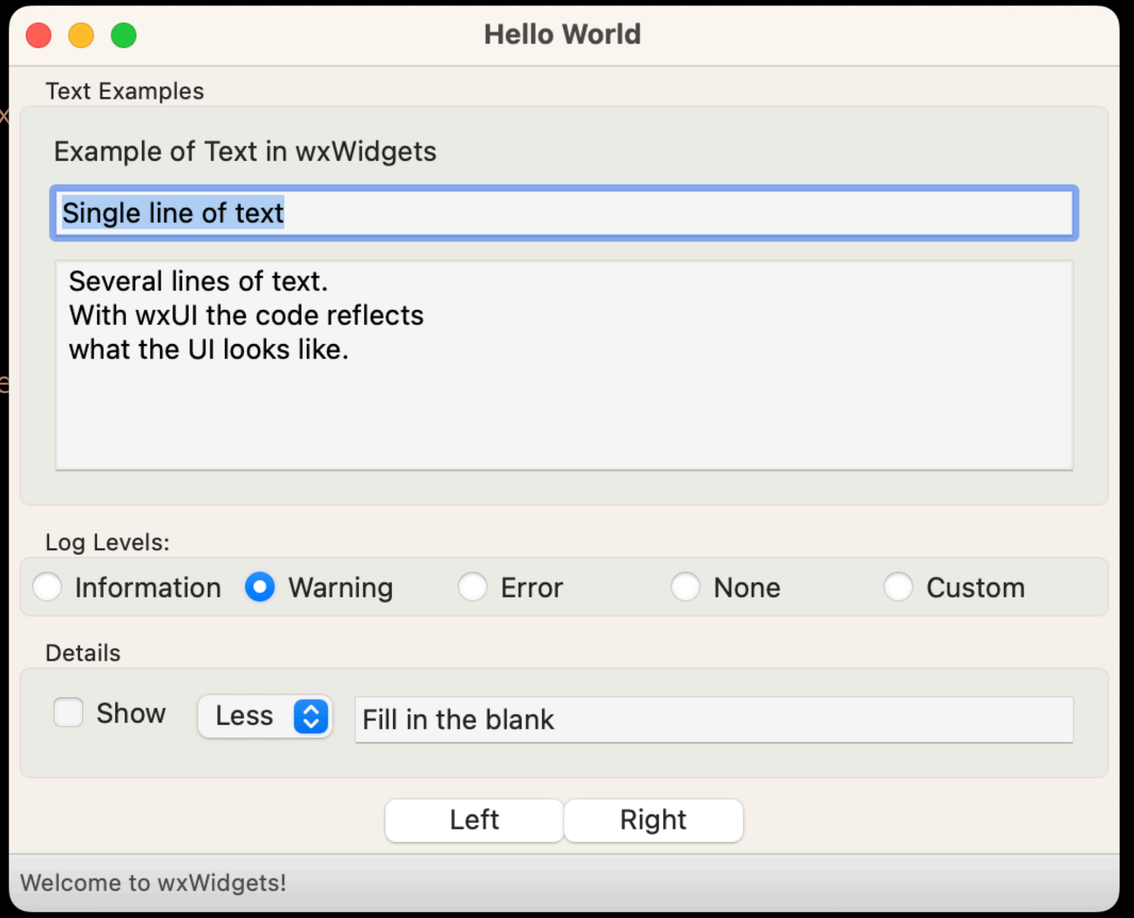


```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerText->Add(checkBox, wxSizerFlags().Border());
sizerText->Add(choice, wxSizerFlags().Border());
sizerText->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

```cpp
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerText->Add(checkBox, wxSizerFlags().Border());
sizerText->Add(choice, wxSizerFlags().Border());
sizerText->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



```cpp
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Ex
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "De
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```



```
// Create the controls.  This is cribbed from the RichTipDial
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Exampl
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Singl
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefa
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show")
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill

textBody->SetMinSize(wxSize(200, 100));
```

```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border())
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

“Declarative programming is a non-imperative style of programming in which programs describe their desired results without explicitly listing commands or steps that must be performed.”

https://en.wikipedia.org/wiki/Declarative_programming

Declarative

```
// Create the controls.  This is cribbed from the RichTipDial
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Exampl
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Singl
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefa
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show")
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill

textBody->SetMinSize(wxSize(200, 100));
```

Imperative

```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border())
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

(more) Declarative

```
// Create the controls.  This is cribbed from the RichTipDial
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Exampl
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Singl
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefa
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show")
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill

textBody->SetMinSize(wxSize(200, 100));
```

(more) Imperative

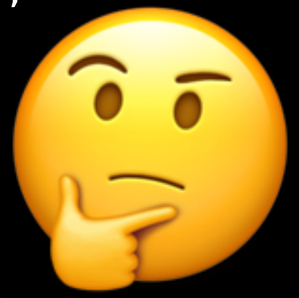
```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border())
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

```
// Create the controls.  This is cribbed from the RichTipDial
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Exampl
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Singl
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefa
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show")
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill

textBody->SetMinSize(wxSize(200, 100));
```

```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border())
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

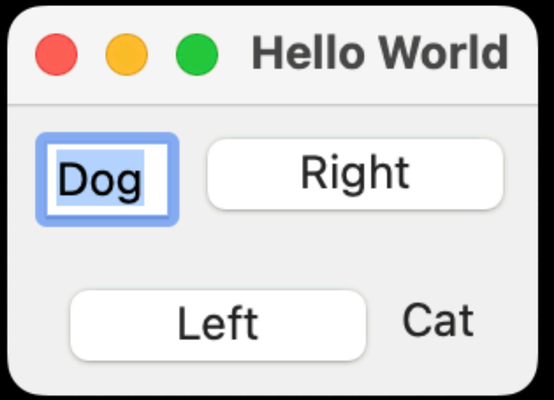
How can we make Layout more Declarative?

```
// Create the controls.  This is cribbed from the RichTipDial
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Exampl
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Singl
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefa
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show")
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosi
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill

textBody->SetMinSize(wxSize(200, 100));
```



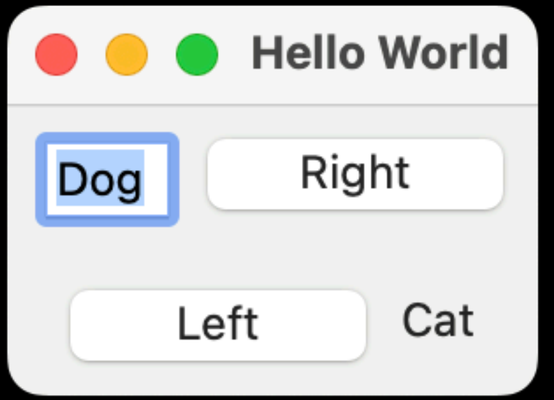
```
// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL,
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZON
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border())
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
    wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
    wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

    wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```



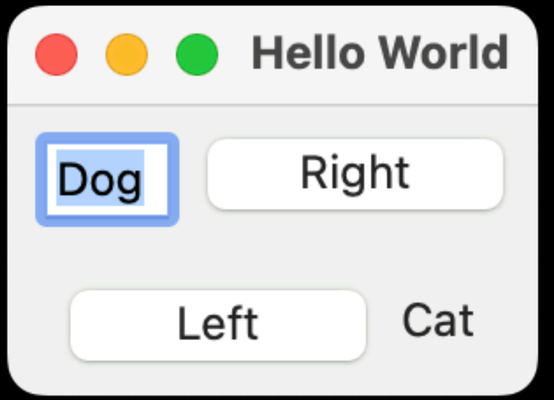
```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
    wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
    wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

    wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Declare the Controls



```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
    wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
    wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

    wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Layout the Controls



```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
    wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
    wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

    wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Almost Always Auto

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
    wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
    wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

    wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Almost Always Auto

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    auto* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    auto* btnRight = new wxButton(this, wxID_ANY, "Right");
    auto* btnLeft = new wxButton(this, wxID_ANY, "Left");
    auto* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    auto* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    auto* btnRight = new wxButton(this, wxID_ANY, "Right");
    auto* btnLeft = new wxButton(this, wxID_ANY, "Left");
    auto* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    auto* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    auto* btnRight = new wxButton(this, wxID_ANY, "Right");
    auto* btnLeft = new wxButton(this, wxID_ANY, "Left");
    auto* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create the controls.
    auto* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
    auto* btnRight = new wxButton(this, wxID_ANY, "Right");
    auto* btnLeft = new wxButton(this, wxID_ANY, "Left");
    auto* text2 = new wxStaticText(this, wxID_ANY, "Cat");

    // Layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
    sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
    sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, “Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, “Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, “Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Common

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Common

"Type" parameters

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Common

“Type” parameters

“Value” parameters

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Let’s refactor

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```
namespace DeclarativeUI {









}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
namespace DeclarativeUI {

template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}

}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Common

```
namespace DeclarativeUI {

template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}

}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```
namespace DeclarativeUI {

template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}

}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Common

“Type” parameters

```
namespace DeclarativeUI {

template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}

}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

Common

“Type” parameters

“Value” parameters

```cpp
namespace DeclarativeUI {

template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}

}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    sizerText->Add(new wxTextCtrl(this, wxID_ANY, "Dog"), wxSizerFlags(1).Expand().Border());
    sizerText->Add(new wxButton(this, wxID_ANY, "Right"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    sizerBtns->Add(new wxButton(this, wxID_ANY, "Left"), wxSizerFlags().Expand().Border());
    sizerBtns->Add(new wxStaticText(this, wxID_ANY, "Cat"), wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
namespace DeclarativeUI {

template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}

}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

    SetSizerAndFit(sizer);
}
```

```
namespace DeclarativeUI {

template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}

}

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}



MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}
```

Widget “Type”

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
template <typename Widget>
void createAndAdd(std::string str, wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
{
    sizer->Add(new Widget(parent, wxID_ANY, str), flags);
}
```

Widget “Type”

Widget “state”

```cpp
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```
template <typename W>
struct Widget {









};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

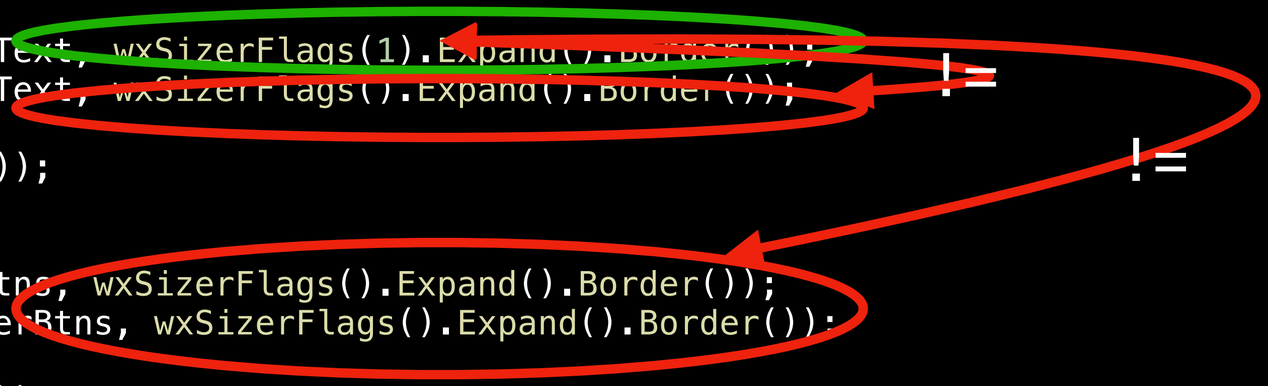
```
template <typename W>
struct Widget {
    Widget(std::string str)
        : str(str)
    {
    }









    std::string str;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```
template <typename W>
struct Widget {
    Widget(std::string str)
        : str(str)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags);
    }

    std::string str;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxTextCtrl>("Dog", this, sizerText, wxSizerFlags(1).Expand().Border());
    createAndAdd<wxButton>("Right", this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd<wxButton>("Left", this, sizerBtns, wxSizerFlags().Expand().Border());
    createAndAdd<wxStaticText>("Cat", this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```
template <typename W>
struct Widget {
    Widget(std::string str)
        : str(str)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags);
    }

    std::string str;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxTextCtrl> { "Dog" }.createAndAdd(this, sizerText, wxSizerFlags(1).Expand().Border());
    Widget<wxButton> { "Right" }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
    Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```
template <typename W>
struct Widget {
    Widget(std::string str)
        : str(str)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags);
    }

    std::string str;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxTextCtrl> { "Dog" }.createAndAdd(this, sizerText, wxSizerFlags(1).Expand().Border());
    Widget<wxButton> { "Right" }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
    Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
template <typename W>
struct Widget {
    Widget(std::string str)
        : str(str)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags);
    }

    std::string str;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxTextCtrl> { "Dog" }.createAndAdd(this, sizerText, wxSizerFlags(1).Expand().Border());
    Widget<wxButton> { "Right" }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
    Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```



```
template <typename W>
struct Widget {
    Widget(std::string str)
        : str(str)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags);
    }

    std::string str;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxTextCtrl> { "Dog" }.createAndAdd(this, sizerText, wxSizerFlags(1).Expand().Border());
    Widget<wxButton> { "Right" }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
    Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
}
```

```cpp
template <typename W>
struct Widget {
    Widget(std::string str, std::optional<wxSizerFlags> flags = {})
        : str(str)
        , flags(flags)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags parentFlags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags ? *flags : parentFlags);
    }

    std::string str;
    std::optional<wxSizerFlags> flags;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxTextCtrl> { "Dog" }.createAndAdd(this, sizerText, wxSizerFlags(1).Expand().Border());
    Widget<wxButton> { "Right" }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
    Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
```

```
template <typename W>
struct Widget {
    Widget(std::string str, std::optional<wxSizerFlags> flags = {})
        : str(str)
        , flags(flags)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags parentFlags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags ? *flags : parentFlags);
    }

    std::string str;
    std::optional<wxSizerFlags> flags;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Bo
    Widget<wxButton> { "Right" }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
    Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
```

```cpp
template <typename W>
struct Widget {
    Widget(std::string str, std::optional<wxSizerFlags> flags = {})
        : str(str)
        , flags(flags)
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags parentFlags)
    {
        sizer->Add(new W(parent, wxID_ANY, str), flags ? *flags : parentFlags);
    }

    std::string str;
    std::optional<wxSizerFlags> flags;
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame(NULL, wxID_ANY, title, pos, size)
{
    using namespace DeclarativeUI;
    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Bo
    Widget<wxButton> { "Right" }.createAndAdd(this, sizerText, wxSizerFlags().Expand().Border());

    sizer->Add(sizerText, wxSizerFlags().Expand().Border());

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
    Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());

    sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);
```

```
auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

Heterogeneous Collection

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

Heterogeneous Collection

std::tuple

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

std::apply

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

std::apply

Defined in header <tuple>

```
template< class F, class Tuple >
constexpr decltype(auto) apply( F&& f, Tuple&& t );                          (since C++17) (until C++23)

template< class F, class Tuple >
constexpr decltype(auto) apply( F&& f, Tuple&& t ) noexcept(/* see below */); (since C++23)
```

Invoke the *Callable* object f with a tuple of arguments.

```cpp
template<typename... Ts>
std::ostream& operator<<(std::ostream& os, std::tuple<Ts...> const& theTuple)
{
    std::apply
    (
        [&os](Ts const&... tupleArgs)
        {
            os << '[';
            std::size_t n{0};
            ((os << tupleArgs << (++n != sizeof...(Ts) ? ", " : "")), ...);
            os << ']';
        }, theTuple
    );
    return os;
}
```

https://en.cppreference.com/w/cpp/utility/apply

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
Widget<wxButton> { "Left" }   .createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
// Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
// Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
```

```cpp
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
// Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
// Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
```

```cpp
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
// Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
// Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
std::apply(
    [](auto&&... tupleArg) {
        (tupleArg.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border()), ...);
    },
    widgets);
```

```cpp
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
// Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
// Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
std::apply(
    [this, sizerBtns](auto&&... tupleArg) {
        (tupleArg.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border()), ...);
    },
    widgets);
```

```cpp
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
// Widget<wxButton> { "Left" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
// Widget<wxStaticText> { "Cat" }.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border());
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
std::apply(
    [this, sizerBtns](auto&&... tupleArg) {
        (tupleArg.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border()), ...);
    },
    widgets);
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
std::apply(
    [this, sizerBtns](auto&&... tupleArg) {
        (tupleArg.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border()), ...);
    },
    widgets);
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
std::apply(
    [this, sizerBtns](auto&&... tupleArg) {
        (tupleArg.createAndAdd(this, sizerBtns, wxSizerFlags().Expand().Border()), ...);
    },
    widgets);
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
std::apply(
    [this, sizerBtns, flags = wxSizerFlags().Expand().Border()](auto&&... tupleArg) {
        (tupleArg.createAndAdd(this, sizerBtns, flags), ...);
    },
    widgets);
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
auto widgets = std::tuple {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
};
std::apply(
    [this, sizerBtns, flags = wxSizerFlags().Expand().Border()](auto&&... tupleArg) {
        (tupleArg.createAndAdd(this, sizerBtns, flags), ...);
    },
    widgets);
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    auto widgets = std::tuple {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    };
    std::apply(
        [this, sizerBtns, flags = wxSizerFlags().Expand().Border()](auto&&... tupleArg) {
            (tupleArg.createAndAdd(this, sizerBtns, flags), ...);
        },
        widgets);
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
```

```
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}
```

```
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
        std::tuple {
            Widget<wxButton> { "Left" },
            Widget<wxStaticText> { "Cat" },
        });
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}

template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, W... widgets)
{
    return createAndAdd(parent, sizer, flags, std::make_tuple(widgets...));
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
        std::tuple {
            Widget<wxButton> { "Left" },
            Widget<wxStaticText> { "Cat" },
        });
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}

template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, W... widgets)
{
    return createAndAdd(parent, sizer, flags, std::make_tuple(widgets...));
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
        std::tuple {
            Widget<wxButton> { "Left" },
            Widget<wxStaticText> { "Cat" },
        });
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}

template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, W... widgets)
{
    return createAndAdd(parent, sizer, flags, std::make_tuple(widgets...));
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" });
```

```cpp
template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}

template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, W... widgets)
{
    return createAndAdd(parent, sizer, flags, std::make_tuple(widgets...));
}
```

```cpp
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" });
```

```
template <typename T>
concept CreateAndAddable
    = requires(T widget, wxWindow* window, wxSizer* sizer) {
          widget.createAndAdd(window, sizer, wxSizerFlags {});
      };

template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}

template <typename... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, W... widgets)
{
    return createAndAdd(parent, sizer, flags, std::make_tuple(widgets...));
}



    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" });
```

```
template <typename T>
concept CreateAndAddable
    = requires(T widget, wxWindow* window, wxSizer* sizer) {
          widget.createAndAdd(window, sizer, wxSizerFlags {});
      };

template <CreateAndAddable... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, std::tuple<W...> widgets)
{
    std::apply([parent, sizer, flags](auto&&... tupleArg) {
        (tupleArg.createAndAdd(parent, sizer, flags), ...);
    },
        widgets);
}

template <CreateAndAddable... W>
auto createAndAdd(wxWindow* parent, wxSizer* sizer, wxSizerFlags flags, W... widgets)
{
    return createAndAdd(parent, sizer, flags, std::make_tuple(widgets...));
}



    auto* sizer = new wxBoxSizer(wxVERTICAL);

    auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
    createAndAdd(this,
        sizerBtns,
        wxSizerFlags().Expand().Border(),
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" });
```

```
// Create the controls.
wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

sizer->Add(sizerText, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
// Create the controls.
wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
sizerText->Add(btnRight, wxSizerFlags().Expand().Border());

sizer->Add(sizerText, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
// Create and layout the controls.
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerText,
    wxSizerFlags().Expand().Border(),
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().
    Widget<wxButton> { "Right" });
sizer->Add(sizerText, wxSizerFlags().Expand().Border());

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerBtns,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" });
sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
// Create and layout the controls.
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerText,
    wxSizerFlags().Expand().Border(),
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
    Widget<wxButton> { "Right" });
sizer->Add(sizerText, wxSizerFlags().Expand().Border());

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerBtns,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" });
sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
// Create and layout the controls.
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerText,
    wxSizerFlags().Expand().Border(),
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
    Widget<wxButton> { "Right" });
sizer->Add(sizerText, wxSizerFlags().Expand().Border());

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerBtns,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" });
sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
// Create and layout the controls.
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerText,
    wxSizerFlags().Expand().Border(),
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
    Widget<wxButton> { "Right" });
sizer->Add(sizerText, wxSizerFlags().Expand().Border());

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerBtns,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" });
sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
// Create and layout the controls.
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerText,
    wxSizerFlags().Expand().Border(),
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
    Widget<wxButton> { "Right" });
sizer->Add(sizerText, wxSizerFlags().Expand().Border());

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerBtns,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" });
sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

Constructs

```
template <CreateAndAddable... W>
struct Sizer {









};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerBtns,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" });
sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());
```

```cpp
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```cpp
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
                        .createAndAdd(this);

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
                        .createAndAdd(this);

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
                        .createAndAdd(this);

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
                        .createAndAdd(this);

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerBtns = Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
                        .createAndAdd(this);

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);

        template <typename T>
        concept CreateAndAddable
            = requires(T widget, wxWindow* window, wxSizer* sizer) {
                widget.createAndAdd(window, sizer, wxSizerFlags {});
              };

    }

    w
    wx
    std::tuple<W...> widgets;
};
                                        wxSizerFlags().Expand().Border(),
                                        Widget<wxButton> { "Left" },
                                        Widget<wxStaticText> { "Cat" },
                                    }
                                        .createAndAdd(this, sizer, wxSizerFlags().Expand().Border
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    wxSizerFlags flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    std::optional<wxSizerFlags> flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }

    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags ? *flags : parentFlags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    std::optional<wxSizerFlags> flags;
    std::tuple<W...> widgets;
};
```

```
template <CreateAndAddable... W>
struct Sizer {
    Sizer(wxOrientation orientation, wxSizerFlags flags, W... widgets)
        : orientation(orientation)
        , flags(flags)
        , widgets(std::make_tuple(widgets...))
    {
    }
    Sizer(wxOrientation orientation, W... widgets)
        : orientation(orientation)
        , flags({})
        , widgets(std::make_tuple(widgets...))
    {
    }
    auto createAndAdd(wxWindow* parent, wxSizer* parentSizer, wxSizerFlags parentFlags)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([this, parent, sizer](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags ? *flags : parentFlags), ...);
        },
            widgets);
        parentSizer->Add(sizer, parentFlags);
        return sizer;
    }

    wxOrientation orientation;
    std::optional<wxSizerFlags> flags;
    std::tuple<W...> widgets;
};
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().E
```

```
template <CreateAndAddable... W>
struct HSizer : Sizer<W...> {
    HSizer(W... widgets)
        : Sizer<W...>(wxHORIZONTAL, widgets...)
    {
    }
    HSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxHORIZONTAL, flags, widgets...)
    {
    }
};

template <CreateAndAddable... W>
struct VSizer : Sizer<W...> {
    VSizer(W... widgets)
        : Sizer<W...>(wxVERTICAL, widgets...)
    {
    }
    VSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxVERTICAL, flags, widgets...)
    {
    }
};
```

```
auto* sizer = new wxBoxSizer(wxVERTICAL);

Sizer {
    wxHORIZONTAL,
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().E
```

```cpp
template <CreateAndAddable... W>
struct HSizer : Sizer<W...> {
    HSizer(W... widgets)
        : Sizer<W...>(wxHORIZONTAL, widgets...)
    {
    }
    HSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxHORIZONTAL, flags, widgets...)
    {
    }
};

template <CreateAndAddable... W>
struct VSizer : Sizer<W...> {
    VSizer(W... widgets)
        : Sizer<W...>(wxVERTICAL, widgets...)
    {
    }
    VSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxVERTICAL, flags, widgets...)
    {
    }
};
```

```cpp
auto* sizer = new wxBoxSizer(wxVERTICAL);

HSizer {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().E
```

```cpp
template <CreateAndAddable... W>
struct HSizer : Sizer<W...> {
    HSizer(W... widgets)
        : Sizer<W...>(wxHORIZONTAL, widgets...)
    {
    }
    HSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxHORIZONTAL, flags, widgets...)
    {
    }
};

template <CreateAndAddable... W>
struct VSizer : Sizer<W...> {
    VSizer(W... widgets)
        : Sizer<W...>(wxVERTICAL, widgets...)
    {
    }
    VSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxVERTICAL, flags, widgets...)
    {
    }
};
```

```cpp
auto* sizer = new wxBoxSizer(wxVERTICAL);

HSizer {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().E
```

```
// Create and layout the controls.
auto* sizer = new wxBoxSizer(wxVERTICAL);

auto* sizerText = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerText,
    wxSizerFlags().Expand().Border(),
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
    Widget<wxButton> { "Right" });
sizer->Add(sizerText, wxSizerFlags().Expand().Border());

auto* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
createAndAdd(this,
    sizerBtns,
    wxSizerFlags().Expand().Border(),
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" });
sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
// Create and layout the controls.
auto* sizer = new wxBoxSizer(wxVERTICAL);

HSizer {
    Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
    Widget<wxButton> { "Right" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());

HSizer {
    Widget<wxButton> { "Left" },
    Widget<wxStaticText> { "Cat" },
}
    .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
template <CreateAndAddable... W>
struct Sizer {
    auto attachTo(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([parent, sizer, flags = flags ? *flags : wxSizerFlags()](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parent->SetSizerAndFit(sizer);
        return sizer;
    }
};


    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    HSizer {
        Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    }
        .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());

    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    }
        .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
```

```
template <CreateAndAddable... W>
struct Sizer {
    auto attachTo(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([parent, sizer, flags = flags ? *flags : wxSizerFlags()](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parent->SetSizerAndFit(sizer);
        return sizer;
    }
};


    // Create and layout the controls.
    auto* sizer = new wxBoxSizer(wxVERTICAL);

    HSizer {
        Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    }
        .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());

    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    }
        .createAndAdd(this, sizer, wxSizerFlags().Expand().Border());

    SetSizerAndFit(sizer);
```

```
template <CreateAndAddable... W>
struct Sizer {
    auto attachTo(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([parent, sizer, flags = flags ? *flags : wxSizerFlags()](auto&&... tupleArg) {
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parent->SetSizerAndFit(sizer);
        return sizer;
    }
};


    // Create and layout the controls.
    VSizer {
        wxSizerFlags().Expand().Border(),
        HSizer {
            Widget<wxTextCtrl> { "Dog", wxSizerFlags(1).Expand().Border() },
            Widget<wxButton> { "Right" },
        },
        HSizer {
            Widget<wxButton> { "Left" },
            Widget<wxStaticText> { "Cat" },
        },
    }
        .attachTo(this);
```

```
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

```
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

```
namespace DeclarativeUI {

template <typename T>
concept CreateAndAddable
    = requires(T widget, wxWindow* window, wxSizer* sizer) {
          widget.createAndAdd(window, sizer, wxSizerFlags {});
      };
```

```cpp
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

```cpp
    }

    auto attachTo(wxWindow* parent)
    {
        auto* sizer = new wxBoxSizer(orientation);
        std::apply([parent, sizer, flags = flags ? *flags : wxSizerFl
            (tupleArg.createAndAdd(parent, sizer, flags), ...);
        },
            widgets);
        parent->SetSizerAndFit(sizer);
        return sizer;
    }

    wxOrientation orientation;
    std::optional<wxSizerFlags> flags;
    std::tuple<W...> widgets;
};

template <CreateAndAddable... W>
struct HSizer : Sizer<W...> {
    HSizer(W... widgets)
        : Sizer<W...>(wxHORIZONTAL, widgets...)
    {
    }
    HSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxHORIZONTAL, flags, widgets...)
    {
    }
};

template <CreateAndAddable... W>
struct VSizer : Sizer<W...> {
    VSizer(W... widgets)
        : Sizer<W...>(wxVERTICAL, widgets...)
    {
    }
    VSizer(wxSizerFlags flags, W... widgets)
        : Sizer<W...>(wxVERTICAL, flags, widgets...)
    {
    }
};
}
```

```
// Create the controls.
wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
sizerText->Add(btnRight, wxSizerFlags().Border());

sizer->Add(sizerText, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

Combining Controllers and Layout makes it easier to see intention.

```
// Create the controls.
wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
sizerText->Add(btnRight, wxSizerFlags().Border());

sizer->Add(sizerText, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

Impossible to forget to add a controller to the layout.

```
// Create the controls.
wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
sizerText->Add(btnRight, wxSizerFlags().Border());

sizer->Add(sizerText, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

Adding things to an incorrect Sizer greatly reduced.

```
// Create the controls.
wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
sizerText->Add(btnRight, wxSizerFlags().Border());

sizer->Add(sizerText, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

Very "DRY"

```
// Create the controls.
wxTextCtrl* text1 = new wxTextCtrl(this, wxID_ANY, "Dog");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxStaticText* text2 = new wxStaticText(this, wxID_ANY, "Cat");

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxBoxSizer* sizerText = new wxBoxSizer(wxHORIZONTAL);
sizerText->Add(text1, wxSizerFlags(1).Expand().Border());
sizerText->Add(btnRight, wxSizerFlags().Border());

sizer->Add(sizerText, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Expand().Border());
sizerBtns->Add(text2, wxSizerFlags().Expand().Border());

sizer->Add(sizerBtns, wxSizerFlags().Expand().Border());

SetSizerAndFit(sizer);
```

```
using namespace DeclarativeUI;
// Create and layout the controls.
VSizer {
    wxSizerFlags().Expand().Border(),
    HSizer {
        Widget<wxTextCtrl> { "Dog",
            wxSizerFlags(1).Expand().Border() },
        Widget<wxButton> { "Right" },
    },
    HSizer {
        Widget<wxButton> { "Left" },
        Widget<wxStaticText> { "Cat" },
    },
}
    .attachTo(this);
```

This is just the beginning…

This is just the beginning…

```
// Create the controls.  This is cribbed from the RichTipDialog example
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Example of Text in wxWidgets");
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Single line of text");
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Examples");
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "Details");
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

This is just the beginning…

```
// Create the controls.  This is cribbed from the RichTipDialog example
wxStaticText* text = new wxStaticText(this, wxID_ANY, "Example of Text in wxWidgets");
wxTextCtrl* textTitle = new wxTextCtrl(this, wxID_ANY, "Single line of text");
wxTextCtrl* textBody = new wxTextCtrl(this, wxID_ANY,
    "Several lines of text.\n"
    "With wxUI the code reflects\n"
    "what the UI looks like.",
    wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
wxButton* btnLeft = new wxButton(this, wxID_ANY, "Left");
wxButton* btnRight = new wxButton(this, wxID_ANY, "Right");

const wxString levels[] = {
    "&Information",
    "&Warning",
    "&Error",
    "&None",
    "&Custom",
};
wxRadioBox* logLevels = new wxRadioBox(
    this, wxID_ANY, "&Log Levels:", wxDefaultPosition, wxDefaultSize,
    WXSIZEOF(levels), levels, 1, wxRA_SPECIFY_ROWS);
logLevels->SetSelection(1);

wxCheckBox* checkBox = new wxCheckBox(this, wxID_ANY, "Show");
const wxString choices[] = {
    "Less",
    "More",
};

wxChoice* choice = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, WXSIZEOF(choices), choices);
wxTextCtrl* blankLine = new wxTextCtrl(this, wxID_ANY, "Fill in the blank");

textBody->SetMinSize(wxSize(200, 100));

// Layout the controls.
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);

wxStaticBoxSizer* sizerText = new wxStaticBoxSizer(wxVERTICAL, this, "Text Examples");
sizerText->Add(text, wxSizerFlags().Expand().Border());
sizerText->Add(textTitle, wxSizerFlags().Expand().Border());
sizerText->Add(textBody, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerText, wxSizerFlags(1).Expand().Border());

sizer->Add(logLevels, wxSizerFlags().Expand().Border());

wxStaticBoxSizer* sizerDetails = new wxStaticBoxSizer(wxHORIZONTAL, this, "Details");
sizerDetails->Add(checkBox, wxSizerFlags().Border());
sizerDetails->Add(choice, wxSizerFlags().Border());
sizerDetails->Add(blankLine, wxSizerFlags(1).Expand().Border());
sizer->Add(sizerDetails, wxSizerFlags().Expand().Border());

wxBoxSizer* sizerBtns = new wxBoxSizer(wxHORIZONTAL);
sizerBtns->Add(btnLeft, wxSizerFlags().Border(wxLEFT));
sizerBtns->Add(btnRight, wxSizerFlags().Border(wxRIGHT));
sizer->Add(sizerBtns, wxSizerFlags().Centre().Border());

SetSizerAndFit(sizer);
```

```
VSizer {
    wxSizerFlags().Expand().Border(),
    VSizer {
        "Text examples",
        Text { "Example of Text in wxUI" },
        TextCtrl { "Single line of text" }
            .withStyle(wxALIGN_LEFT),
        TextCtrl {
            "Several lines of text.\n"
            "With wxUI the code reflects\n"
            "what the UI looks like." }
            .withStyle(wxTE_MULTILINE)
            .withSize(wxSize(200, 100))
    },

    RadioBox { "&Log Levels:", { "&Information", "&Warning", "&Error", "&No
        .withStyle(wxRA_SPECIFY_ROWS)
        .withMajorDim(1)
        .withSelection(1),

    HSizer {
        "Details",
        CheckBox { "Show" },
        Choice { { "Less", "More" } },
        TextCtrl { wxSizerFlags(1).Expand().Border(), "Fill in the blank" }
            .withStyle(wxALIGN_LEFT),
    },

    HSizer {
        wxSizerFlags().Center().Border(),
        Button { wxSizerFlags().Border(wxRIGHT), "Left" }
            .bind([] { wxLogMessage("Pressed Left"); }),
        Button { wxSizerFlags().Border(wxLEFT), "Right" }
            .bind([](wxCommandEvent&) { wxLogMessage("Pressed Right"); }),
    },
}
    .attachTo(this);
```

- Observer in your program sections that are *Imperative* and try to think *Declarative*.

- Observer in your program sections that are *Imperative* and try to think *Declarative*.
  - Change the “How to do” parts into “What I want” code.

- Observer in your program sections that are *Imperative* and try to think *Declarative*.
  - Change the “How to do” parts into “What I want” code.
- Refactoring can cause Patterns to emerge.

- Observer in your program sections that are *Imperative* and try to think *Declarative*.
  - Change the “How to do” parts into “What I want” code.
- Refactoring can cause Patterns to emerge.
- Know your types, algorithms, and emerging C++ constructs.

Thank You

Questions?