slides.md

Development and use of support tools and libraries in programming courses

Ole Christian Eidheim

Applied Information Technology, IDI Kalvskinnet

Programming languages used

- IDRI1002 Informatics 1 and IDRI1005 Object-oriented programming with systems analysis
  - JavaScript
- TDAT2003 Software Engineering 2 with web applications
  - JavaScript
- TDAT2004 Data Communications and Network Programming
  - C++/Java/Rust and a voluntary programming language (not Python/JavaScript)
- TDAT3025 Applied Machine Learning with Project
  - Python
- TDAT3020 Network and Software Security
  - Assembly/C/C++/JavaScript
- TDAT3023 3D Computer Graphics with project
  - C++/Java/C#/Python

- IINI4003 C++ for Programmers

Open Source libraries/tools

- https://gitlab.com/eidheim/Simple-Web-Server
  - C++ HTTP(S) library
  - Stars: 1628 + 54
  - Forked by Tesla Motors
- https://gitlab.com/eidheim/Simple-WebSocket-Server
  - C++ WS(S) library
  - Stars: 535 + 13
- https://gitlab.com/cppit/jucipp
  - C++ IDE
  - Stars: 883 + 62
- https://gitlab.com/eidheim/tiny-process-library
  - C++ computer process library
  - Stars: 224 + 5
- https://gitlab.com/eidheim/react-simplified
  - JavaScript UI-library built on top of React.js
  - Stars: 0 + 3

tiny-process-library

Platform independent:

```
#include "process.hpp"
#include <iostream>
```

```
using namespace std;
using namespace TinyProcessLib;

int main() {
  Process process("echo Hello World", "", [](const char *bytes, size_t n) {
    cout.write(bytes, n);
  });

  int exit_status = process.get_exit_status();
  cout << "Exit status: " << exit_status << endl;
}

// Output:
// Hello World
// Exit status: 0
```

Unix-like systems:

```
#include "process.hpp"
#include <iostream>

using namespace std;
using namespace TinyProcessLib;

int main() {
  Process process([] {
    cout << "Hello World\n";
    exit(0);
  }, [](const char *bytes, size_t n) {
    cout.write(bytes, n);
  });

  int exit_status = process.get_exit_status();
  cout << "Exit status: " << exit_status << endl;
```

```
}

// Output:
// Hello World
// Exit status: 0
```

react-simplified

Example GUI libraries

Dear ImGui

```
...

int main() {

    ...

    int counter = 0;

    while (run) {
        ...

        ImGui::Begin("ImGui");

        if (ImGui::Button("Increase counter"))
          ++counter;

        ImGui::Text("Counter: %d", counter);

        ImGui::End();

        ...
    }
```

```
...

}
```

Example GUI libraries

gtkmm

```
...

int main() {
  Gtk::Main gtk_main;

  Gtk::Window window;

  Gtk::VBox vbox;
  Gtk::HBox hbox;

  Gtk::Button button;
  Gtk::Label text_label;
  Gtk::Label counter_label;

  int counter = 0;

  window.set_title("Gtk example");

  button.set_label("Increase counter");
  button.signal_clicked().connect([&] {
    counter_label.set_label(std::to_string(++counter));
  });

  text_label.set_label("Counter: ");
  counter_label.set_label("0");
```

```
  vbox.add(button);
  hbox.add(text_label);
  hbox.add(counter_label);
  vbox.add(hbox);

  window.add(vbox);
  window.show_all();

  gtk_main.run(window);
}
```

Example GUI libraries

gtkmm using UI XML

```
...

int main() {
  Gtk::Main gtk_main;

  auto builder = Builder::create();
  builder->add_from_file("view.glade");

  auto window = builder->get_window_widget<Gtk::Window>("window");

  auto button = builder->get_widget<Gtk::Button>("button");
  auto counter_label = builder->get_widget<Gtk::Label>("counter_label");

  int counter = 0;

  button->signal_clicked().connect([&] {
    counter_label->set_label(std::to_string(++counter));
  });

  window->show_all();
```

```
  gtk_main.run(*window);
}
```

React

```
...

class App extends React.Component<{}, { counter: number }> {
  state = { counter: 0 };

  constructor() {
    super();

    this.increaseCounter = this.increaseCounter.bind(this);
  }

  render() {
    return (
      <div>
        <div>
          <button onClick={this.increaseCounter}>Increase counter</button>
        </div>
        Counter: {this.state.counter}
      </div>
    );
  }

  increaseCounter() {
    this.setState({ counter: this.state.counter + 1 });
  }
}

const root = document.getElementById('root');
if (root) ReactDOM.render(<App />, root);
```

react-simplified

```
...

class App extends Component {
  counter = 0;

  render() {
    return (
      <div>
        <div>
          <button onClick={this.increaseCounter}>Increase counter</button>
        </div>
        Counter: {this.counter}
      </div>
    );
  }

  increaseCounter() {
    this.counter++;
  }
}

const root = document.getElementById('root');
if (root) ReactDOM.render(<App />, root);
```

juCi++ (jucipp)

Goal for next semester: early support for Lifetime profile

- Example use: https://godbolt.org/z/szJjnH.