Удаленный клиент для симулятора управления воздушным движением

Лазаревич Андрей Андреевич

научный руководитель: Е.И. Жидков

СПб АУ НОЦНТ РАН

13 июня 2016 г.

Введение

- Высокая потребность в обучении авиадиспетчеров
- Высокая стоимость оборудования и обучения
- Переход на распределенную архитектуру с тонким клиентом
- Итеративный переход, как можно меньше меняющий исходное приложение

Возможное решение проблемы

Запуск клиента в ”облаке” с дальнейшей трансляцией через удаленный рабочий стол

- Отсутствие поддержки OpenGL > 1.1
- Увеличенная задержка пользовательского интерфейса (>1000ms)
- Подробности тестирования приложений в облаке в статье [1]

[1] Bernhard Tritsch, Kristin Griffin, Ruben Spruijt ”Comparing remoteapp performance on Microsoft Azure and On-Premises deployments”, March 2015

Цели и задач

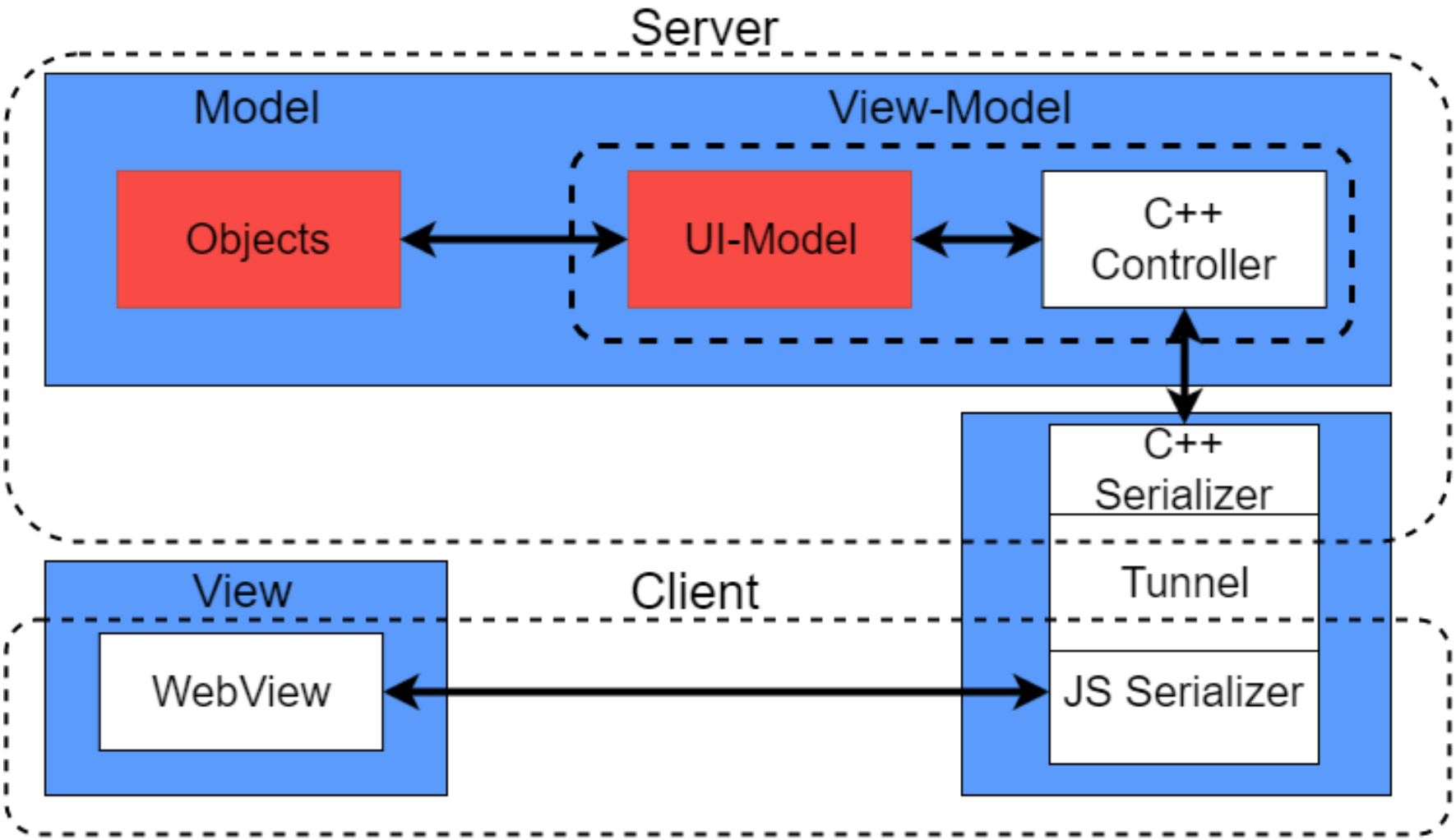
Цель:

- Разработать удаленный клиент для симулятора управления воздушным движением

Задачи:

- Разработать механизм сериализации и десериализации удаленных вызовов C++ и JS
  - Разработка протокола для обмена вызовами и данными, как со стороны C++, так и JS,
  - Разработка сервера и клиента на обоих сторонах приложения, обеспечивающих удаленный вызов методов
- Обеспечить интеграцию веб-интерфейсов в текущее приложение
  - Выбор фреймворка для интеграции браузера в текущее приложение
  - Разработка и тестирование некоторых элементов UI

Выбранная архитектура



Протокол

```
{
  "client_id": 0, // Идентификатор клиента
  "object_id": 1, /* Идентификатор объекта,
  у которого происходит вызов метода */
  "method_name": "foo", //Имя вызываемого метода
  "params": ["some", "params"], // Сериализованные параметры
  "request_id": 13 /* Идентификатор запроса
  для возвращаемого значения */
}
```

```
{
  "response_id": 13, /* Идентификатор ответа
  для сопоставления с запросом */
  "return_value": "success" /* Сериализованное возвращаемое
  значение */
}
```

Возможности:

- Регистрация C++ объекта с методами, как объекта-контроллера в JS
- Вызов зарегистрированных JS методов

Порядок действий:

- Регистрация сервером удаленных вызовов
- Обработка событий на клиенте
- Получение сообщения о возможности вызывать JS методы

Javascript

Возможности:

- Регистрация JS методов в C++
- Вызов зарегистрированных C++ методов у контроллера

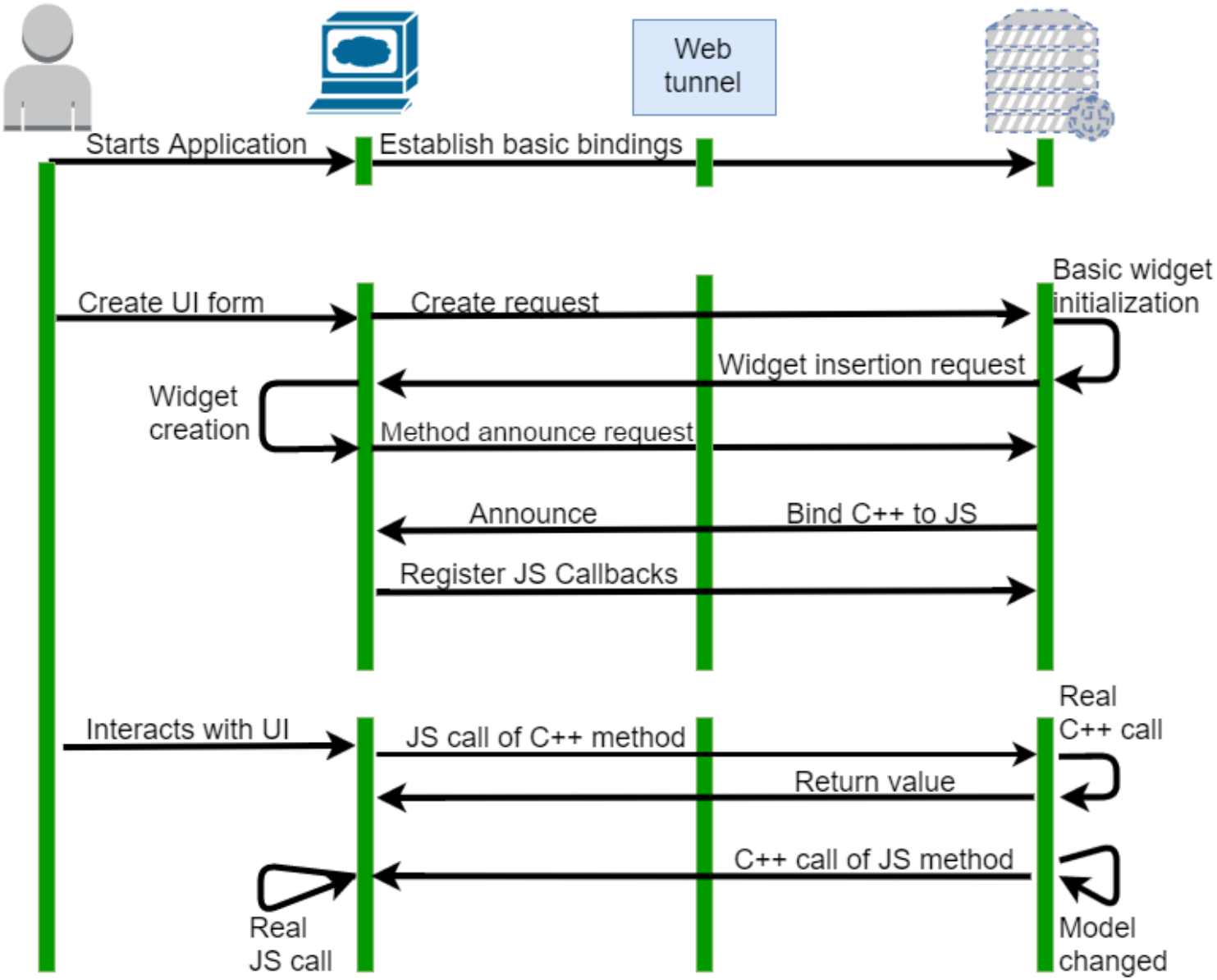
Порядок действий:

- Обработка событий на сервере
- Регистрация клиентом удаленных вызовов

Интеграция браузера

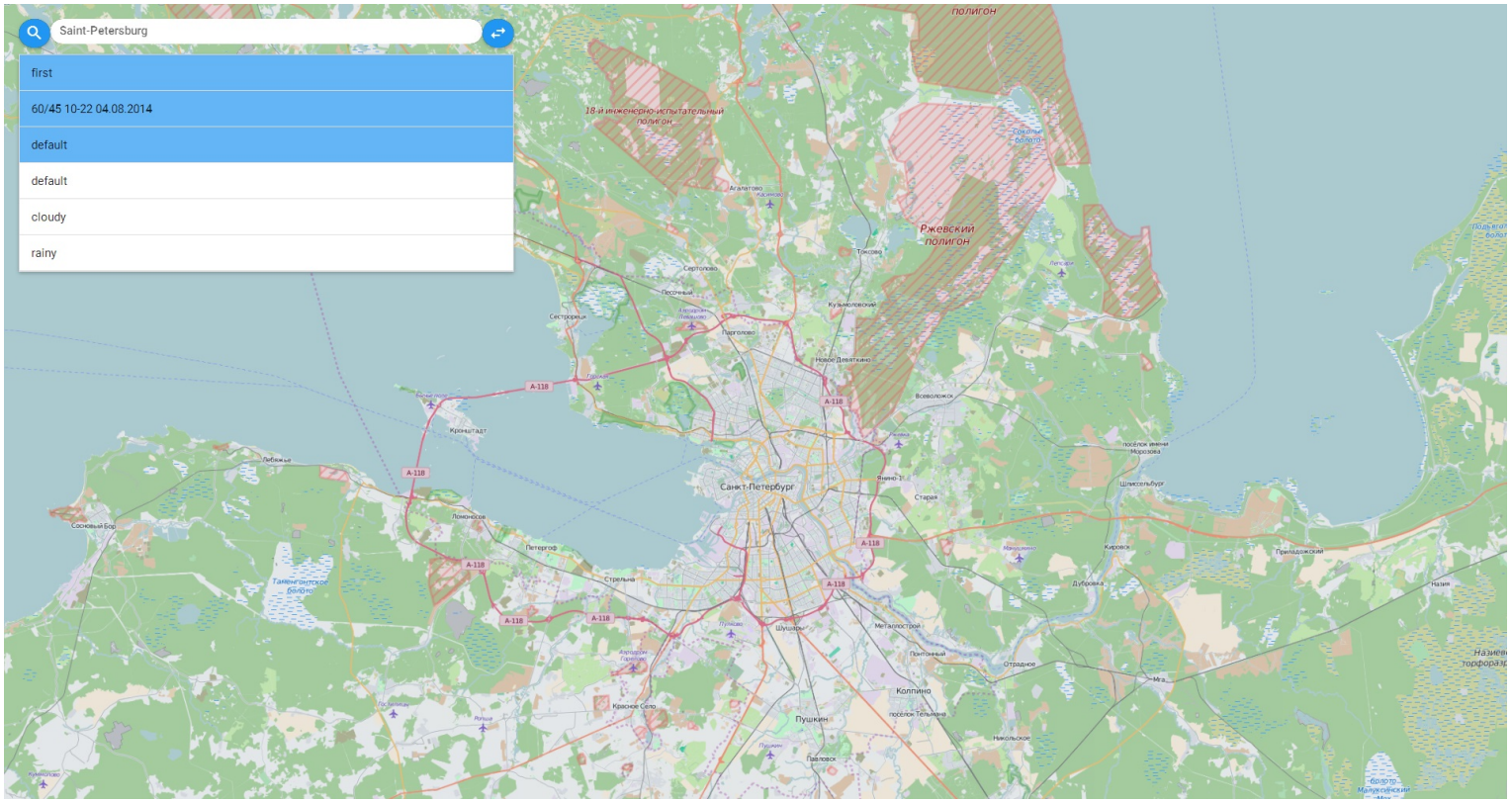
- Awesomium
  - Высокий уровень абстракции
  - Высокая производительность
  - Закрытый код, платная лицензия
- ChromiumEmbedded Framework(CEF)
  - Низкий уровень абстракции
  - Высокая производительность
  - Открытый исходный код

Workflow


Web
tunnel
Starts Application
Establish basic bindings
Create UI form
Create request
Basic widget
initialization
Widget insertion request
Widget
creation
Method announce request
Announce
Bind C++ to JS
Register JS Callbacks
Interacts with UI
JS call of C++ method
Real
C++ call
Return value
C++ call of JS method
Real
JS call
Model
changed

Результаты

- Спроектирована и разработана система для сериализации и десериализации вызовов методов C++ и Javascript
- Созданы и интегрированы в текущее приложение некоторые веб-компоненты

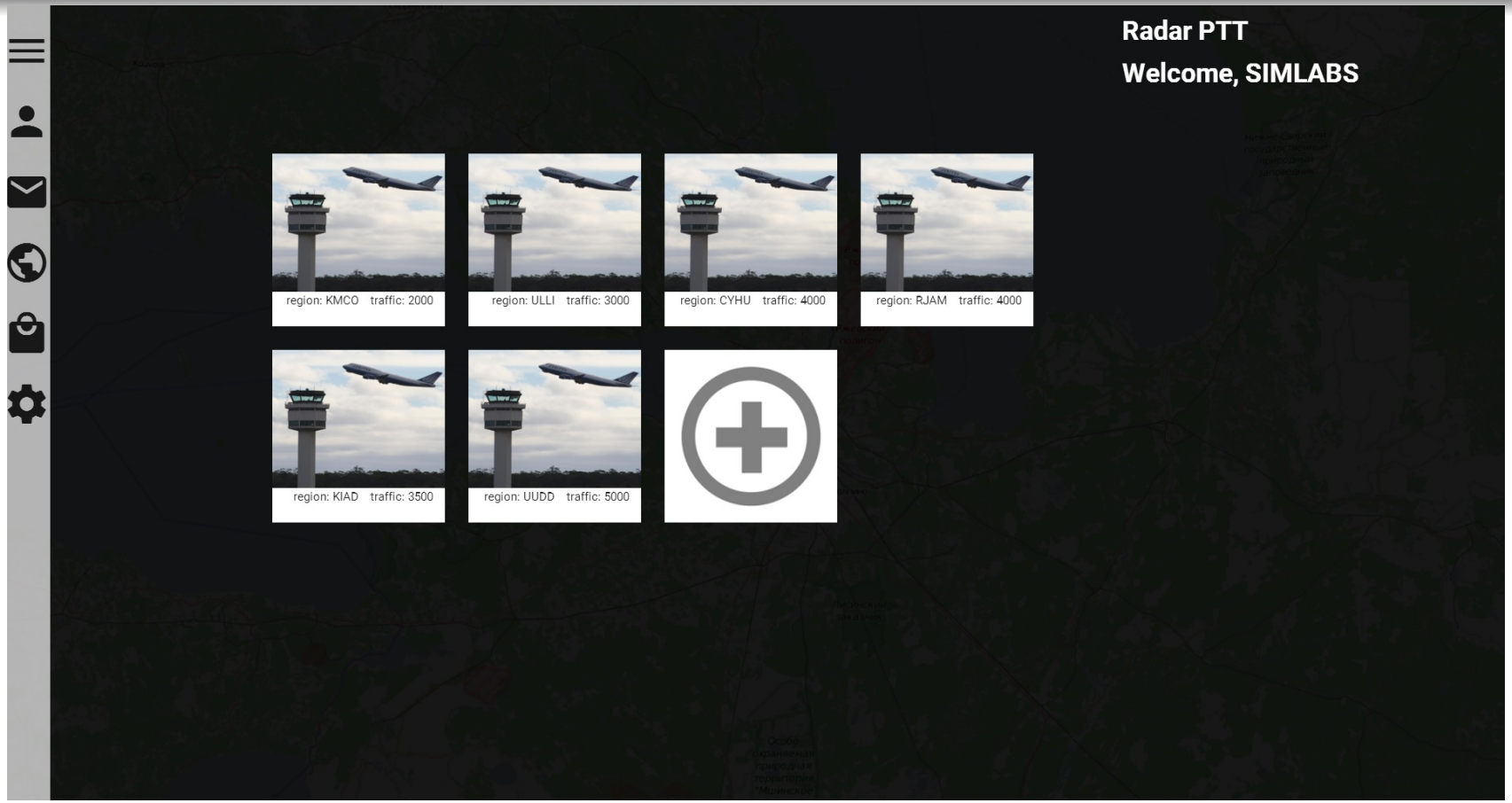


Спасибо за внимание

```
1  void web_tunnel::test_widget::initialize_context(
       context_t & context)
2  {
3    bind_method("foo", &test_widget::foo, this, context);
4    bind_function("bar", &bar, context);
5  }
6
7  void web_tunnel::test_widget::on_js_context_created()
8  {
9      js_executor_.js_foo();
10 }
```

```
1  onComponentReady: function(controller) {
2    controller.registerCallback("js_foo", this.foo, this);
3    controller.foo();
4  }
```


Login
Password
SIGN IN


Radar PTT
Welcome, SIMLABS
region: KMCO traffic: 2000
region: ULLI traffic: 3000
region: CYHU traffic: 4000
region: RJAM traffic: 4000
region: KIAD traffic: 3500
region: UUDD traffic: 5000