

C++Now 2016

# LET'S MAKE A WEB MATCH-3 GAME IN C++14

<https://github.com/modern-cpp-examples/match3>

**BEING CROSS-PLATFORM?**

# WRITE ONCE, DEPLOY EVERYWHERE!

Platform	
Desktop	Windows / OS X / Linux
Mobile	Android / iOS / Windows
Web	Chrome / IE / Safari / Firefox

## Applications

---

Mobile	QT/NDK+djinni
Desktop	QT/wxWidgets
Web	Emscripten/Cheerp/WebAssembly

## Games

Mobile	NDK+OpenGL+SDL/Cocos2d-x/Marmalade
Desktop	Unreal Engine/Cocos2d-x
Web	Emscripten/Cheerp/WebAssembly

# COMBINE C++ WITH JS

- Cordova
- Titanium
- ReactNative
- QML

# C++ AND THE WEB



EMSCRIPTEN IS AN LLVM-BASED PROJECT  
THAT COMPILES C AND C++ INTO HIGHLY-  
OPTIMIZABLE JAVASCRIPT IN ASM.JS FORMAT



```
#include <iostream>
int main() {
    std::cout << "hello world!" << std::endl;
}
```

```
em++ hello_world.cpp -o index.html
```

```
$browser index.html
```



## CHEERP IS THE C++ COMPILER FOR THE WEB

write a web application, or port your existing one, all in C++.  
cheerp will generate JavaScript code that can run on any  
browser

# WEBASSEMBLY

WEBASSEMBLY OR WASM IS A NEW PORTABLE, SIZE- AND  
LOAD-TIME-EFFICIENT FORMAT SUITABLE FOR  
COMPILATION TO THE WEB

---

In Progress!

# LET'S MATCH SOME CANDIES / JEWELS



# CORE MECHANICS

1. Swipe items (2 given items)
2. For given items
  - 2.1 Find matches
  - 2.2 Remove matches
  - 2.3 Scroll down items (which were above removed)
  - 2.4 Generate new items
  - 2.6 Find affected items (new or scrolled items)
  - 2.7 For affected (new given items) items go to 2

## MATCH 3

---



## MATCH 3 - TWICE

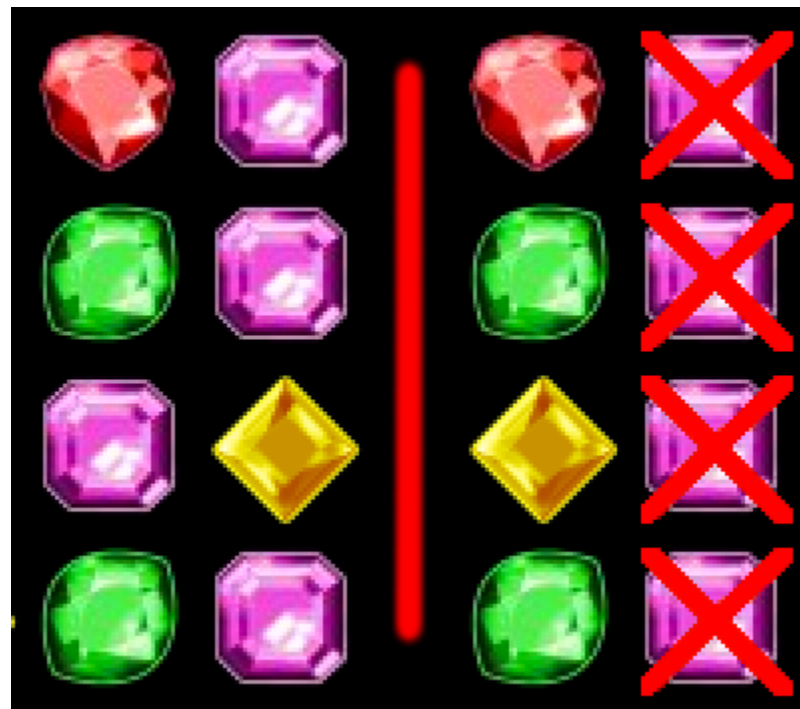
---





## MATCH 4

---



## MATCH 5

---



## MATCH L

---



## MATCH CAN GENERATE MORE MATCHES

---



# REQUIREMENTS

## C++14 / STL

- Clang-3.7+
- GCC-6+
- Emscripten-1.35

# DEPENDENCIES

# SDL2

---

<https://www.libsdl.org>

---

- SDL2\_image
- SDL2\_ttf



# EXPERIMENTAL BOOST.DI

---

<https://github.com/boost-experimental/di>

```
#include <boost/di.hpp>

namespace di = boost::di;

struct renderer { int device; };
struct view { view(std::string title, const renderer&) {} };
class model {};
struct controller { controller(model&, view&) {} };
class user {};
struct app { app(controller&, const user&) {} };

int main() {
    auto injector = di::make_injector();
    injector.create<app>();
}
```

# EXPERIMENTAL BOOST.MSM-LITE

---

<https://github.com/boost-experimental/msm-lite>

```
#include <boost/msm-lite.hpp>

namespace msm = boost::msm::lite;

auto guard = [] { return true; };
auto action = [] { std::cout << "action" << std::endl; };

struct hello_world {
    auto configure() const noexcept {
        using namespace msm;
        return make_transition_table(
            * "idle"_s + event1 = "s1"_s
            , "s1"_s + event2 [ guard ] / action = "s2"_s
        );
    }
};
```

```
int main() {
    msm::sm<hello_world> sm;
    using namespace msm;
    sm.process_event(event1{});
    sm.process_event(event2{});
}
```

# RANGE-V3

---

<https://github.com/ericniebler/range-v3>

```
#include <range/v3/all.hpp>

int main() {
    using namespace ranges::view;

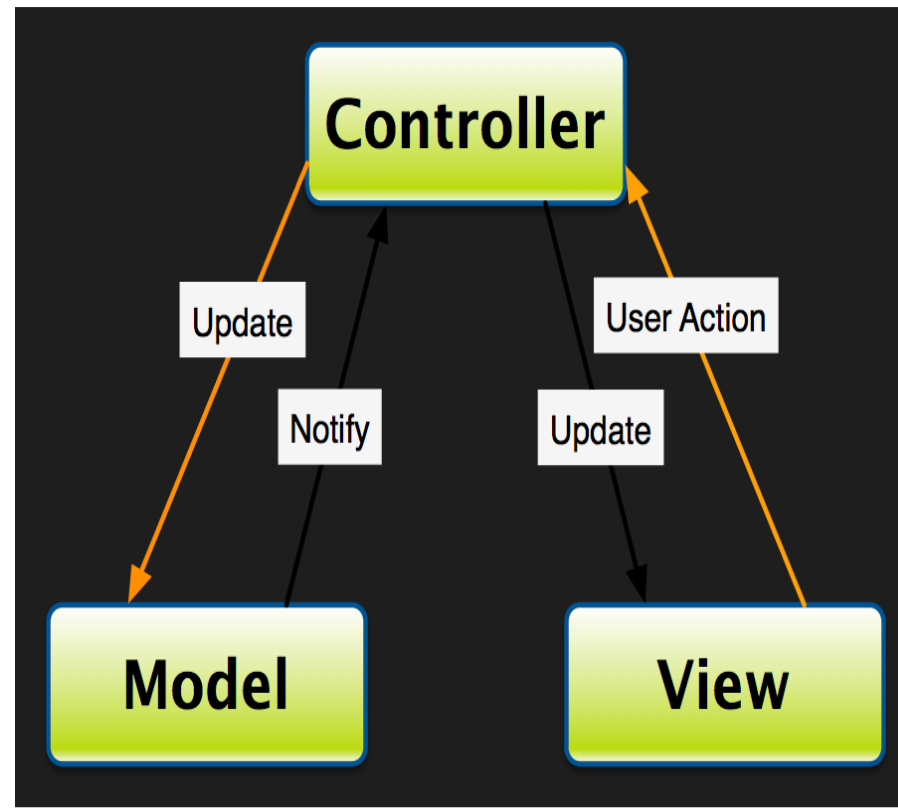
    ranges::recursive_range_fn<int> const fibs { [&]{
        return concat(
            closed_ints(0,1)
            , zip_with(std::plus<int>{}, fibs(), tail(fibs()))
        );
    }};

    auto x = take(fibs(), 20);
    ranges::for_each(x, [](int i) { std::cout << i << std::endl; });
}
```

**DESIGN**

# MODEL-VIEW-CONTROLLER

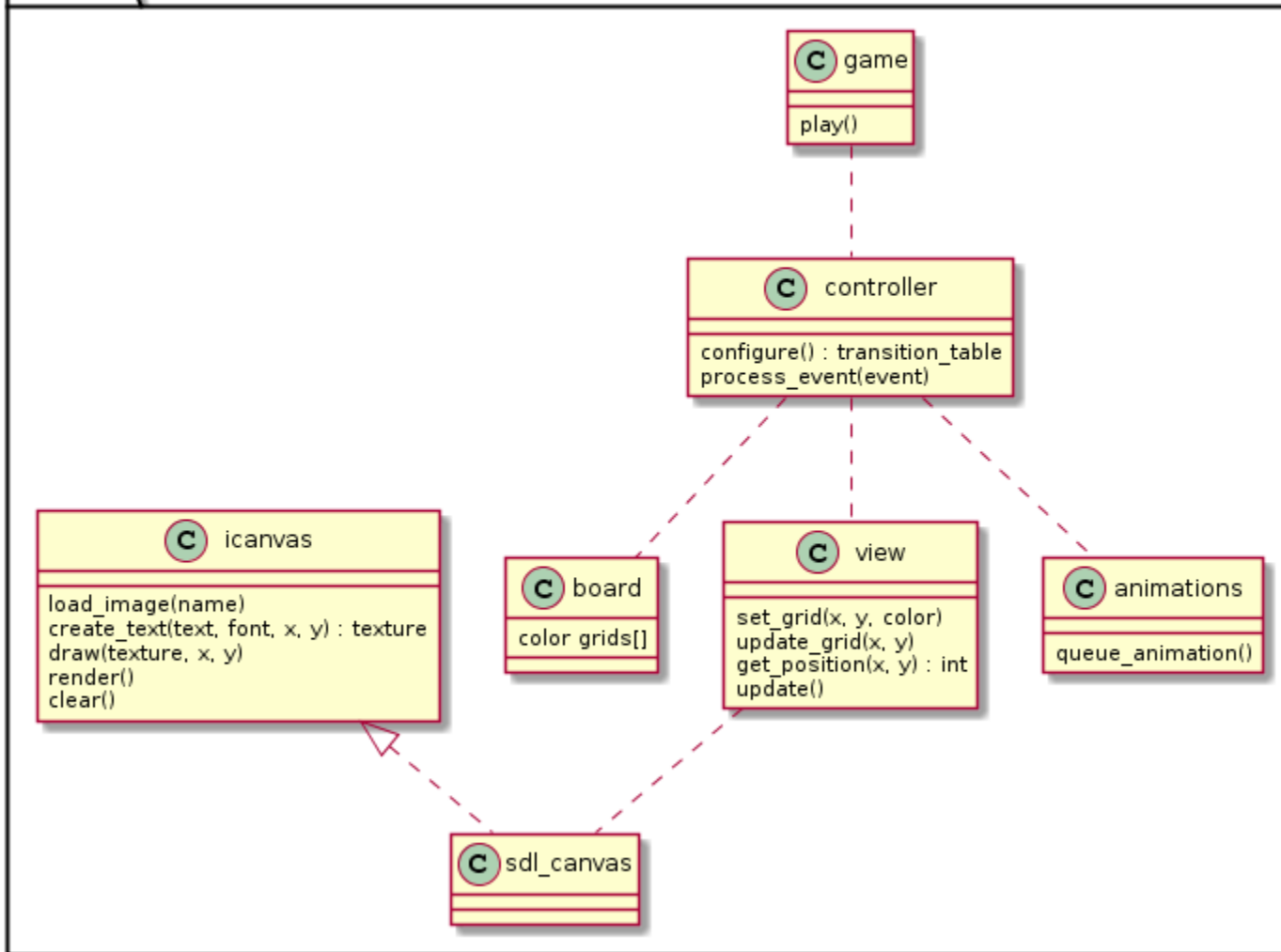




- Separates business logic from UI
- Good for applications and/or games
- Compatible with Entity-Component-System

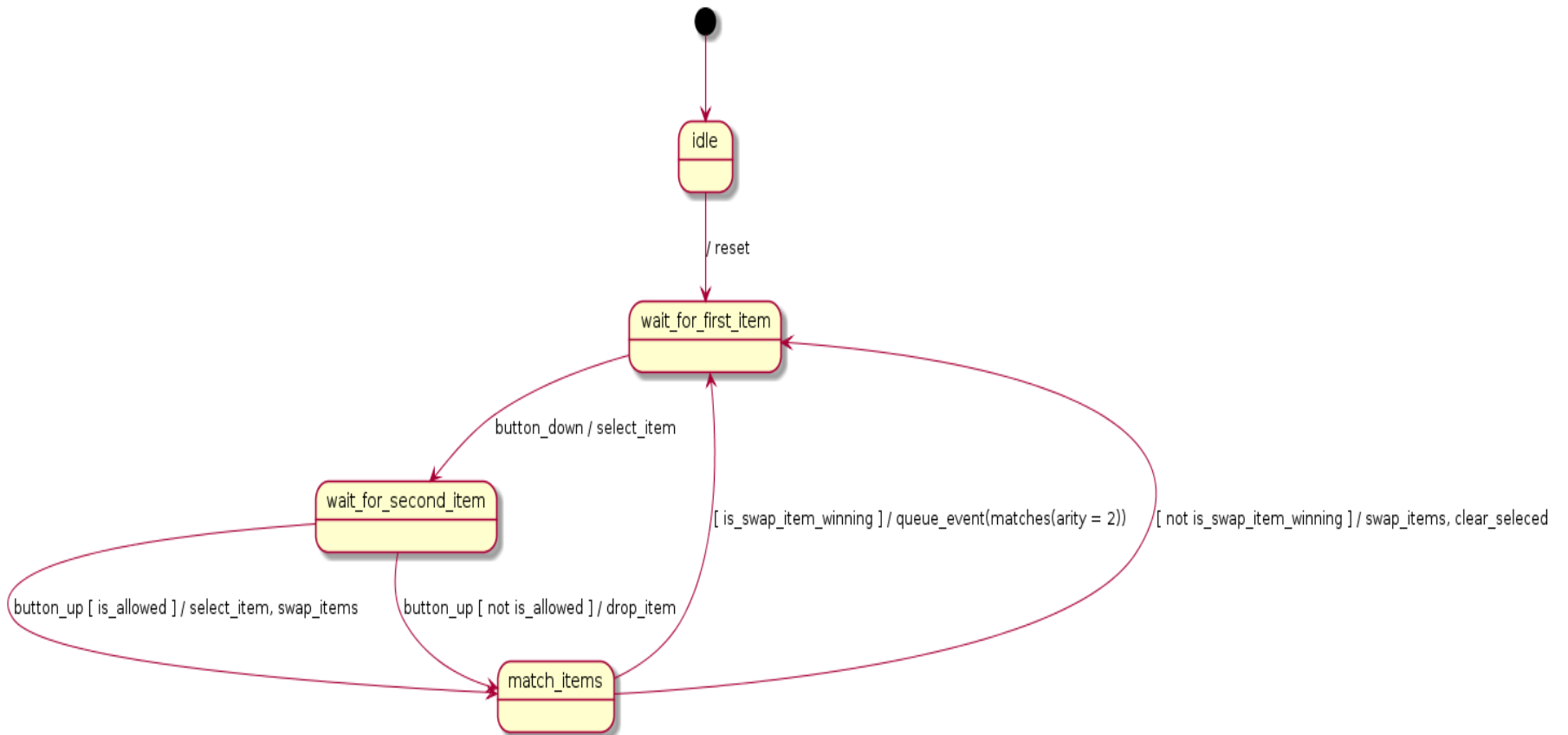
# CLASS DIAGRAM

match3

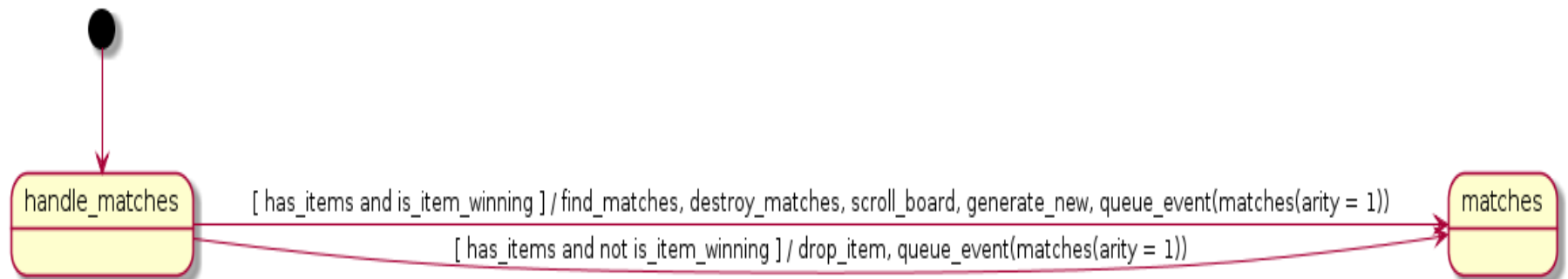


# STATE DIAGRAM

## Swap



## Match



**COMPILE & RUN**



**DESKTOP**

## COMPILE

```
clang++ -std=c++14 -lSDL2 -lSDL2_image -lSDL2_ttf  
-o match3 src/main.cpp
```

## RUN

```
./match3
```

# CMAKE

```
mkdir build  
cmake ..  
make app
```

**WEB**

**EMSCRIPTEN**

Linux / Mac OSX

<https://s3.amazonaws.com/mozilla-games/emscripten/releases/emsdk-portable.tar.gz>

---

Windows (\*Installer available)

<https://s3.amazonaws.com/mozilla-games/emscripten/releases/emsdk-1.35.0-portable-64bit.zip>

```
# Fetch the latest registry of available tools.  
./emsdk update
```

```
# Download and install the latest SDK tools.  
./emsdk install latest
```

```
# Make the "latest" SDK "active"  
./emsdk activate latest
```

```
# Set the current Emscripten path on Linux/Mac OSX  
source ./emsdk_env.sh
```

```
./emsdk list
```



The following precompiled tool packages are available for download:

	clang-e1.30.0-64bit	
	clang-e1.34.1-64bit	
(*)	clang-e1.35.0-64bit	INSTALLED
(*)	node-4.1.1-64bit	INSTALLED
	spidermonkey-37.0.1-64bit	
	spidermonkey-nightly-2015-04-12-64bit	
	emscripten-1.30.0	
	emscripten-1.34.1	
(*)	emscripten-1.35.0	INSTALLED
	crunch-1.04	

The following tools can be compiled from source:

clang-tag-e1.36.2-32bit  
clang-tag-e1.36.3-32bit  
clang-tag-e1.36.2-64bit  
clang-tag-e1.36.3-64bit  
clang-incoming-32bit  
clang-incoming-64bit  
clang-master-32bit  
clang-master-64bit  
emscripten-tag-1.36.2-32bit  
emscripten-tag-1.36.3-32bit  
emscripten-tag-1.36.2-64bit  
emscripten-tag-1.36.3-64bit  
emscripten-incoming-32bit  
emscripten-master-32bit  
emscripten-incoming-64bit  
emscripten-master-64bit

The following precompiled SDKs are available for download:

	sdk-1.30.0-64bit	
	sdk-1.34.1-64bit	
*	sdk-1.35.0-64bit	INSTALLED

The following SDKs can be compiled from source:

sdk-incoming-64bit  
sdk-master-64bit

## COMPILE

```
em++ -std=c++14 -s USE_SDL=2 -s USE_SDL_IMAGE=2 -s USE_SDL_TTF=2  
--emrun # log to the console  
--preload-file data # images and fonts  
--use-preload-plugins # load data  
--shell-file data/template.html # custom html template  
-o index.html src/main.cpp
```

## RUN

```
$browser index.html
```

## OR

```
emrun --port 8080 index.html # start http server  
$browser localhost:8080
```

# CMAKE

```
mkdir build  
CXX=em++ cmake ..  
make web emrun
```

## COMPILATION TIME BENCHMARK

---

Compiler (-O2)	Note	Time
Clang-3.8	Without Range-V3	2.122s
Clang-3.8	With Range-V3	9.521s
Emscripten-1.35	Without Range-v3	2.822s
Emscripten-1.35	With Range-V3	9.912s

# IMPLEMENTATION

# MATCH-3 GAME

---

<https://github.com/modern-cpp-examples/match3>

## FILES...

```
▼ src/  
  ▼ controller/  
    controller.hpp  
    logic.hpp  
  ▼ model/  
    board.hpp  
    config.hpp  
  ▼ view/  
    animations.hpp  
    icanvas.hpp  
    sdl_canvas.hpp  
    view.hpp  
  pph.hpp  
  game.hpp  
  main.cpp
```

**MAIN**



## CREATE AND PLAY THE GAME

```
int main() {  
    auto injector = di::make_injector(configuration());  
    injector.create<match3::game>().play();  
}
```

**DI CONFIGURATION**

# EXPERIMENTAL BOOST.DI

---

<https://github.com/boost-experimental/di>

## BIND ICANVAS TO SDL\_CANVAS

```
di::bind<icanvas>.to<SDL_canvas>()
```

## SET-UP CONFIGURATION DETAILS

```
di::bind<>.to(  
  config{.win_title = "match3",  
    .win_width = 320,  
    .win_height = 480,  
    .board_width = 7,  
    .board_height = 10,  
    .board_colors = 5,  
    .max_moves = 10}  
)
```

## SET-UP BOARD

```
di::bind<board::color[]>.to({  
    3, 5, 1, 4, 3, 2, 2,  
    1, 1, 4, 2, 5, 1, 3,  
    5, 3, 5, 4, 5, 3, 2,  
    4, 4, 2, 1, 3, 4, 5,  
    5, 1, 1, 2, 4, 5, 1,  
    5, 2, 3, 5, 4, 2, 1,  
    1, 5, 5, 1, 5, 5, 4,  
    2, 3, 3, 1, 3, 3, 4,  
    3, 2, 2, 5, 4, 4, 1,  
    1, 2, 3, 4, 1, 3, 4  
})
```

## BIND RANDOMIZER

```
di::bind<randomize>.to([](int begin, int end) {  
    static std::random_device rd;  
    std::mt19937 gen(rd());  
    std::uniform_int_distribution<int> dis(begin, end);  
    return dis(gen);  
})
```

# GAME LOOP



# PLAY

```
class game {
public:
    explicit game(msm::sm<controller>& c) : controller_(c) {}

    void play() {
        EM(emsripten_set_main_loop_arg(
            play_impl, reinterpret_cast<void*>(&controller_), 0, 0))
        (play_impl(reinterpret_cast<void*>(&controller_)));
    }
}
```

## HANDLE EVENTS

```
static void play_impl(void* c) {
    auto& controller_ =
        *reinterpret_cast<msm::sm<controller>*>(c);

    do {
        auto dispatch_event = msm::make_dispatch_table<
            SDL_Event, SDL_QUIT, SDL_FINGERMOTION>(controller_);
        controller_.process_event(time_tick{});
        SDL_Event event;
        while (SDL_PollEvent(&event)) {
            dispatch_event(event, event.type);
        }
    } while (EM(false && )() !controller_.is(msm::X));
}
```

**MODEL**

## BOARD

```
struct board {  
    using color = int;  
    std::vector<color> grids;  
};
```

**VIEW**

# SDL2

---

<https://www.libsdl.org>

# CANVAS

```
class icanvas {
public:
    virtual ~icanvas() noexcept = default;
    virtual std::shared_ptr<void>
    load_image(const std::string&) const = 0;
    virtual std::shared_ptr<void>
    create_text(
        const std::string&, const std::string&, int) const = 0;
    virtual void draw(
        std::shared_ptr<void>, int x = 0, int y = 0, bool = true) = 0;
    virtual void render() = 0;
    virtual void clear() = 0;
};
```

## VIEW

```
class view {  
    view(icanvas&, config);  
    void set_grid(int x, int y, int c);  
    void update_grid(int x, int y);  
    auto get_position(int x, int y) const;  
    void set_text(const std::string& text, int x, int y);  
    void update() { canvas_.render(); }  
    void clear() { canvas_.clear(); }  
};
```



# ANIMATIONS

```
class animations {  
    explicit animations(view& v);  
    void queue_animation(const std::function<void()>& f  
                        , std::chrono::milliseconds length = {});  
    void update();  
};
```

**CONTROLLER**

# EXPERIMENTAL BOOST.MSM-LITE

---

<https://github.com/boost-experimental/msm-lite>

```

return make_transition_table(
// +-----+
    "wait_for_first_item"_s <= *("idle"_s) / (reset, show_board, show_points, show_moves)

    , "wait_for_click"_s <= "wait_for_first_item"_s [ not [](moves& m) { return m > 0; } ] / show_game_over
    , "wait_for_first_item"_s <= "wait_for_click"_s + event<button_up> [ not is_mobile ] / (reset, show_board, show_points, show_moves)

    , "wait_for_second_item"_s <= "wait_for_first_item"_s + event<button_down> / select_item
    , "match_items"_s <= "wait_for_second_item"_s + event<button_up> [ is_allowed ] / (select_item, swap_items, show_swap)
    , "wait_for_first_item"_s <= "wait_for_second_item"_s + event<button_up> [ not is_allowed ] / drop_item

    , "wait_for_first_item"_s <= "match_items"_s [ is_swap_items_winning ] / (
        [](moves& m) {--m;}, show_moves,
        msm::queue_event(matches{.arity = 2})
    )

    , "wait_for_first_item"_s <= "match_items"_s / (swap_items, show_swap, clear_selected)
// +-----+
    , *("handle_matches"_s) + event<matches> [ has_items and is_item_winning ] / (
        find_matches, show_matches
        , destroy_matches, show_board
        , add_points, show_points
        , scroll_board, show_board
        , generate_new, show_board
        , msm::queue_event(matches{.arity = 1})
    )

    , "handle_matches"_s + event<matches> [ has_items and not is_item_winning ] / (
        drop_item, msm::queue_event(matches{.arity = 1})
    )
// +-----+
    , X <= *("wait_for_client"_s) + event<key_pressed> [ is_key(SDLK_ESCAPE) ]
    , X <= "wait_for_client"_s + event<quit>
// +-----+
    , *("handle_animations"_s) + event<time_tick> / [](animations& a) { a.update(); }
// +-----+
);

```

**GUARDS/ACTIONS**

## IS MOVE ALLOWED

```
auto is_allowed =  
    [](auto event, const view& v, const selected& s, config c) {  
        assert(!s.empty());  
        const auto _1 = s.back();  
        const auto _2 = v.get_position(event.x, event.y);  
        const auto diff = std::abs(_1 - _2);  
        const auto board_size = c.board_width * c.board_height;  
        return (_1 >= 0 && _1 < board_size) &&  
            (_2 >= 0 && _2 < board_size) &&  
            (diff == 1 || diff == c.board_width);  
    };
```

## GENERATE NEW ITEMS

```
auto generate_new = [](board& b, const auto& m, selected& s,  
                      config c, randomize r) {  
    ranges::action::transform(b.grids,  
        [c, r](auto i) { return i ? i : r(1, c.board_colors); }  
    );  
  
    s |= ranges::action::push_front(affected(m.matches, c.board_width),  
        | ranges::action::sort | ranges::action::unique;  
};
```

## SWAP ITEMS

```
auto swap_items = [](const selected& s, board& b) {  
    assert(s.size() >= 2);  
    std::swap(b.grids[s[0]], b.grids[s[1]]);  
};
```



## MORE GUARDS/ACTIONS

---

<https://github.com/modern-cpp-examples/match3/blob/master/src/controller/controller.hpp>

**LOGIC**

# RANGE-V3

---

<https://github.com/ericniebler/range-v3>

**ROW VIEW**

## IDEA

```
| 1 2 3 |  
| 3 4 5 | => n:1 -> | 1 2 3 |  
| 6 7 8 |           | [3] [4] [5] |  
| 6 7 8 |           | 6 7 8 |
```

## ROW

```
auto row = [](View&& v, Number n, Number width) {  
    return v | ranges::view::drop(width * n) |  
              ranges::view::take(width); // or slice  
};
```

# COLUMN VIEW

## IDEA

```
| 1 2 3 |  
| 3 4 5 | => n:1 -> | 1 [2] 3 |  
| 6 7 8 |           | 3 [4] 5 |  
                   | 6 [7] 8 |
```

## COLUMN

```
auto col = [](View&& v, Number n, Number width) {  
    return v | ranges::view::drop(n) |  
              ranges::view::stride(int(width));  
};
```

**MATCH N**



## IDEA

```
| 1 3 3 3 2 1 | => color:3, n:3 -> {begin: 1, length: 3}  
| 1 2 3 3 3 3 | => color:3, n:3 -> {begin: 2, length: 4}
```

## MATCH 3

```
auto match_n = [] (View&& v, Color color, Number n = 3) {  
    const auto&& matches =  
        ranges::view::ints |  
        ranges::view::take(ranges::size(v) - n + 1) |  
        ranges::view::transform([=] (auto i) {  
            return ranges::count(v | ranges::view::drop(i) |  
                                ranges::view::take(n), color) == n;  
        });  
};
```

## MATCH 3

```
constexpr auto is_match = true;
const auto it = ranges::find(matches, is_match);
const auto found = it != ranges::end(matches);
const auto mlength =
    found ? ranges::count(matches, is_match) + (n - 1) : 0;
const auto mbegin =
    found ? ranges::distance(ranges::begin(matches), it) : 0;

struct { decltype(mbegin) begin; decltype(mlength) length; }
    result{mbegin, mlength};

return result;
};
```

**IS MATCH**

## IDEA - MATCH FOUND

2 3 4 5		2 3 4 5		2 3 4 5	
7 7 9 3		7 7 9 3		7 7 9 3	
3 4 5 3	=> value:10 ->	3 4 [5] 3	=>	3 4 [5] 3	=> true
1 3 5 1		1 3 5 1		1 3 [5] 1	
2 1 5 8		2 1 5 8		2 1 [5] 8	

## IDEA - NO MATCH

2 3 4 5		2 3 4 5		2 3 4 5	
7 7 9 3		7 7 9 3		7 7 9 3	
3 4 5 3	=> value:11 ->	3 4 5 [3]	=>	3 4 5 3	=> false
1 3 5 1		1 3 5 1		1 3 5 1	
2 1 5 8		2 1 5 8		2 1 5 8	

## IS MATCH

```
auto is_match = [](View&& v, Number value, Number width) {  
    const auto color = v[value];  
    const auto x = value % width;  
    const auto y = value / width;  
    return match_n(row(v, y, width), color).length ||  
           match_n(col(v, x, width), color).length;  
};
```

**MATCH**

## IDEA

```
| 1 2 3 4 5 |
| 6 7 7 9 3 |
| 2 5 5 5 3 | => value:13 ->
| 2 1 3 5 1 |
| 4 2 1 5 8 |

| 1 2 3 4 5 |
| 6 7 7 9 3 |
| 2 5 5 [5] 3 | =>
| 2 1 3 5 3 |
| 4 2 1 5 8 |

=>
| 1 2 3 4 5 |
| 6 7 7 9 3 |
| 2 [5] [5] [5] 3 | => [11, 12, 13, 18, 23]
| 2 1 3 [5] 3 |
| 4 2 1 [5] 8 |
```



## MATCH 1/2

```
auto match = [](View&& v, Number value, Number width) {  
    const auto color = v[value];  
    const auto x = value % width;  
    const auto y = value / width;  
    const auto match_r = match_n(row(v, y, width), color);  
    const auto match_c = match_n(col(v, x, width), color);  
    const auto transform = [](auto length, auto expr) {  
        return ranges::view::ints | ranges::view::take(length) |  
            ranges::view::transform(expr);  
    };  
};
```

## MATCH 2/2

```
std::vector<decltype(value)> result = ranges::view::concat(
    transform(match_r.length,
        [=](auto i) { return y * width + match_r.begin + i; }),
    transform(match_c.length, [=](auto i) {
        return (match_c.begin + i) * width + x;
    }));
result |= ranges::action::sort | ranges::action::unique;
return result;
};
```

**SCROLL**

## IDEA

3		3		0
0		[0]		0
0	=> value:1 ->	0	=>	0
0		0		3
4		4		4

## SCROLL

```
auto scroll = [] (View&& v, const Container& value, Number width) {  
    const auto&& c = col(v, value % width, width) |  
        ranges::view::take(value / width + 1);  
    auto begin = ranges::begin(c);  
    ranges::advance(begin, value / width);  
    ranges::rotate(c, begin);  
};  
,
```

**AFFECTED ITEMS**

## IDEA

```
[11, 12, 13, 18, 23] => | 1  2  3  4  5 |
                        | 6  7  7  9  3 |
                        | 2 [5] [5] [5] 3 | =>
                        | 2  1  3  [5] 3 |
                        | 4  2  1  [5] 8 |

=> | 1  [2] [3] [4] 5 |
   | 6  [7] [7] [9] 3 |
   | 2  [5] [5] [5] 3 | => [1, 2, 3, 6, 7, 8, 11, 12, 13, 18, 23]
   | 2  1  3  [5] 3 |
   | 4  2  1  [5] 8 |
```

## AFFECTED

```
auto affected = [](const Container& matches, Number width) {
    const auto&& columns =
        matches | ranges::view::transform([=](auto m) {
            return ranges::view::ints |
                ranges::view::take(m / width + 1) |
                ranges::view::transform([=](auto i) {
                    return m % width + (i * width);
                });
        });

    std::decay_t<decltype(matches)> result =
        columns | ranges::view::join;

    result |= ranges::action::sort | ranges::action::unique;
    return result;
};
```



# WARNING

IT'S NOT 100% C++14!

- `"idle"_s` - GNU extension / string-literal-operator-template
  - Standard replacement: `state<class idle>`
- `matches{.arity = 2}` - C99 / designated-initializer
  - Standard replacement: `matches{2}`

**TESTS**

# UNIT TESTS

**LOGIC**

```
"scroll"_test = [] {  
  int v[] = {1, 1, 3, 4, 0, 2, 7, 2, 3};  
  scroll(v, 4, 3);  
  expect(ranges::equal({1, 0, 3, 4, 1, 2, 7, 2, 3}, v));  
};
```

**GUARDS**

```
"is key"_test = [] {  
    constexpr auto key = 42;  
    struct {  
        int key;  
    } event{key};  
    expect(is_key(key) (event));  
};
```

**ACTIONS**



```
"swap_items"_test = [] {  
  board b;  
  b.grid<int>s = {1, 2};  
  selected s = {0, 1};  
  swap_items(s, b);  
  expect(ranges::equal({2, 1}, b.grid<int>s));  
};
```

# FUNCTIONAL TESTS

# FAKEIT - MOCKING FRAMEWORK

---

<https://github.com/eranpeer/Fakelt.git>

**HELPERS**

## FAKE SWIPE

```
template <class SM>
void
swipe(SM& sm, std::pair<int, int> from, std::pair<int, int> to) {
    sm.process_event(
        make_click_event<match3::button_down>(from.first, from.second));
    sm.process_event(
        make_click_event<match3::button_up>(to.first, to.second));
};
```

**MOCKS PROVIDER**

# AUTOMATICALLY CREATES MOCKS USING FAKEIT FOR ABSTRACT TYPES

---

## DI - MOCKS PROVIDER

```
auto injector = di::make_injector<mocks_provider>(...);
```

**GIVEN**



## CONFIGURATION

```
di::bind<>.to(match3::config{"", 0, 0, 7, 10, 5, 10})
```

## BOARD

```
di::bind<match3::board::color[]>.to({  
    /*0 1 2 3 4 5 6*/  
    /*0*/ 3,5,1,4,3,2,2,  
    /*1*/ 1,1,4,2,5,1,3,  
    /*2*/ 5,3,5,4,5,3,2,  
    /*3*/ 4,4,2,1,3,4,5,  
    /*4*/ 5,1,1,2,4,5,1,  
    /*5*/ 5,2,3,5,4,2,1,  
    /*6*/ 1,5,5,1,5,5,4,  
    /*7*/ 2,3,3,1,3,3,4,  
    /*8*/ 3,2,2,5,4,4,1,  
    /*9*/ 1,2,3,4,1,3,4  
})
```

## FAKE RANDOMIZER

```
di::bind<match3::randomize>.to([](int, int) {  
    static auto i = 42; return i++;  
})
```

## CANVAS MOCK

```
using namespace fakeit;
auto&& canvas = mocks_provider::get_mock<match3::icanvas>();
When(Method(canvas, load_image)).AlwaysReturn(shared_ptr<void>{});
When(Method(canvas, create_text)).AlwaysReturn(shared_ptr<void>{});
When(Method(canvas, draw)).AlwaysDo([](...){});
When(Method(canvas, render)).AlwaysDo([]{});
When(Method(canvas, clear)).AlwaysDo([]{});
```

## CONTROLLER

```
auto sm = injector.create<msm::sm<match3::controller>>();
```

**WHEN**

## TRIGGER SWIPE

```
swipe(sm, {3, 5}, {3, 6});
```

**THEN**



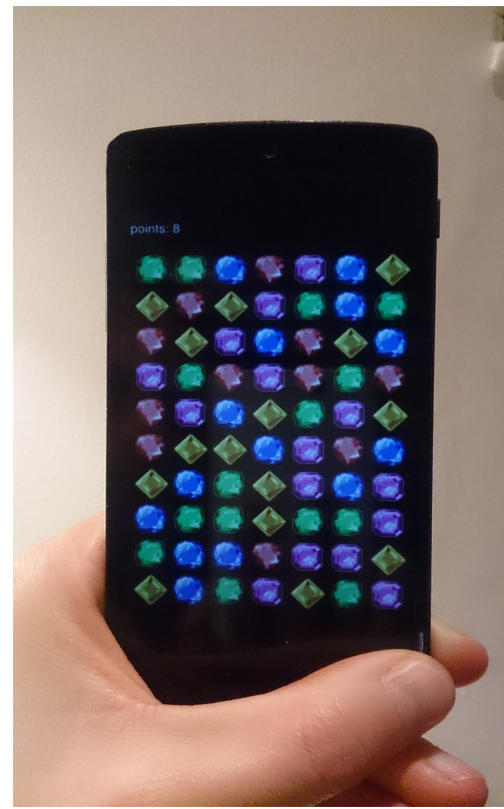
## BOARD SHOULD CHANGE

```
expect (ranges::equal ({/*0 1 2 3 4 5 6*/  
    /*0*/ 3, 42, 43, 44, 45, 46, 2,  
    /*1*/ 1, 5, 1, 4, 3, 2, 3,  
    /*2*/ 5, 1, 4, 2, 5, 1, 2,  
    /*3*/ 4, 3, 5, 4, 5, 3, 5,  
    /*4*/ 5, 4, 2, 1, 3, 4, 1,  
    /*5*/ 5, 1, 1, 2, 4, 5, 1,  
    /*6*/ 1, 2, 3, 1, 4, 2, 4,  
    /*7*/ 2, 3, 3, 1, 3, 3, 4,  
    /*8*/ 3, 2, 2, 5, 4, 4, 1,  
    /*9*/ 1, 2, 3, 4, 1, 3, 4},  
    injector.create<match3::board>().grids));  
};
```

LET'S TRY SOMETHING CRAZY?



**C++ -> JS -> MOBILE -> BROWSER**



# EMBEDDING JS IN C++

## IS MOBILE DEVICE

```
auto is_mobile = [] {  
    return bool(EM_ASM_INT_V({  
        return /iPhone|iPad|iPod|Android/i.test(navigator.userAgent);  
    }));  
};
```

**FINGER TOUCH INSTEAD OF BUTTON CLICK**

## TRANSITION TABLE

```
"second_item"_s <= "first_item"_s + touch_down [is_mobile] ...  
"match_items"_s <= "second_item"_s + touch_up   [is_mobile] ...  
  
"second_item"_s <= "first_item"_s + button_down [!is_mobile] ...  
"match_items"_s <= "second_item"_s + button_up   [!is_mobile] ...
```

# PROBLEMS / LIMITATIONS



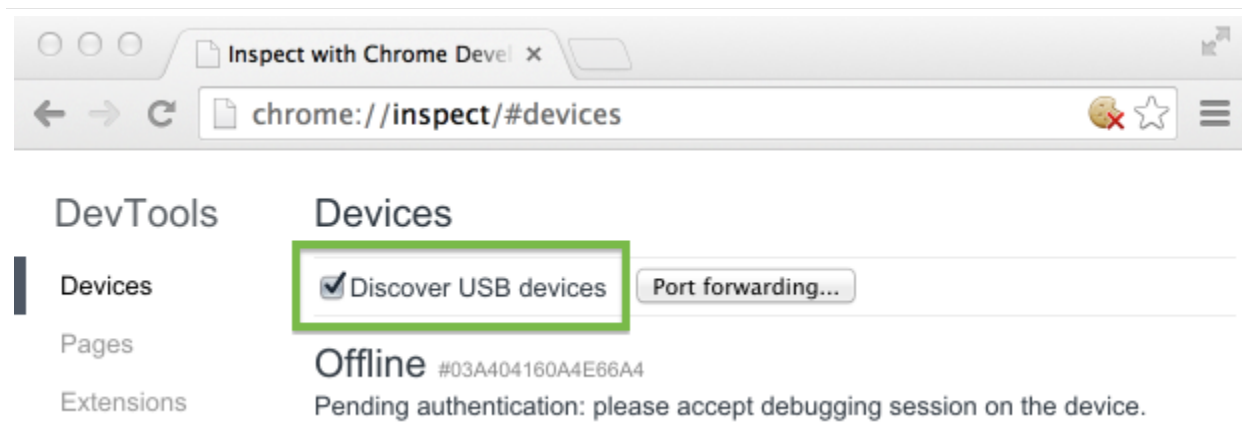
# IOS

## SECURITY REASONS

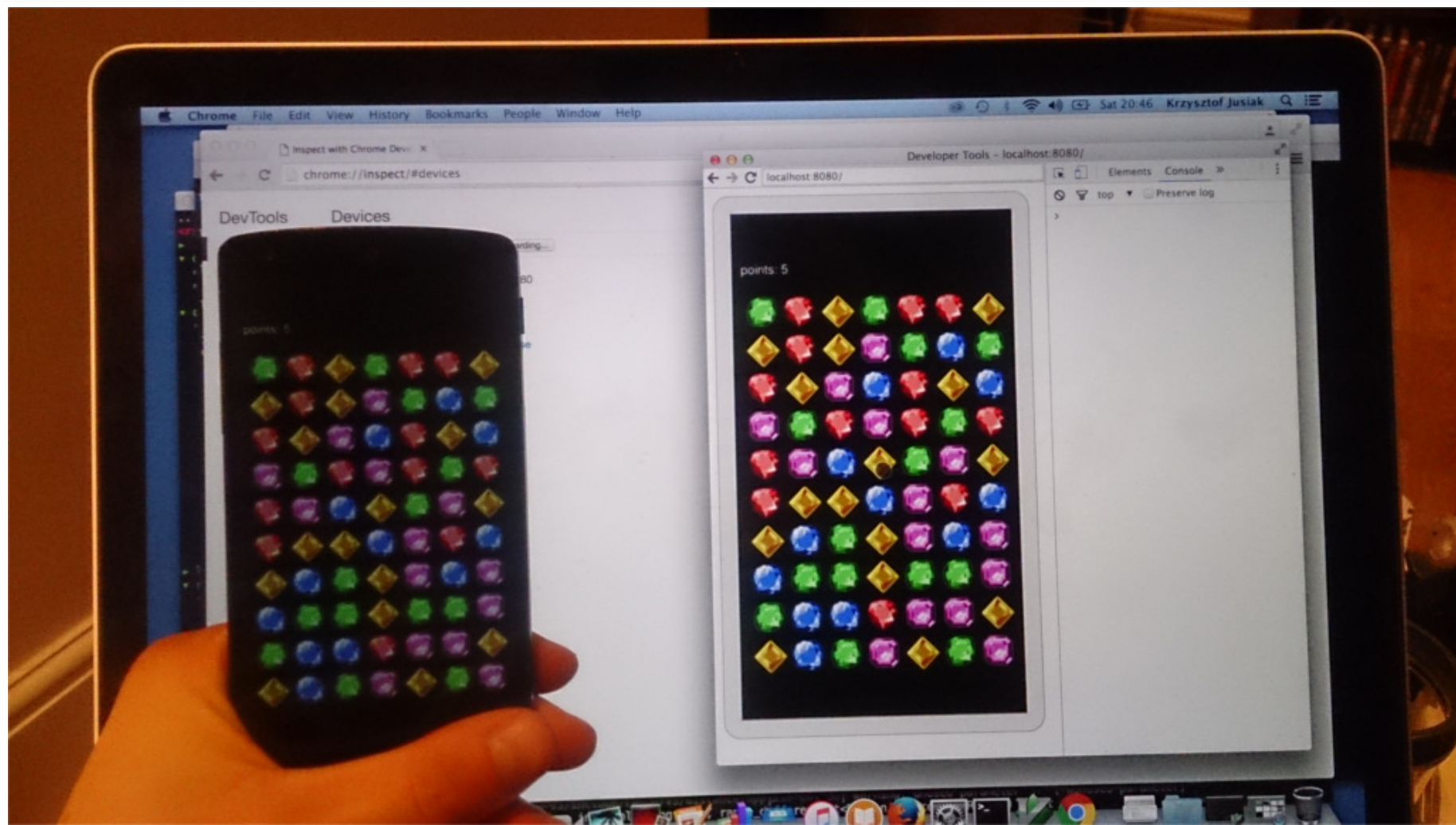
- JS is not compiled causing a slow execution
- Full screen is not allowed

**DEBUGGING C++ -> JS -> MOBILE -> CHROME**

**ANDROID / CHROME**



```
chrome://inspect/#devices
```



**IOS / SAFARI**

# SAFARI WEB INSPECTOR

# BEAT THE RECORD / PLAY





# QUESTIONS?

---

Match-3 Game

<https://github.com/modern-cpp-examples/match3>

---

Experimental  
Boost.DI

<https://github.com/boost-experimental/di>

---

Experimental  
Boost.MSM-lite

<https://github.com/boost-experimental/msm-lite>

---

Range-V3

<https://github.com/ericniebler/range-v3>

---

Fakelt

<https://github.com/eranpeer/Fakelt.git>

---



**Thank you**