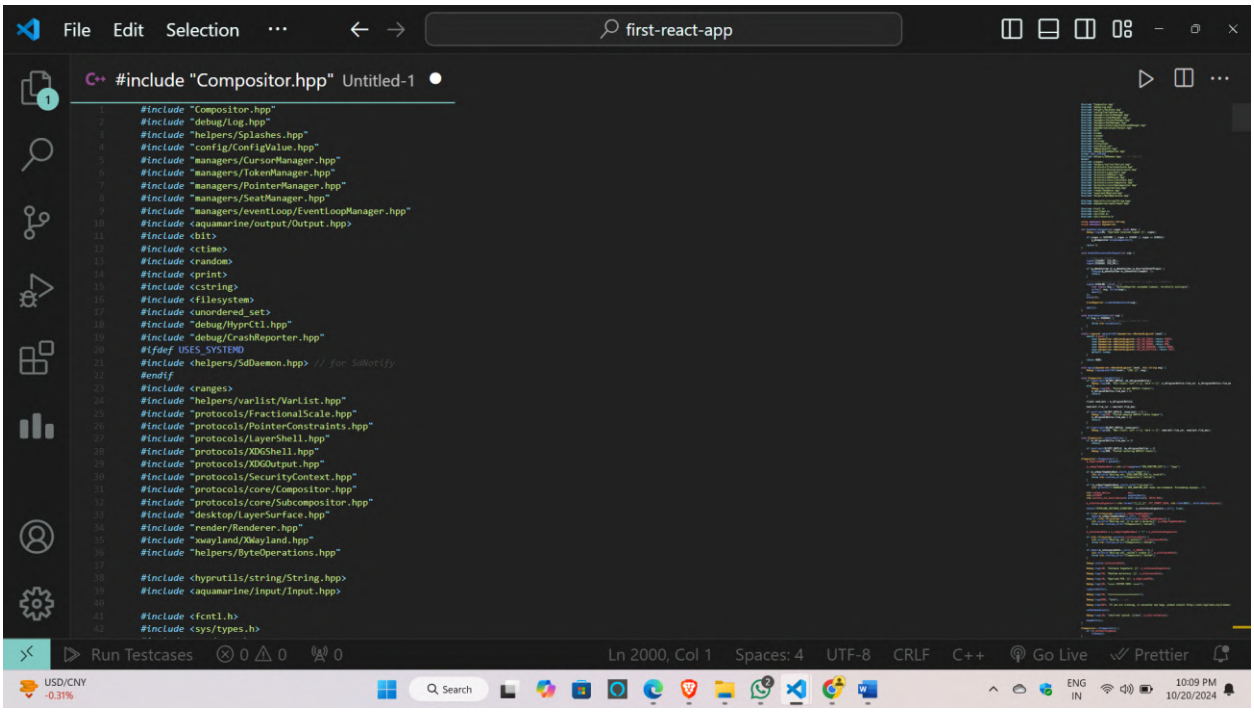
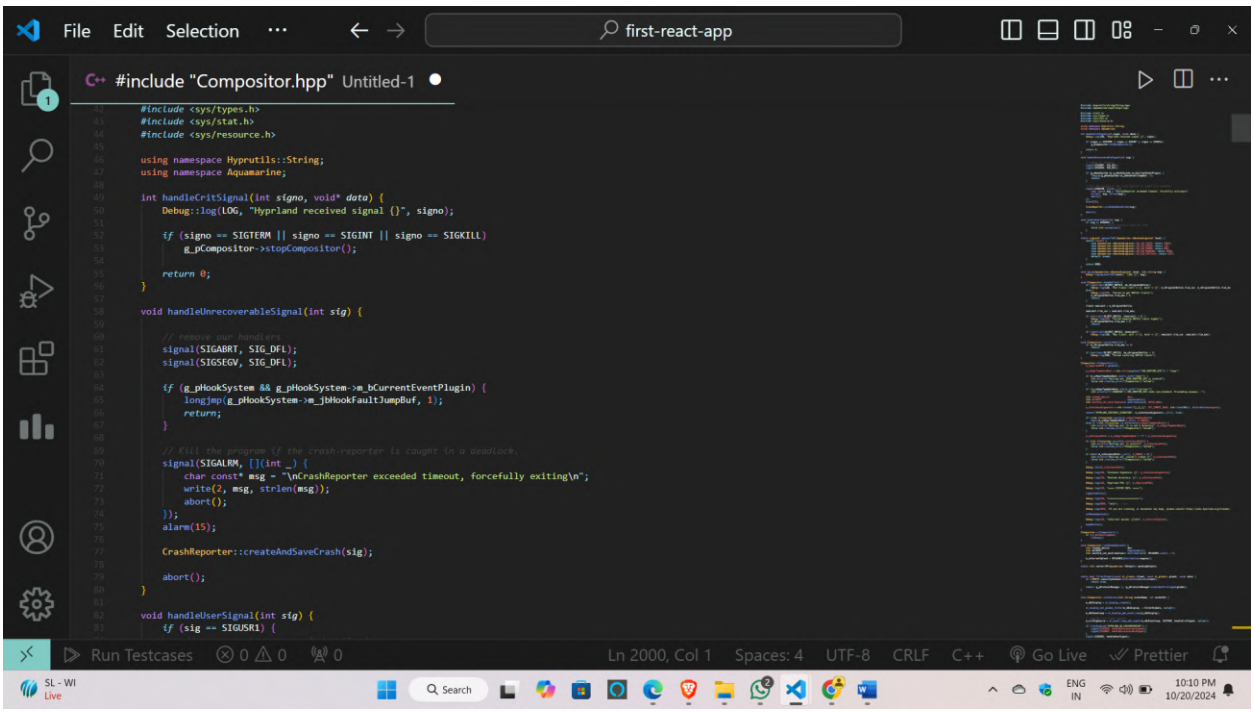
LAB – 7 _ IT314

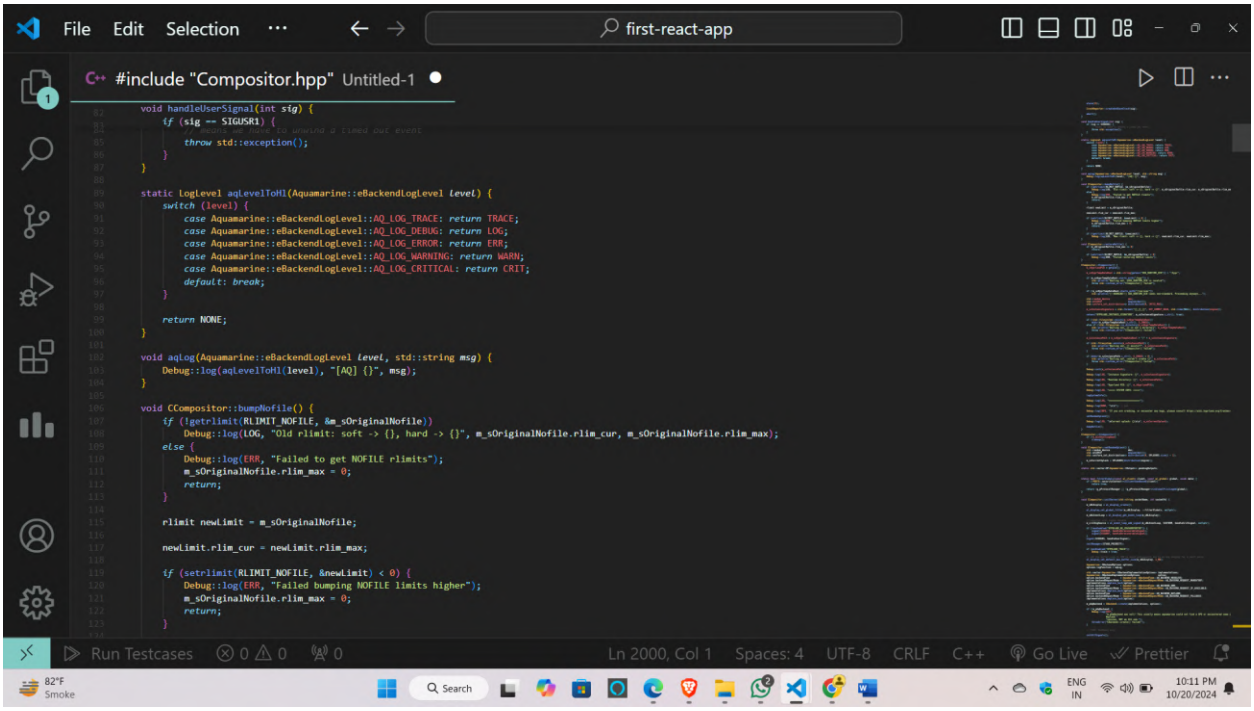
ID : 202201482

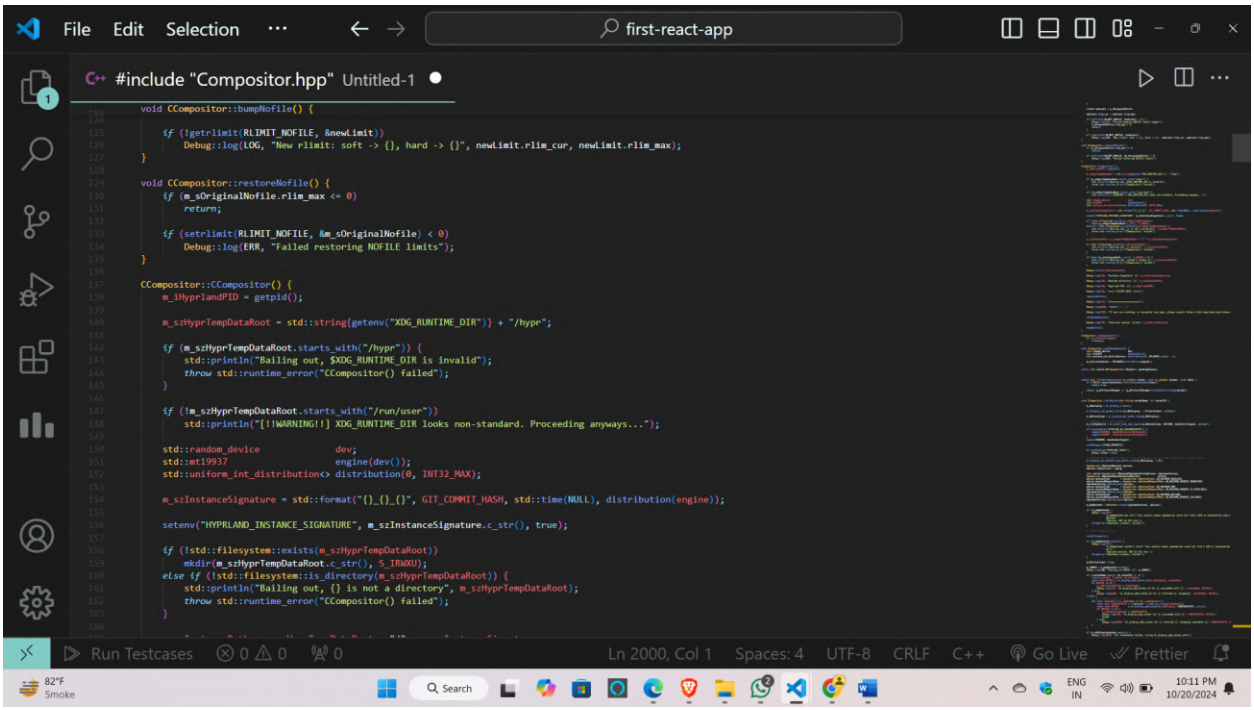
NAME : Zalavadiya Meet

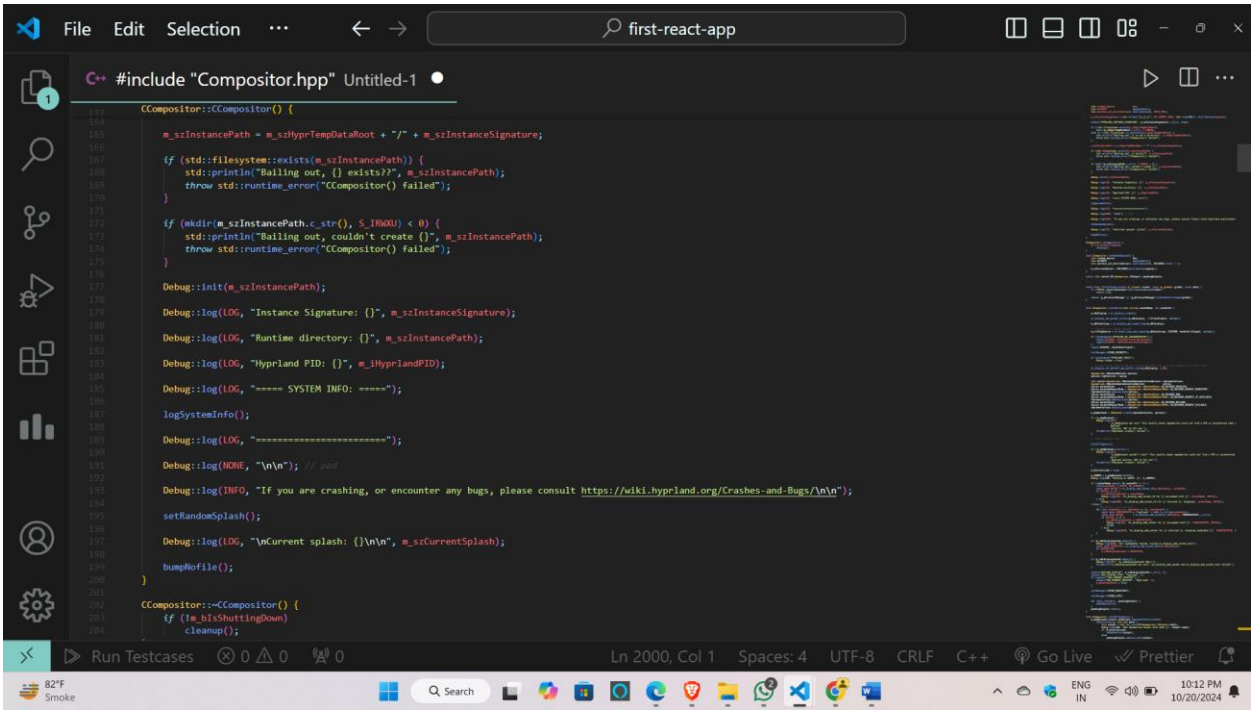
SECTION _ I

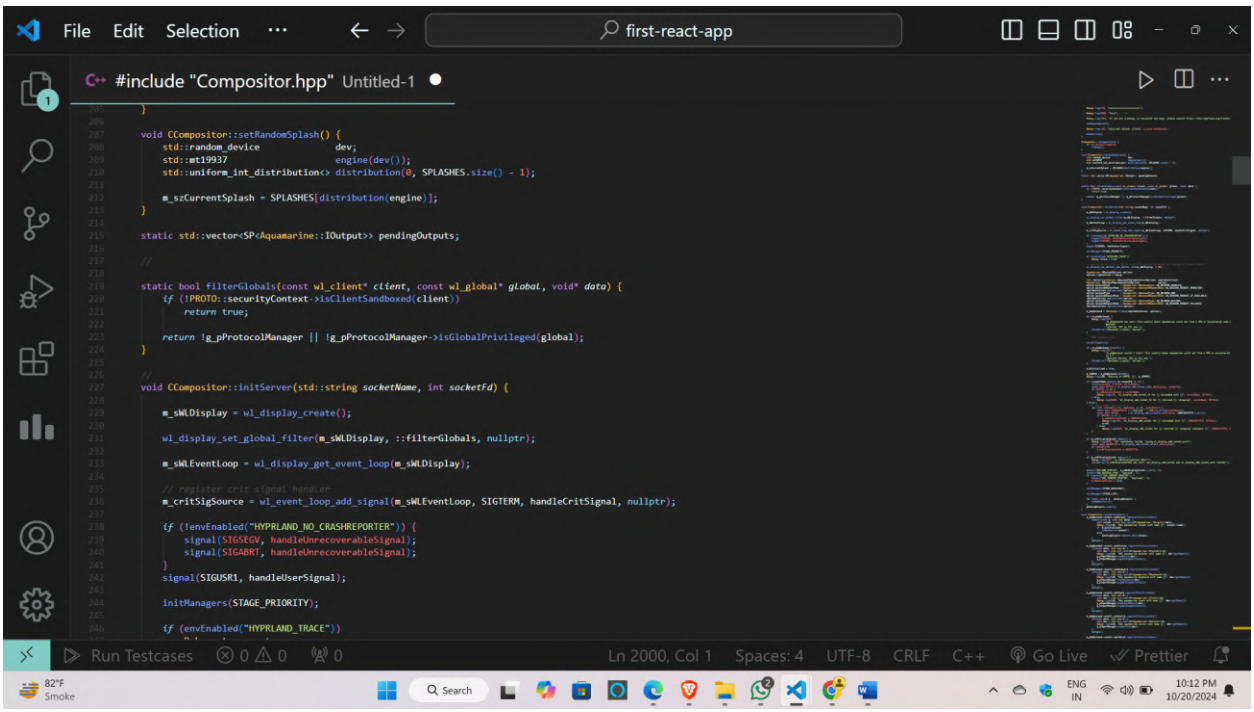

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
#include "Compositor.hpp"
#include "debug/Log.hpp"
#include "helpers/Splashes.hpp"
#include "config/ConfigValue.hpp"
#include "managers/CursorManager.hpp"
#include "managers/TokenManager.hpp"
#include "managers/PointerManager.hpp"
#include "managers/SeatManager.hpp"
#include "managers/eventLoop/EventLoopManager.hpp"
#include <aquamarine/output/Output.hpp>
#include <bit>
#include <ctime>
#include <random>
#include <print>
#include <cstring>
#include <filesystem>
#include <unordered_set>
#include "debug/HyprCtl.hpp"
#include "debug/CrashReporter.hpp"
#ifdef USES_SYSTEMD
#include <helpers/SdDaemon.hpp> // for SdNotify
#endif
#include <ranges>
#include "helpers/varlist/VarList.hpp"
#include "protocols/FractionalScale.hpp"
#include "protocols/PointerConstraints.hpp"
#include "protocols/LayerShell.hpp"
#include "protocols/XDGShell.hpp"
#include "protocols/XDGOutput.hpp"
#include "protocols/SecurityContext.hpp"
#include "protocols/core/Compositor.hpp"
#include "protocols/core/Subcompositor.hpp"
#include "desktop/LayerSurface.hpp"
#include "render/Renderer.hpp"
#include "xwayland/XWayland.hpp"
#include "helpers/ByteOperations.hpp"
#include <hyprutils/string/String.hpp>
#include <aquamarine/input/Input.hpp>
#include <fcntl.h>
#include <sys/types.h>
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
USD/CNY -0.31%
Search
ENG IN
10:09 PM 10/20/2024

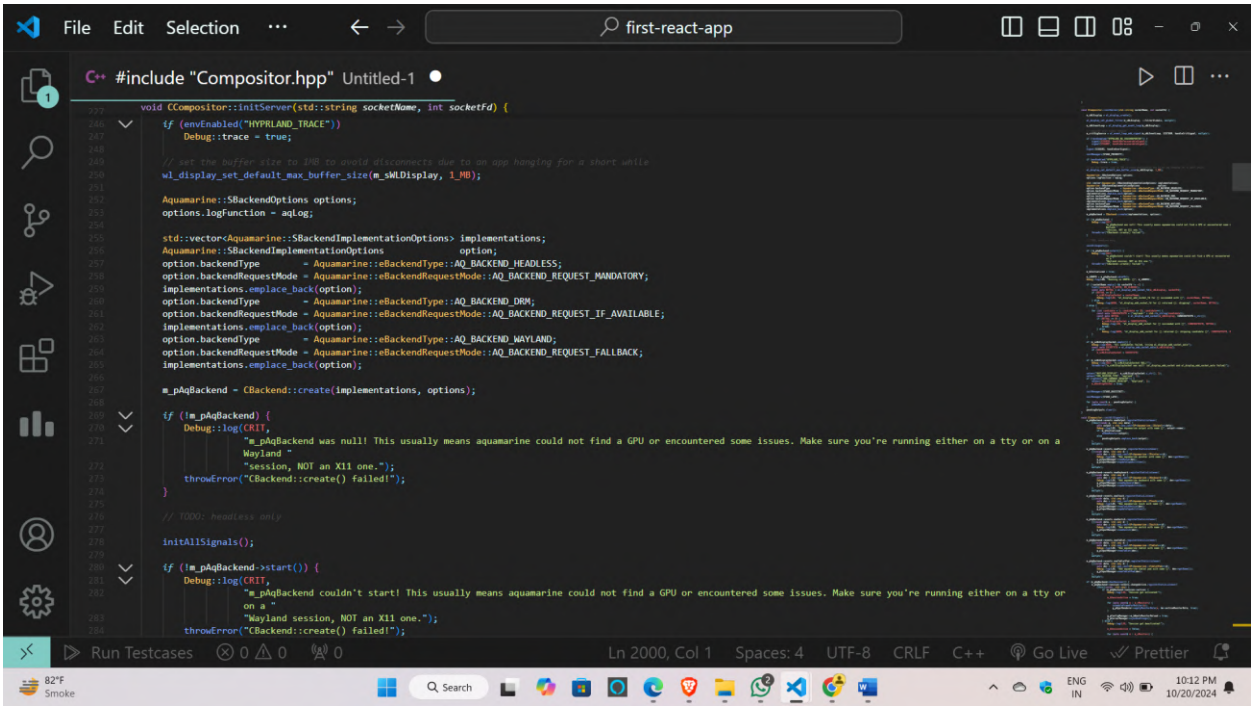

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/resource.h>
using namespace Hyprutils::String;
using namespace Aquamarine;
int handleCritSignal(int signo, void* data) {
Debug::log(LOG, "Hyprland received signal {}", signo);
if (signo == SIGTERM || signo == SIGINT || signo == SIGKILL)
g_pCompositor->stopCompositor();
return 0;
}
void handleUnrecoverableSignal(int sig) {
// remove our handlers
signal(SIGABRT, SIG_DFL);
signal(SIGSEGV, SIG_DFL);
if (g_pHookSystem && g_pHookSystem->m_bCurrentEventPlugin) {
longjmp(g_pHookSystem->m_jbHookFaultJumpBuf, 1);
return;
}
// Kill the program if the crash reporter is caught in a deadlock.
signal(SIGALRM, [](int _) {
char const* msg = "\nCrashReporter exceeded timeout, forcefully exiting\n";
write(2, msg, strlen(msg));
abort();
});
alarm(15);
CrashReporter::createAndSaveCrash(sig);
abort();
}
void handleUserSignal(int sig) {
if (sig == SIGUSR1) {
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
SL - WI Live
Search
ENG IN
10:10 PM 10/20/2024

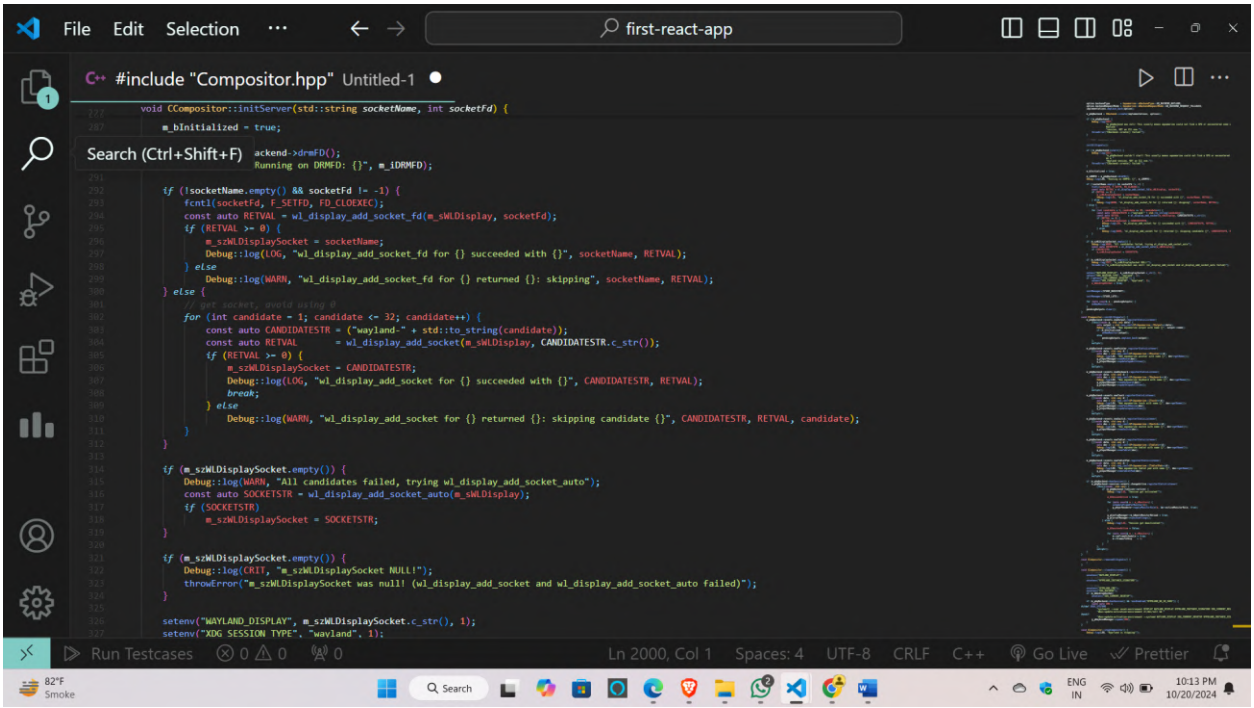

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void handleUserSignal(int sig) {
if (sig == SIGUSR1) {
throw std::exception();
}
}
static LogLevel aqLevelToHl(Aquamarine::eBackendLogLevel level) {
switch (level) {
case Aquamarine::eBackendLogLevel::AQ_LOG_TRACE: return TRACE;
case Aquamarine::eBackendLogLevel::AQ_LOG_DEBUG: return LOG;
case Aquamarine::eBackendLogLevel::AQ_LOG_ERROR: return ERR;
case Aquamarine::eBackendLogLevel::AQ_LOG_WARNING: return WARN;
case Aquamarine::eBackendLogLevel::AQ_LOG_CRITICAL: return CRIT;
default: break;
}
return NONE;
}
void aqLog(Aquamarine::eBackendLogLevel level, std::string msg) {
Debug::log(aqLevelToHl(level), "[AQ] {}", msg);
}
void CCompositor::bumpNofile() {
if (!getrlimit(RLIMIT_NOFILE, &m_sOriginalNofile))
Debug::log(LOG, "Old rlimit: soft -> {}, hard -> {}", m_sOriginalNofile.rlim_cur, m_sOriginalNofile.rlim_max);
else {
Debug::log(ERR, "Failed to get NOFILE rlimits");
m_sOriginalNofile.rlim_max = 0;
return;
}
rlimit newLimit = m_sOriginalNofile;
newLimit.rlim_cur = newLimit.rlim_max;
if (setrlimit(RLIMIT_NOFILE, &newLimit) < 0) {
Debug::log(ERR, "Failed bumping NOFILE limits higher");
m_sOriginalNofile.rlim_max = 0;
return;
}
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:11 PM 10/20/2024

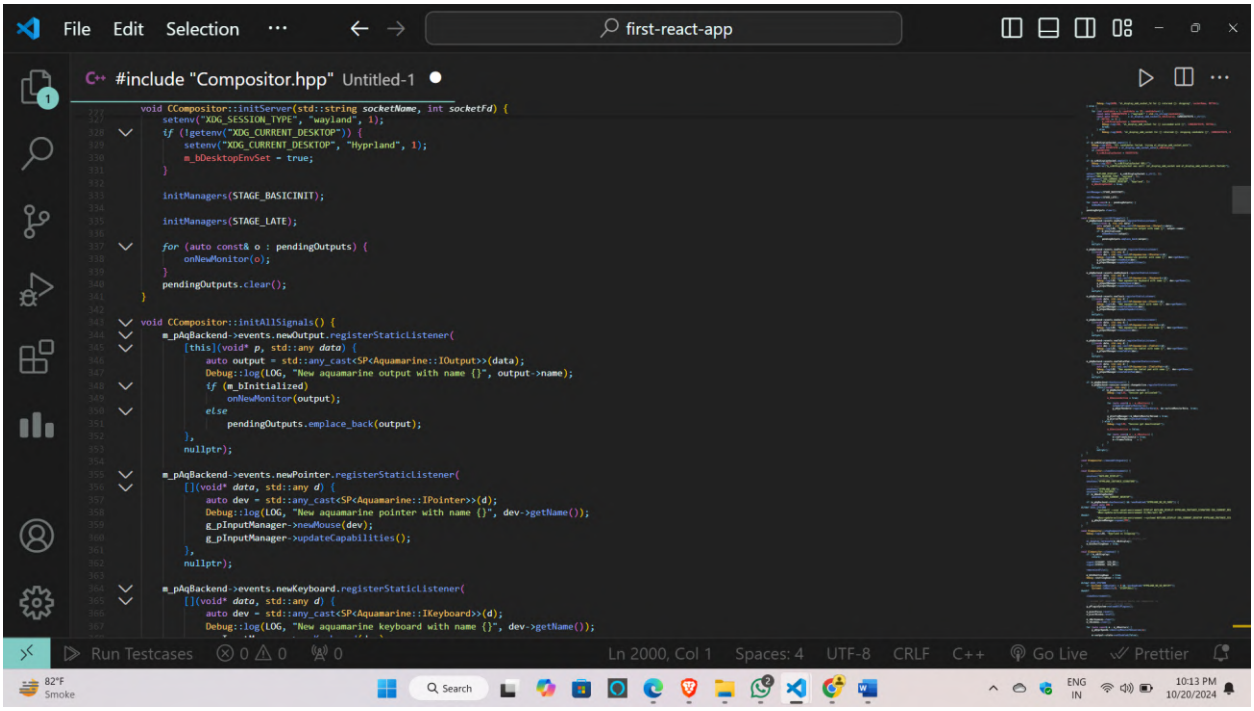

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::bumpNofile() {
if (!getrlimit(RLIMIT_NOFILE, &newLimit))
Debug::log(LOG, "New rlimit: soft -> {}, hard -> {}", newLimit.rlim_cur, newLimit.rlim_max);
}
void CCompositor::restoreNofile() {
if (m_sOriginalNofile.rlim_max <= 0)
return;
if (setrlimit(RLIMIT_NOFILE, &m_sOriginalNofile) < 0)
Debug::log(ERR, "Failed restoring NOFILE limits");
}
CCompositor::CCompositor() {
m_iHyprlandPID = getpid();
m_szHyprTempDataRoot = std::string(getenv("XDG_RUNTIME_DIR")) + "/hypr";
if (m_szHyprTempDataRoot.starts_with("/hypr")) {
std::println("Bailing out, $XDG_RUNTIME_DIR is invalid");
throw std::runtime_error("CCompositor() failed");
}
if (!m_szHyprTempDataRoot.starts_with("/run/user"))
std::println("[!!WARNING!!] XDG_RUNTIME_DIR looks non-standard. Proceeding anyways...");
std::random_device dev;
std::mt19937 engine(dev());
std::uniform_int_distribution<> distribution(0, INT32_MAX);
m_szInstanceSignature = std::format("{}_{}_{}", GIT_COMMIT_HASH, std::time(NULL), distribution(engine));
setenv("HYPRLAND_INSTANCE_SIGNATURE", m_szInstanceSignature.c_str(), true);
if (!std::filesystem::exists(m_szHyprTempDataRoot))
mkdir(m_szHyprTempDataRoot.c_str(), S_IRWXU);
else if (!std::filesystem::is_directory(m_szHyprTempDataRoot)) {
std::println("Bailing out, {} is not a directory", m_szHyprTempDataRoot);
throw std::runtime_error("CCompositor() failed");
}
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:11 PM 10/20/2024

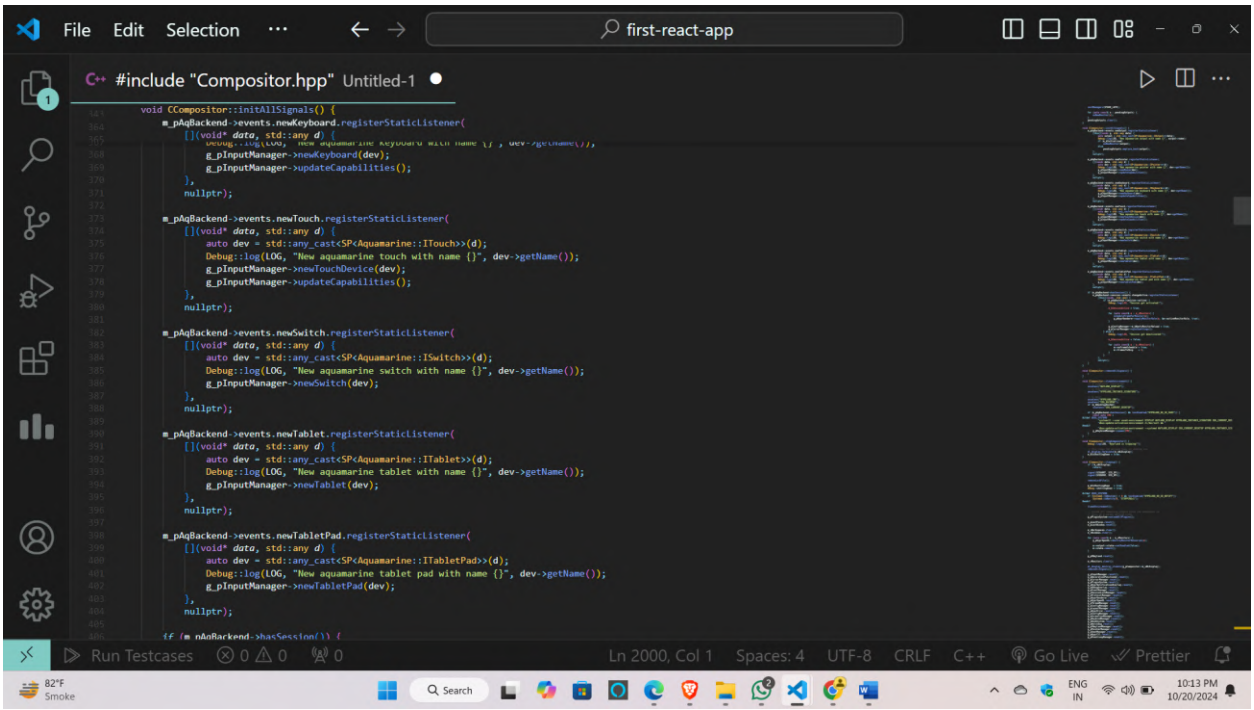

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
CCompositor::CCompositor() {
m_szInstancePath = m_szHyprTempDataRoot + "/" + m_szInstanceSignature;
if (std::filesystem::exists(m_szInstancePath)) {
std::println("Bailing out, {} exists??", m_szInstancePath);
throw std::runtime_error("CCompositor() failed");
}
if (mkdir(m_szInstancePath.c_str(), S_IRWXU) < 0) {
std::println("Bailing out, couldn't create {}", m_szInstancePath);
throw std::runtime_error("CCompositor() failed");
}
Debug::init(m_szInstancePath);
Debug::log(LOG, "Instance Signature: {}", m_szInstanceSignature);
Debug::log(LOG, "Runtime directory: {}", m_szInstancePath);
Debug::log(LOG, "Hyprland PID: {}", m_iHyprlandPID);
Debug::log(LOG, "===== SYSTEM INFO: =====");
logSystemInfo();
Debug::log(LOG, "========================");
Debug::log(NONE, "\n\n"); // pad
Debug::log(INFO, "If you are crashing, or encounter any bugs, please consult https://wiki.hyprland.org/Crashes-and-Bugs/\n\n");
setRandomSplash();
Debug::log(LOG, "\nCurrent splash: {}\n\n", m_szCurrentSplash);
bumpNofile();
}
CCompositor::~CCompositor() {
if (!m_bIsShuttingDown)
cleanup();
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:12 PM
10/20/2024

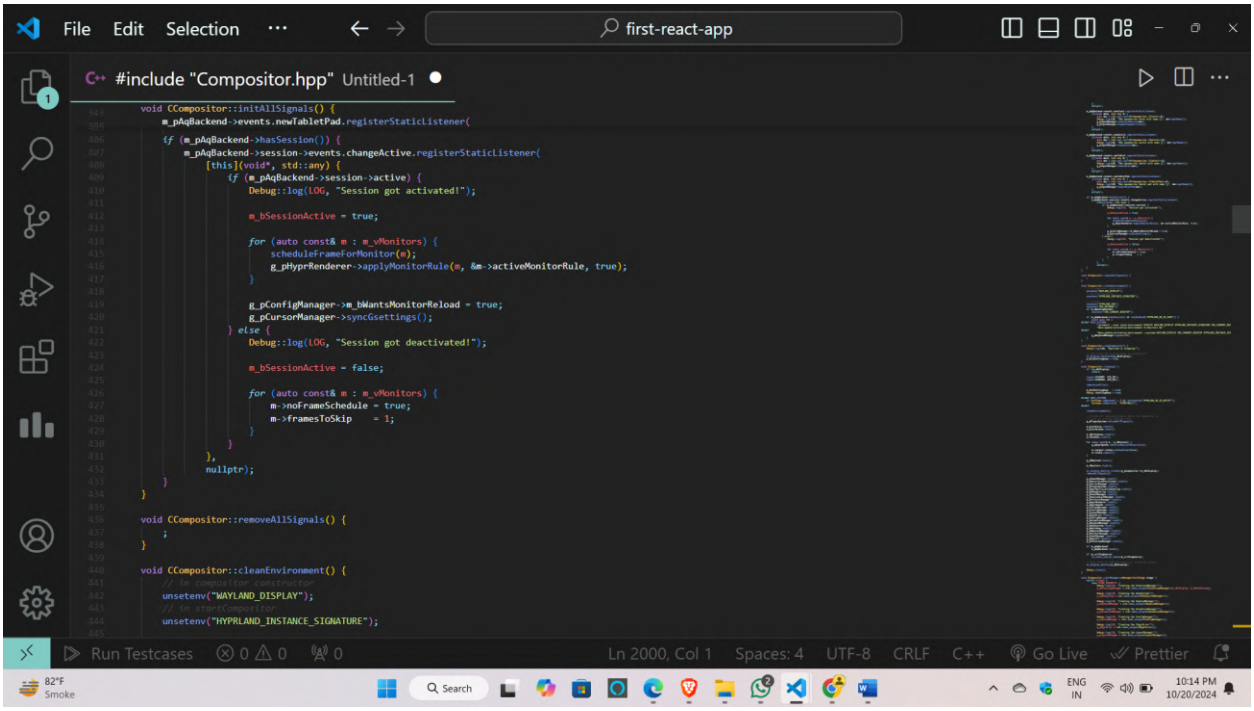

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
}
void CCompositor::setRandomSplash() {
std::random_device dev;
std::mt19937 engine(dev());
std::uniform_int_distribution<> distribution(0, SPLASHES.size() - 1);
m_szCurrentSplash = SPLASHES[distribution(engine)];
}
static std::vector<SP<Aquamarine::IOutput>> pendingOutputs;
//
static bool filterGlobals(const wl_client* client, const wl_global* global, void* data) {
if (!PROTO::securityContext->isClientSandboxed(client))
return true;
return !g_pProtocolManager || !g_pProtocolManager->isGlobalPrivileged(global);
}
//
void CCompositor::initServer(std::string socketName, int socketFd) {
m_sWLDisplay = wl_display_create();
wl_display_set_global_filter(m_sWLDisplay, ::filterGlobals, nullptr);
m_sWLEventLoop = wl_display_get_event_loop(m_sWLDisplay);
// register crit signal handler
m_critSigSource = wl_event_loop_add_signal(m_sWLEventLoop, SIGTERM, handleCritSignal, nullptr);
if (!envEnabled("HYPRLAND_NO_CRASHREPORTER")) {
signal(SIGSEGV, handleUnrecoverableSignal);
signal(SIGABRT, handleUnrecoverableSignal);
}
signal(SIGUSR1, handleUserSignal);
initManagers(STAGE_PRIORITY);
if (envEnabled("HYPRLAND_TRACE"))
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:12 PM
10/20/2024

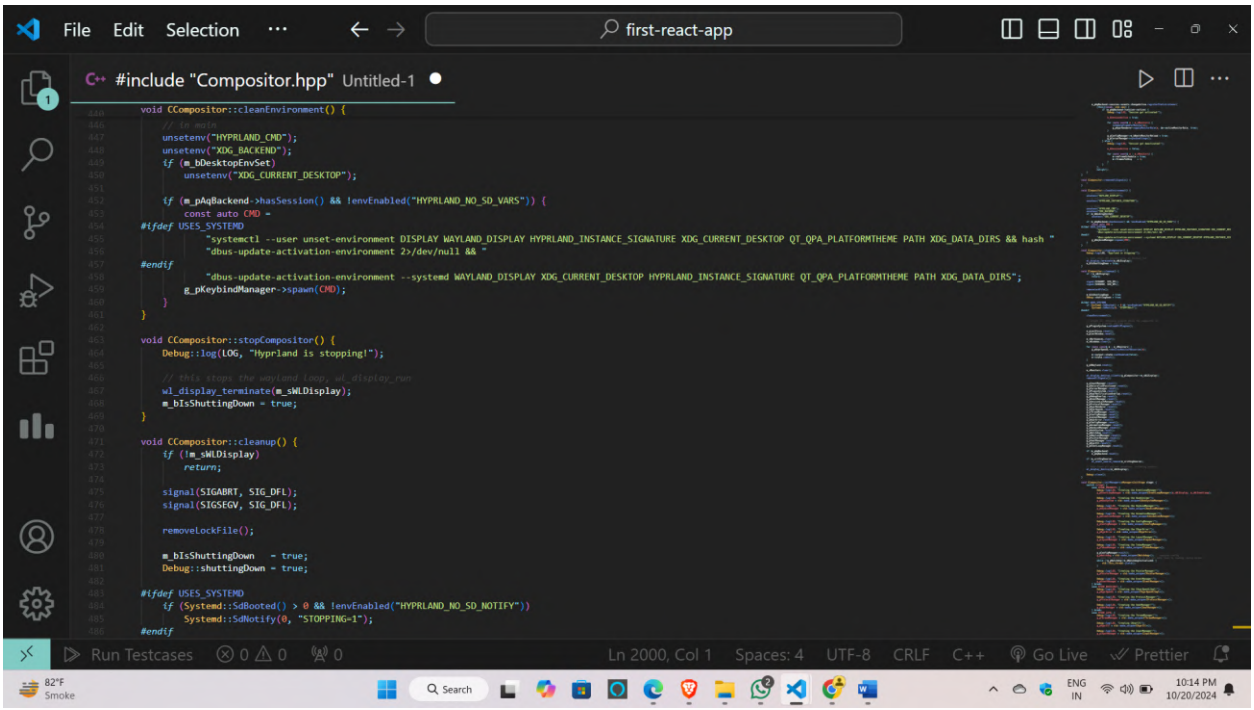

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::initServer(std::string socketName, int socketFd) {
if (envEnabled("HYPRLAND_TRACE"))
Debug::trace = true;
// set the buffer size to 1MB to avoid disconnects due to an app hanging for a short while
wl_display_set_default_max_buffer_size(m_sWLDisplay, 1_MB);
Aquamarine::SBackendOptions options;
options.logFunction = aqLog;
std::vector<Aquamarine::SBackendImplementationOptions> implementations;
Aquamarine::SBackendImplementationOptions option;
option.backendType = Aquamarine::eBackendType::AQ_BACKEND_HEADLESS;
option.backendRequestMode = Aquamarine::eBackendRequestMode::AQ_BACKEND_REQUEST_MANDATORY;
implementations.emplace_back(option);
option.backendType = Aquamarine::eBackendType::AQ_BACKEND_DRM;
option.backendRequestMode = Aquamarine::eBackendRequestMode::AQ_BACKEND_REQUEST_IF_AVAILABLE;
implementations.emplace_back(option);
option.backendType = Aquamarine::eBackendType::AQ_BACKEND_WAYLAND;
option.backendRequestMode = Aquamarine::eBackendRequestMode::AQ_BACKEND_REQUEST_FALLBACK;
implementations.emplace_back(option);
m_pAqBackend = CBackend::create(implementations, options);
if (!m_pAqBackend) {
Debug::log(CRIT,
"m_pAqBackend was null! This usually means aquamarine could not find a GPU or encountered some issues. Make sure you're running either on a tty or on a
Wayland "
"session, NOT an X11 one.");
throwError("CBackend::create() failed!");
}
// TODO: headless only
initAllSignals();
if (!m_pAqBackend->start()) {
Debug::log(CRIT,
"m_pAqBackend couldn't start! This usually means aquamarine could not find a GPU or encountered some issues. Make sure you're running either on a tty or
on a "
"Wayland session, NOT an X11 one.");
throwError("CBackend::create() failed!");
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:12 PM
10/20/2024

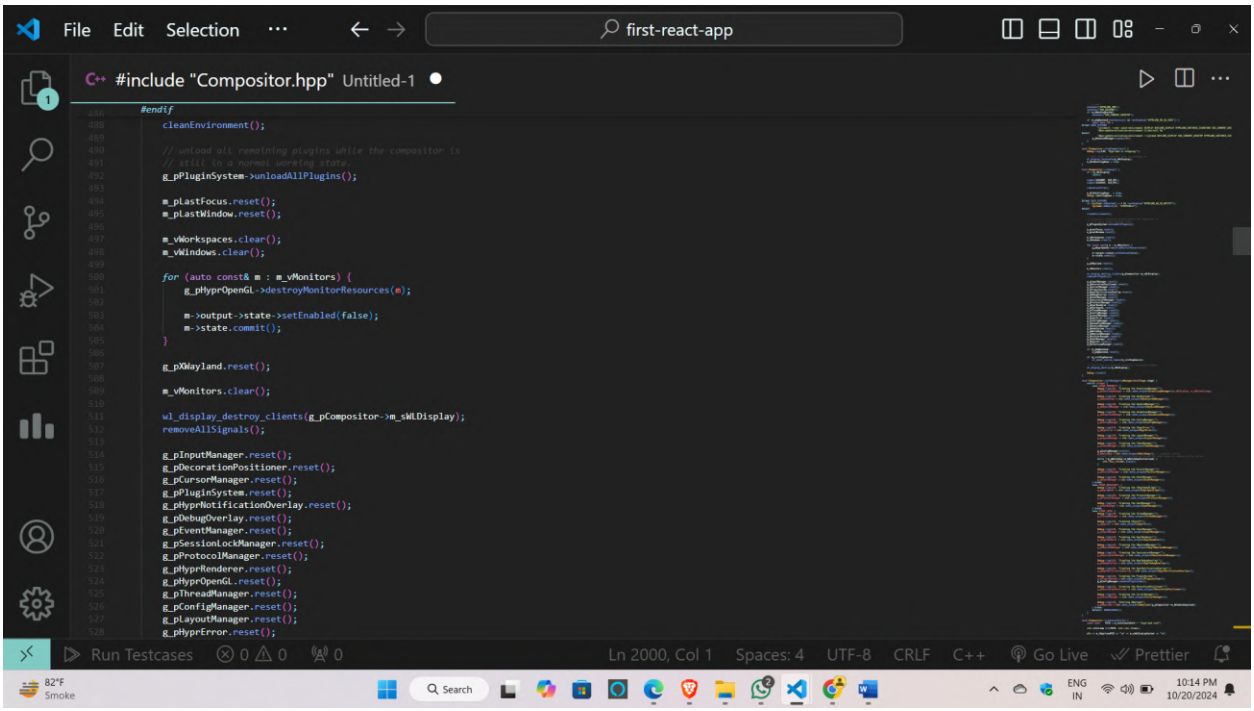

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::initServer(std::string socketName, int socketFd) {
m_bInitialized = true;
Search (Ctrl+Shift+F)
ackend->drmFD();
Running on DRMFD: {}", m_iDRMFD);
if (!socketName.empty() && socketFd != -1) {
fcntl(socketFd, F_SETFD, FD_CLOEXEC);
const auto RETVAL = wl_display_add_socket_fd(m_sWLDisplay, socketFd);
if (RETVAL >= 0) {
m_szWLDisplaySocket = socketName;
Debug::log(LOG, "wl_display_add_socket_fd for {} succeeded with {}", socketName, RETVAL);
} else
Debug::log(WARN, "wl_display_add_socket_fd for {} returned {}: skipping", socketName, RETVAL);
} else {
// get socket, avoid using 0
for (int candidate = 1; candidate <= 32; candidate++) {
const auto CANDIDATESTR = ("wayland-" + std::to_string(candidate));
const auto RETVAL = wl_display_add_socket(m_sWLDisplay, CANDIDATESTR.c_str());
if (RETVAL >= 0) {
m_szWLDisplaySocket = CANDIDATESTR;
Debug::log(LOG, "wl_display_add_socket for {} succeeded with {}", CANDIDATESTR, RETVAL);
break;
} else
Debug::log(WARN, "wl_display_add_socket for {} returned {}: skipping candidate {}", CANDIDATESTR, RETVAL, candidate);
}
}
if (m_szWLDisplaySocket.empty()) {
Debug::log(WARN, "All candidates failed, trying wl_display_add_socket_auto");
const auto SOCKETSTR = wl_display_add_socket_auto(m_sWLDisplay);
if (SOCKETSTR)
m_szWLDisplaySocket = SOCKETSTR;
}
if (m_szWLDisplaySocket.empty()) {
Debug::log(CRIT, "m_szWLDisplaySocket NULL!");
throwError("m_szWLDisplaySocket was null! (wl_display_add_socket and wl_display_add_socket_auto failed)");
}
setenv("WAYLAND_DISPLAY", m_szWLDisplaySocket.c_str(), 1);
setenv("XDG_SESSION_TYPE", "wayland", 1);
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:13 PM
10/20/2024

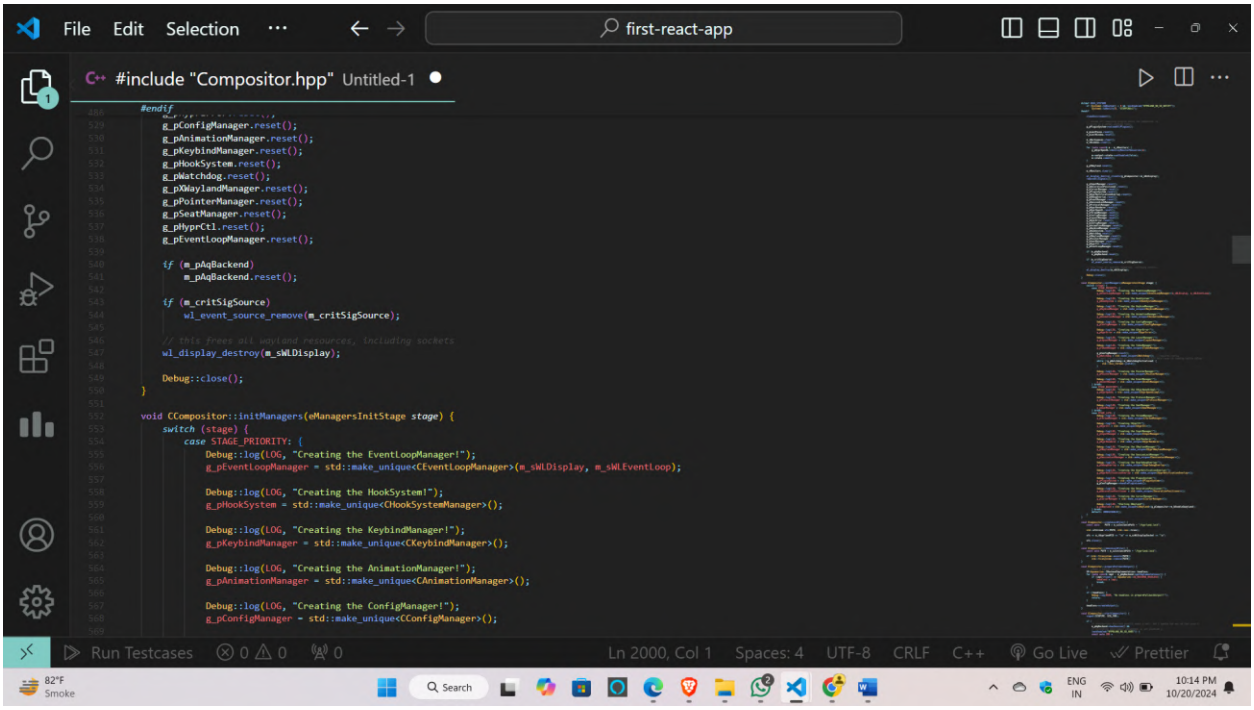

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::initServer(std::string socketName, int socketFd) {
setenv("XDG_SESSION_TYPE", "wayland", 1);
if (!getenv("XDG_CURRENT_DESKTOP")) {
setenv("XDG_CURRENT_DESKTOP", "Hyprland", 1);
m_bDesktopEnvSet = true;
}
initManagers(STAGE_BASICINIT);
initManagers(STAGE_LATE);
for (auto const& o : pendingOutputs) {
onNewMonitor(o);
}
pendingOutputs.clear();
}
void CCompositor::initAllSignals() {
m_pAqBackend->events.newOutput.registerStaticListener(
[this](void* p, std::any data) {
auto output = std::any_cast<SP<Aquamarine::IOutput>>(data);
Debug::log(LOG, "New aquamarine output with name {}", output->name);
if (m_bInitialized)
onNewMonitor(output);
else
pendingOutputs.emplace_back(output);
},
nullptr);
m_pAqBackend->events.newPointer.registerStaticListener(
[](void* data, std::any d) {
auto dev = std::any_cast<SP<Aquamarine::IPointer>>(d);
Debug::log(LOG, "New aquamarine pointer with name {}", dev->getName());
g_pInputManager->newMouse(dev);
g_pInputManager->updateCapabilities();
},
nullptr);
m_pAqBackend->events.newKeyboard.registerStaticListener(
[](void* data, std::any d) {
auto dev = std::any_cast<SP<Aquamarine::IKeyboard>>(d);
Debug::log(LOG, "New aquamarine keyboard with name {}", dev->getName());
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:13 PM
10/20/2024

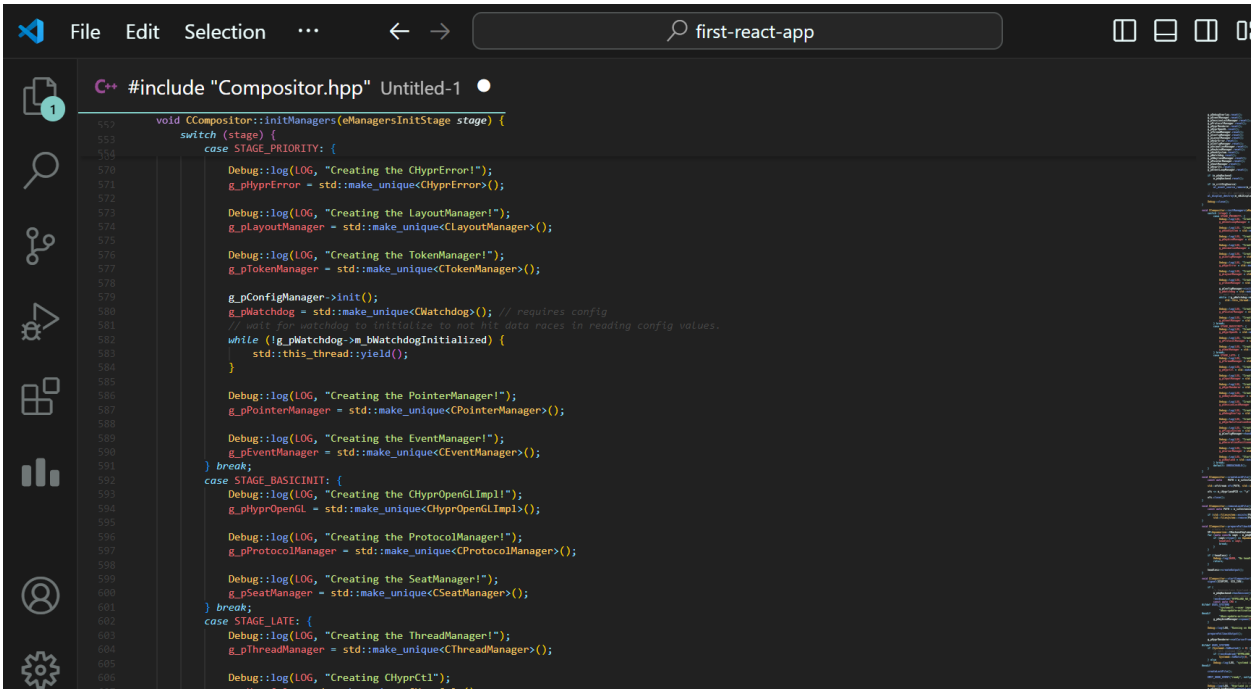

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::initAllSignals() {
m_pAqBackend->events.newKeyboard.registerStaticListener(
[](void* data, std::any d) {
g_pInputManager->newKeyboard(dev);
g_pInputManager->updateCapabilities();
},
nullptr);
m_pAqBackend->events.newTouch.registerStaticListener(
[](void* data, std::any d) {
auto dev = std::any_cast<SP<Aquamarine::ITouch>>(d);
Debug::log(LOG, "New aquamarine touch with name {}", dev->getName());
g_pInputManager->newTouchDevice(dev);
g_pInputManager->updateCapabilities();
},
nullptr);
m_pAqBackend->events.newSwitch.registerStaticListener(
[](void* data, std::any d) {
auto dev = std::any_cast<SP<Aquamarine::ISwitch>>(d);
Debug::log(LOG, "New aquamarine switch with name {}", dev->getName());
g_pInputManager->newSwitch(dev);
},
nullptr);
m_pAqBackend->events.newTablet.registerStaticListener(
[](void* data, std::any d) {
auto dev = std::any_cast<SP<Aquamarine::ITablet>>(d);
Debug::log(LOG, "New aquamarine tablet with name {}", dev->getName());
g_pInputManager->newTablet(dev);
},
nullptr);
m_pAqBackend->events.newTabletPad.registerStaticListener(
[](void* data, std::any d) {
auto dev = std::any_cast<SP<Aquamarine::ITabletPad>>(d);
Debug::log(LOG, "New aquamarine tablet pad with name {}", dev->getName());
g_pInputManager->newTabletPad(dev);
},
nullptr);
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:13 PM
10/20/2024

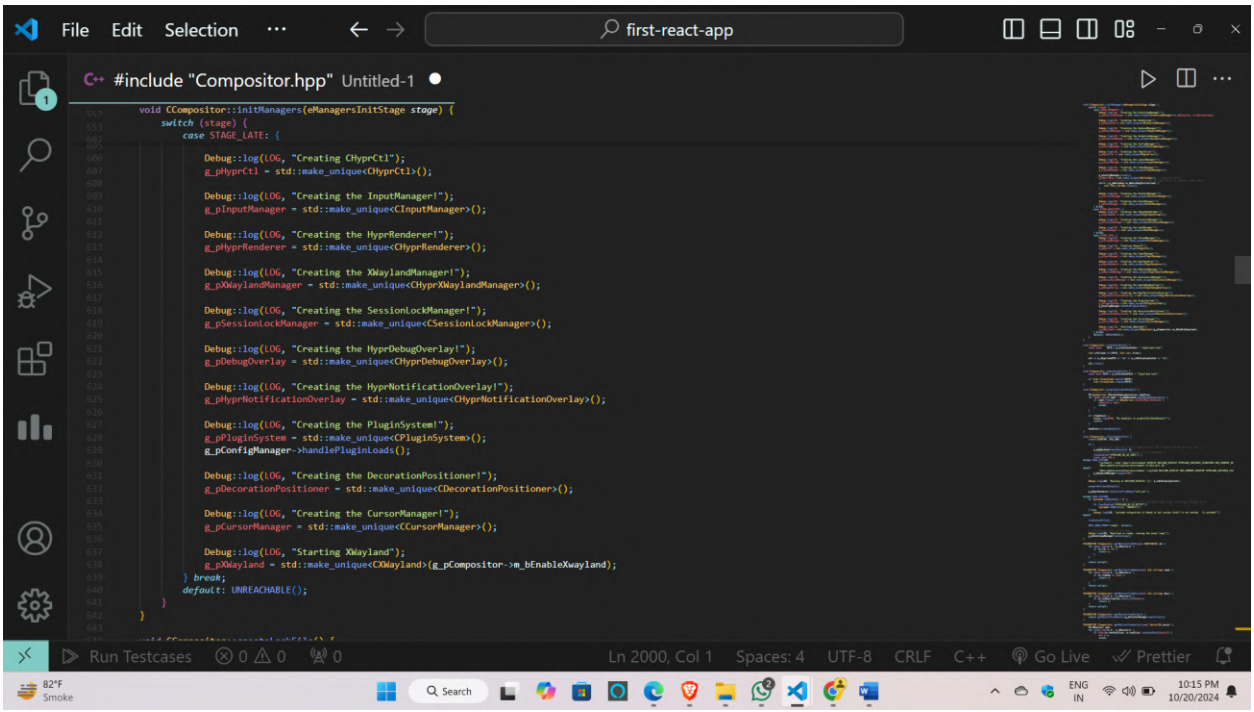

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::initAllSignals() {
m_pAqBackend->events.newTabletPad.registerStaticListener(
if (m_pAqBackend->hasSession()) {
m_pAqBackend->session->events.changeActive.registerStaticListener(
[this](void*, std::any) {
if (m_pAqBackend->session->active) {
Debug::log(LOG, "Session got activated!");
m_bSessionActive = true;
for (auto const& m : m_vMonitors) {
scheduleFrameForMonitor(m);
g_pHyprRenderer->applyMonitorRule(m, &m->activeMonitorRule, true);
}
g_pConfigManager->m_bWantsMonitorReload = true;
g_pCursorManager->syncGsettings();
} else {
Debug::log(LOG, "Session got deactivated!");
m_bSessionActive = false;
for (auto const& m : m_vMonitors) {
m->noFrameSchedule = true;
m->framesToSkip = 1;
}
}
),
nullptr);
}
}
void CCompositor::removeAllSignals() {
;
}
void CCompositor::cleanEnvironment() {
unsetenv("WAYLAND_DISPLAY");
unsetenv("HYPRLAND_INSTANCE_SIGNATURE");
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:14 PM
10/20/2024

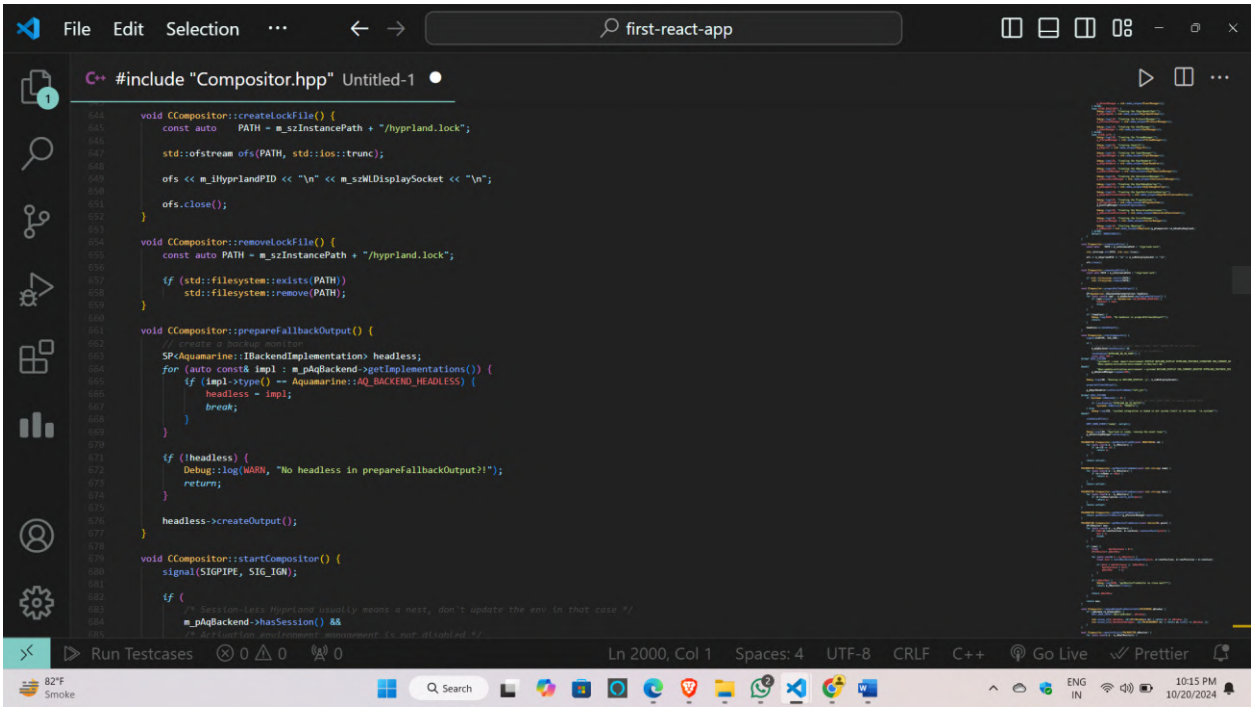

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::cleanEnvironment() {
unsetenv("HYPRLAND_CMD");
unsetenv("XDG_BACKEND");
if (m_bDesktopEnvSet)
unsetenv("XDG_CURRENT_DESKTOP");
if (m_pAqBackend->hasSession() && !envEnabled("HYPRLAND_NO_SD_VARS")) {
const auto CMD =
#ifdef USES_SYSTEMD
"systemctl --user unset-environment DISPLAY WAYLAND_DISPLAY HYPRLAND_INSTANCE_SIGNATURE XDG_CURRENT_DESKTOP QT_QPA_PLATFORMTHEME PATH XDG_DATA_DIRS && hash "
"dbus-update-activation-environment 2>/dev/null && "
#endif
"dbus-update-activation-environment --systemd WAYLAND_DISPLAY XDG_CURRENT_DESKTOP HYPRLAND_INSTANCE_SIGNATURE QT_QPA_PLATFORMTHEME PATH XDG_DATA_DIRS";
g_pKeybindManager->spawn(CMD);
}
}
void CCompositor::stopCompositor() {
Debug::log(LOG, "Hyprland is stopping!");
wl_display_terminate(m_sWLDisplay);
m_bIsShuttingDown = true;
}
void CCompositor::cleanup() {
if (!m_sWLDisplay)
return;
signal(SIGABRT, SIG_DFL);
signal(SIGSEGV, SIG_DFL);
removeLockFile();
m_bIsShuttingDown = true;
Debug::shuttingDown = true;
#ifdef USES_SYSTEMD
if (Systemd::SdBooted() > 0 && !envEnabled("HYPRLAND_NO_SD_NOTIFY"))
Systemd::SdNotify(0, "STOPPING=1");
#endif
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:14 PM
10/20/2024

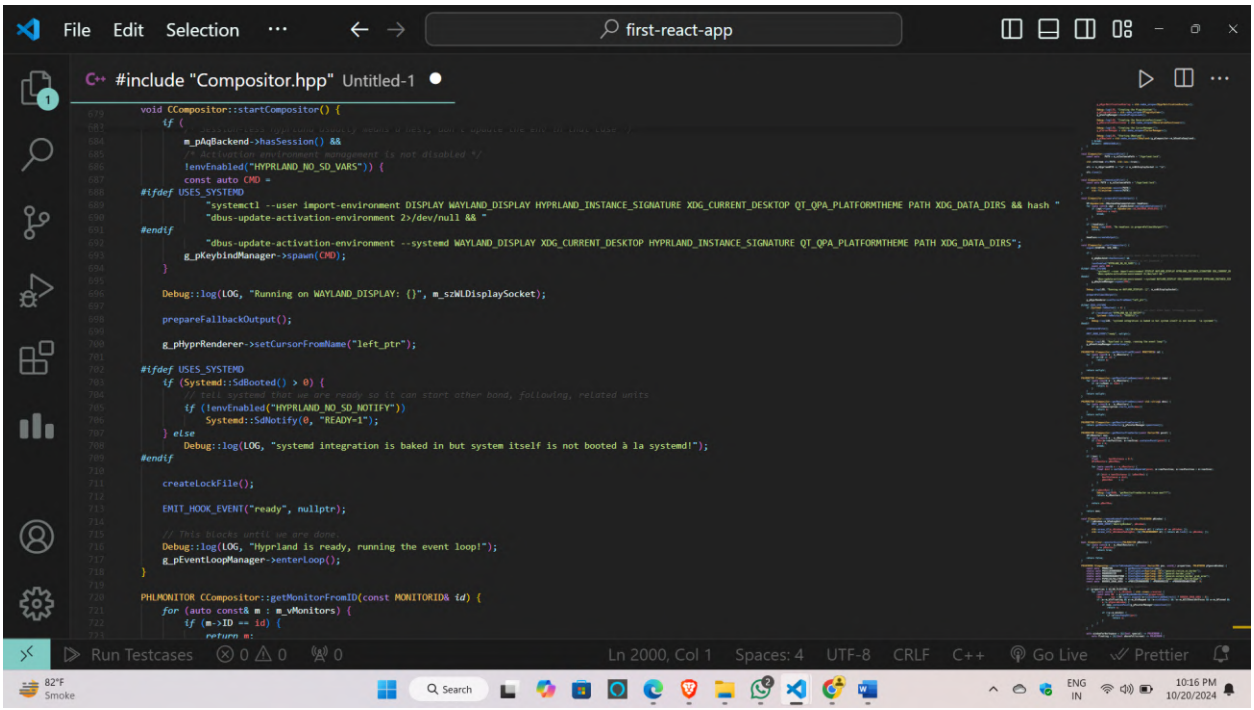

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
#endif
cleanEnvironment();
// unload all remaining plugins while the compositor is
// still in a normal working state
g_pPluginSystem->unloadAllPlugins();
m_pLastFocus.reset();
m_pLastWindow.reset();
m_vWorkspaces.clear();
m_vWindows.clear();
for (auto const& m : m_vMonitors) {
g_pHyprOpenGL->destroyMonitorResources(m);
m->output->state->setEnabled(false);
m->state.commit();
}
g_pXWayland.reset();
m_vMonitors.clear();
wl_display_destroy_clients(g_pCompositor->m_sWLDisplay);
removeAllSignals();
g_pInputManager.reset();
g_pDecorationPositioner.reset();
g_pCursorManager.reset();
g_pPluginSystem.reset();
g_pHyprNotificationOverlay.reset();
g_pDebugOverlay.reset();
g_pEventManager.reset();
g_pSessionLockManager.reset();
g_pProtocolManager.reset();
g_pHyprRenderer.reset();
g_pHyprOpenGL.reset();
g_pThreadManager.reset();
g_pConfigManager.reset();
g_pLayoutManager.reset();
g_pHyprError.reset();
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:14 PM
10/20/2024

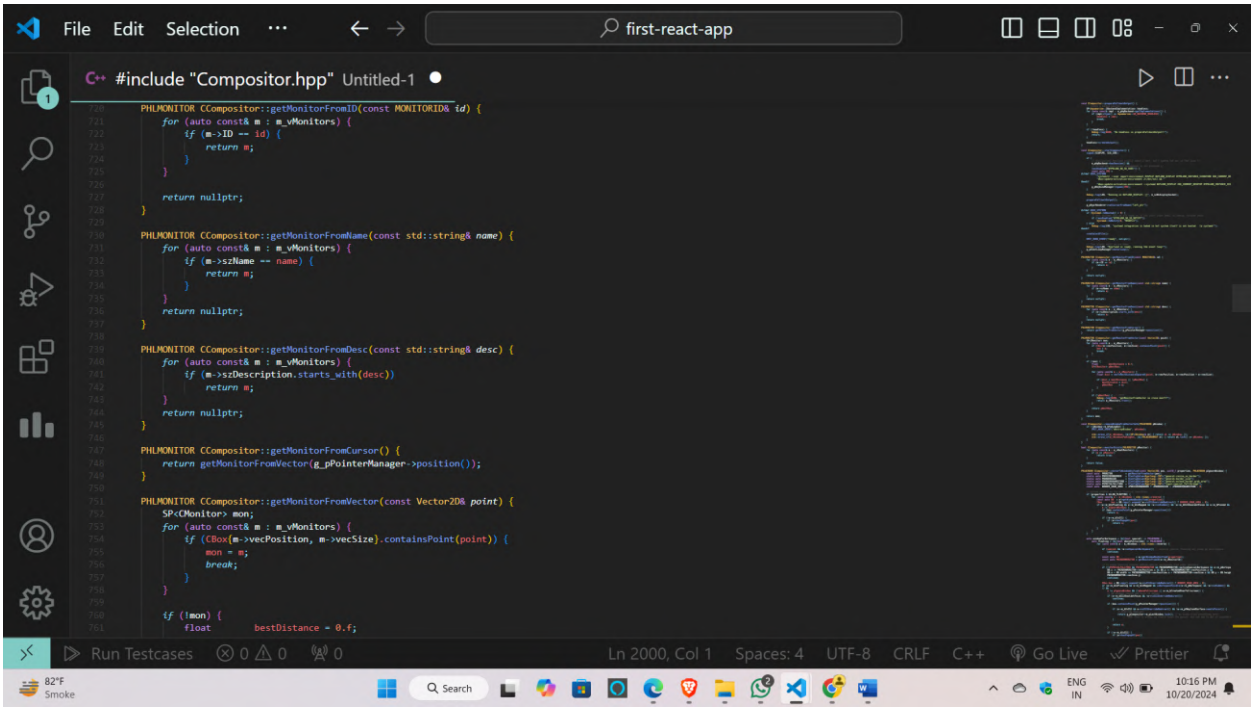

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
#endif
g_pConfigManager.reset();
g_pAnimationManager.reset();
g_pKeybindManager.reset();
g_pHookSystem.reset();
g_pWatchdog.reset();
g_pXWaylandManager.reset();
g_pPointerManager.reset();
g_pSeatManager.reset();
g_pHyprCtl.reset();
g_pEventLoopManager.reset();
if (m_pAqBackend)
m_pAqBackend.reset();
if (m_critSigSource)
wl_event_source_remove(m_critSigSource);
// this frees all wayland resources, including sockets
wl_display_destroy(m_sWLDisplay);
Debug::close();
}
void CCompositor::initManagers(eManagersInitStage stage) {
switch (stage) {
case STAGE_PRIORITY: {
Debug::log(LOG, "Creating the EventLoopManager!");
g_pEventLoopManager = std::make_unique<CEventLoopManager>(m_sWLDisplay, m_sWLEventLoop);
Debug::log(LOG, "Creating the HookSystem!");
g_pHookSystem = std::make_unique<CHookSystemManager>();
Debug::log(LOG, "Creating the KeybindManager!");
g_pKeybindManager = std::make_unique<CKeybindManager>();
Debug::log(LOG, "Creating the AnimationManager!");
g_pAnimationManager = std::make_unique<CAnimationManager>();
Debug::log(LOG, "Creating the ConfigManager!");
g_pConfigManager = std::make_unique<CConfigManager>();
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:14 PM
10/20/2024

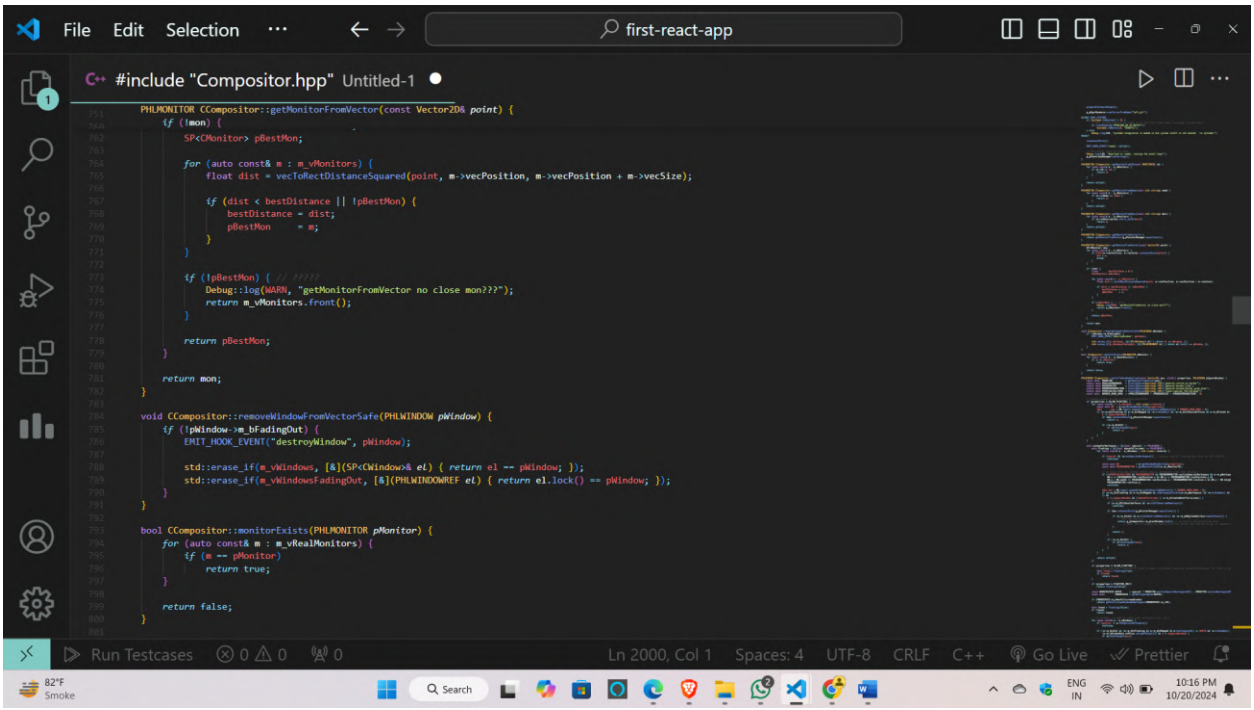

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::initManagers(eManagersInitStage stage) {
switch (stage) {
case STAGE_PRIORITY: {
Debug::log(LOG, "Creating the CHyprError!");
g_pHyprError = std::make_unique<CHyprError>();
Debug::log(LOG, "Creating the LayoutManager!");
g_pLayoutManager = std::make_unique<CLayoutManager>();
Debug::log(LOG, "Creating the TokenManager!");
g_pTokenManager = std::make_unique<CTokenManager>();
g_pConfigManager->init();
g_pWatchdog = std::make_unique<CWatchdog>(); // requires config
// wait for watchdog to initialize to not hit data races in reading config values.
while (!g_pWatchdog->m_bWatchdogInitialized) {
std::this_thread::yield();
}
Debug::log(LOG, "Creating the PointerManager!");
g_pPointerManager = std::make_unique<CPointerManager>();
Debug::log(LOG, "Creating the EventManager!");
g_pEventManager = std::make_unique<CEventManager>();
} break;
case STAGE_BASICINIT: {
Debug::log(LOG, "Creating the CHyprOpenGLImpl!");
g_pHyprOpenGL = std::make_unique<CHyprOpenGLImpl>();
Debug::log(LOG, "Creating the ProtocolManager!");
g_pProtocolManager = std::make_unique<CProtocolManager>();
Debug::log(LOG, "Creating the SeatManager!");
g_pSeatManager = std::make_unique<CSeatManager>();
} break;
case STAGE_LATE: {
Debug::log(LOG, "Creating the ThreadManager!");
g_pThreadManager = std::make_unique<CThreadManager>();
Debug::log(LOG, "Creating CHyprCtl");

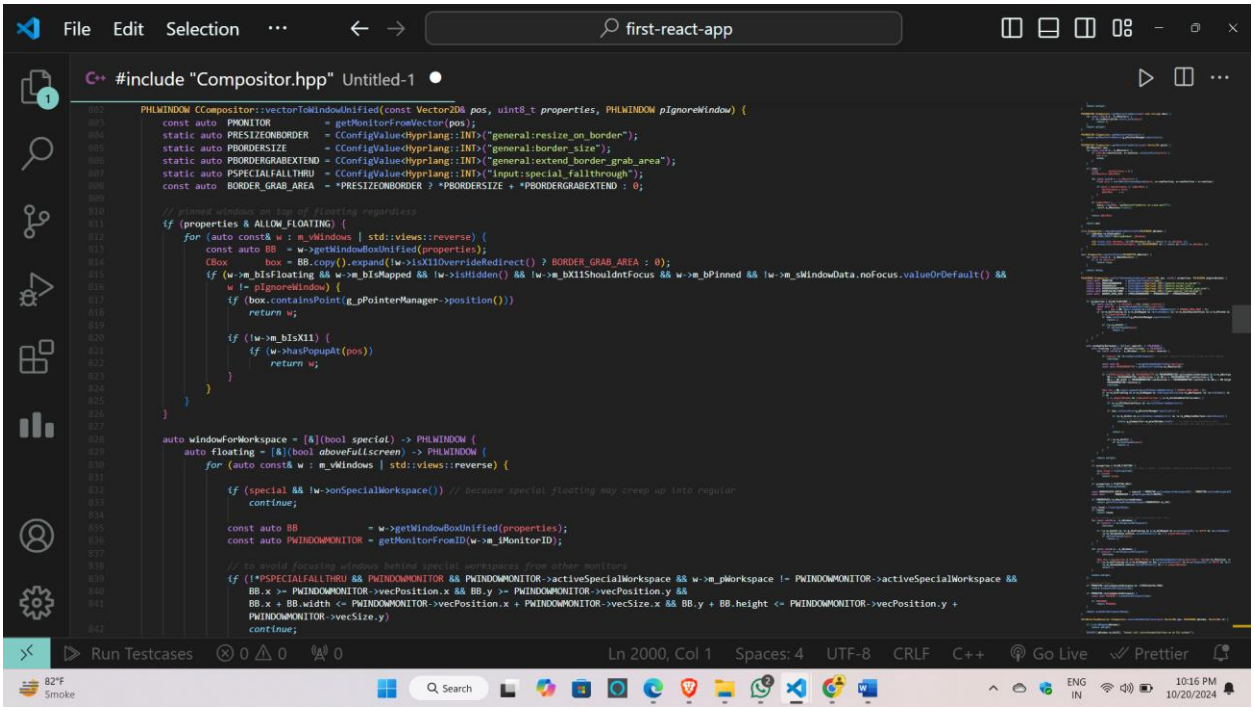

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::initManagers(eManagersInitStage stage) {
switch (stage) {
case STAGE_LATE: {
Debug::log(LOG, "Creating CHyprCtl");
g_pHyprCtl = std::make_unique<CHyprCtl>();
Debug::log(LOG, "Creating the InputManager!");
g_pInputManager = std::make_unique<CInputManager>();
Debug::log(LOG, "Creating the HyprRenderer!");
g_pHyprRenderer = std::make_unique<CHyprRenderer>();
Debug::log(LOG, "Creating the XWaylandManager!");
g_pXWaylandManager = std::make_unique<CHyprXWaylandManager>();
Debug::log(LOG, "Creating the SessionLockManager!");
g_pSessionLockManager = std::make_unique<CSessionLockManager>();
Debug::log(LOG, "Creating the HyprDebugOverlay!");
g_pDebugOverlay = std::make_unique<CHyprDebugOverlay>();
Debug::log(LOG, "Creating the HyprNotificationOverlay!");
g_pHyprNotificationOverlay = std::make_unique<CHyprNotificationOverlay>();
Debug::log(LOG, "Creating the PluginSystem!");
g_pPluginSystem = std::make_unique<CPluginSystem>();
g_pConfigManager->handlePluginLoads();
Debug::log(LOG, "Creating the DecorationPositioner!");
g_pDecorationPositioner = std::make_unique<CDecorationPositioner>();
Debug::log(LOG, "Creating the CursorManager!");
g_pCursorManager = std::make_unique<CCursorManager>();
Debug::log(LOG, "Starting XWayland");
g_pXWayland = std::make_unique<CXWayland>(g_pCompositor->m_bEnableXwayland);
} break;
default: UNREACHABLE();
}
}
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:15 PM
10/20/2024

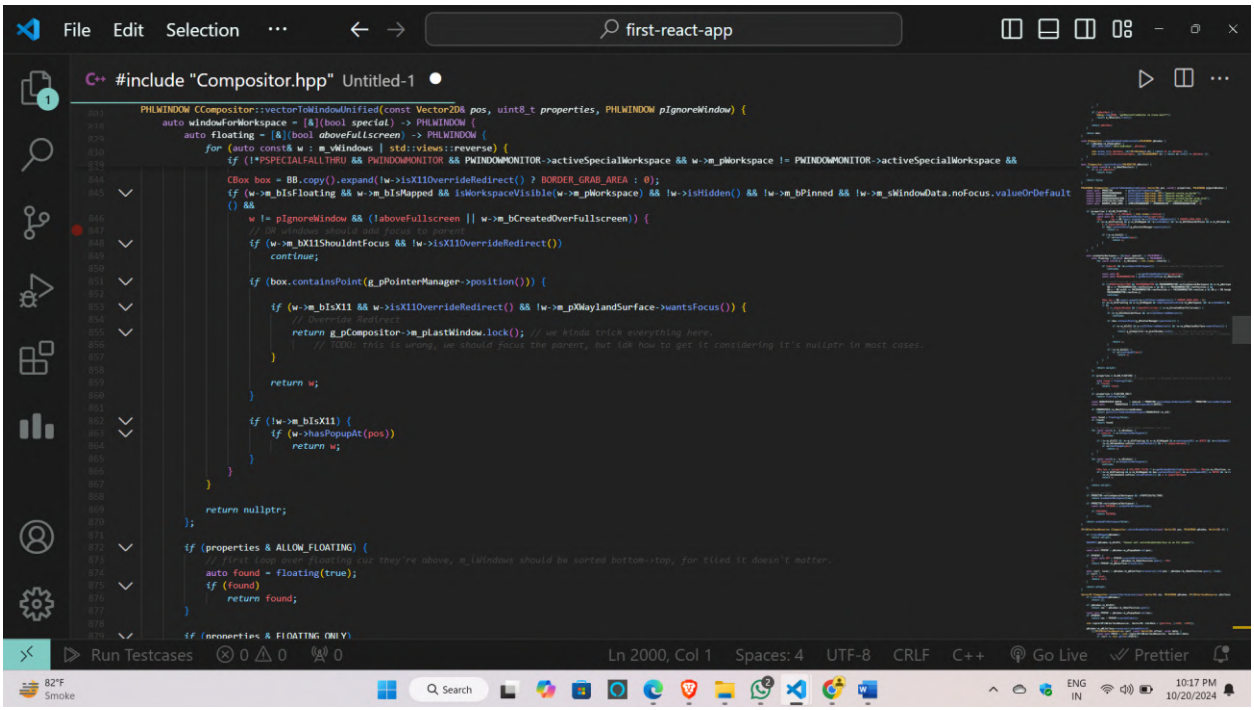

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::createLockFile() {
const auto PATH = m_szInstancePath + "/hyprland.lock";
std::ofstream ofs(PATH, std::ios::trunc);
ofs << m_iHyprlandPID << "\n" << m_szWLDisplaySocket << "\n";
ofs.close();
}
void CCompositor::removeLockFile() {
const auto PATH = m_szInstancePath + "/hyprland.lock";
if (std::filesystem::exists(PATH))
std::filesystem::remove(PATH);
}
void CCompositor::prepareFallbackOutput() {
SP<Aquamarine::IBackendImplementation> headless;
for (auto const& impl : m_pAqBackend->getImplementations()) {
if (impl->type() == Aquamarine::AQ_BACKEND_HEADLESS) {
headless = impl;
break;
}
}
if (!headless) {
Debug::log(WARN, "No headless in prepareFallbackOutput?!");
return;
}
headless->createOutput();
}
void CCompositor::startCompositor() {
signal(SIGPIPE, SIG_IGN);
if (
m_pAqBackend->hasSession() &&
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:15 PM
10/20/2024

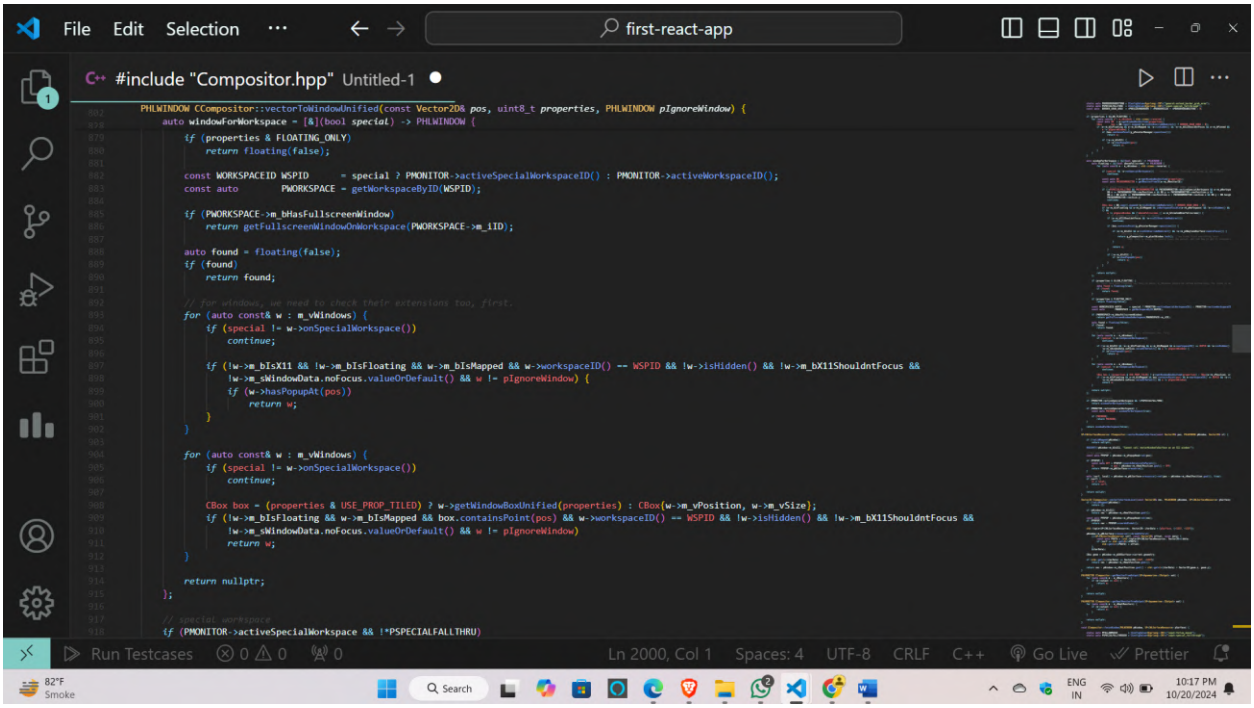

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::startCompositor() {
if (
m_pAqBackend->hasSession() &&
!envEnabled("HYPRLAND_NO_SD_VARS")) {
const auto CMD =
#ifdef USES_SYSTEMD
"systemctl --user import-environment DISPLAY WAYLAND_DISPLAY HYPRLAND_INSTANCE_SIGNATURE XDG_CURRENT_DESKTOP QT_QPA_PLATFORMTHEME PATH XDG_DATA_DIRS && hash "
"dbus-update-activation-environment 2>/dev/null && "
#endif
"dbus-update-activation-environment --systemd WAYLAND_DISPLAY XDG_CURRENT_DESKTOP HYPRLAND_INSTANCE_SIGNATURE QT_QPA_PLATFORMTHEME PATH XDG_DATA_DIRS";
g_pKeybindManager->spawn(CMD);
}
Debug::log(LOG, "Running on WAYLAND_DISPLAY: {}", m_szWLDisplaySocket);
prepareFallbackOutput();
g_pHyprRenderer->setCursorFromName("left_ptr");
#ifdef USES_SYSTEMD
if (Systemd::SdBooted() > 0) {
if (!envEnabled("HYPRLAND_NO_SD_NOTIFY"))
Systemd::SdNotify(0, "READY=1");
} else
Debug::log(LOG, "systemd integration is baked in but system itself is not booted à la systemd!");
#endif
createLockFile();
EMIT_HOOK_EVENT("ready", nullptr);
Debug::log(LOG, "Hyprland is ready, running the event loop!");
g_pEventLoopManager->enterLoop();
}
PHLMONITOR CCompositor::getMonitorFromID(const MONITORID& id) {
for (auto const& m : m_vMonitors) {
if (m->ID == id) {
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:16 PM
10/20/2024

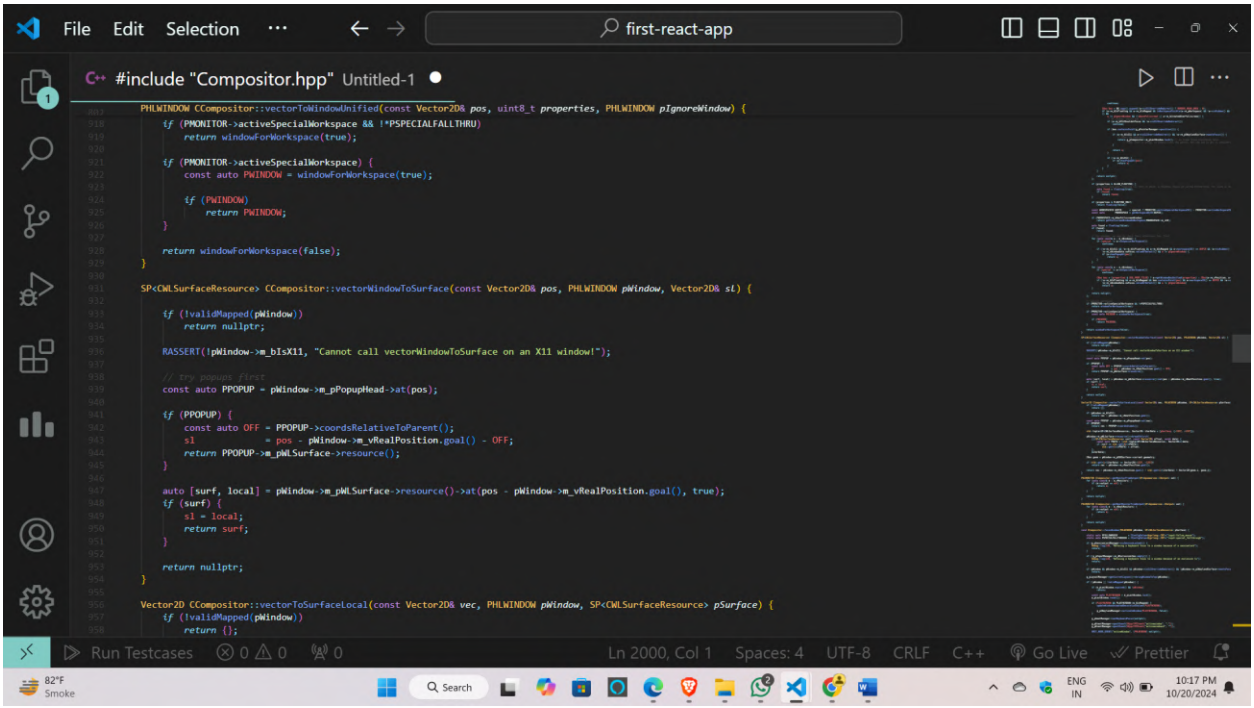

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLMONITOR CCompositor::getMonitorFromID(const MONITORID& id) {
for (auto const& m : m_vMonitors) {
if (m->ID == id) {
return m;
}
}
return nullptr;
}
PHLMONITOR CCompositor::getMonitorFromName(const std::string& name) {
for (auto const& m : m_vMonitors) {
if (m->szName == name) {
return m;
}
}
return nullptr;
}
PHLMONITOR CCompositor::getMonitorFromDesc(const std::string& desc) {
for (auto const& m : m_vMonitors) {
if (m->szDescription.starts_with(desc))
return m;
}
return nullptr;
}
PHLMONITOR CCompositor::getMonitorFromCursor() {
return getMonitorFromVector(g_pPointerManager->position());
}
PHLMONITOR CCompositor::getMonitorFromVector(const Vector2D& point) {
SP<CMonitor> mon;
for (auto const& m : m_vMonitors) {
if (CBox{m->vecPosition, m->vecSize}.containsPoint(point)) {
mon = m;
break;
}
}
if (!mon) {
float bestDistance = 0.f;
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:16 PM
10/20/2024

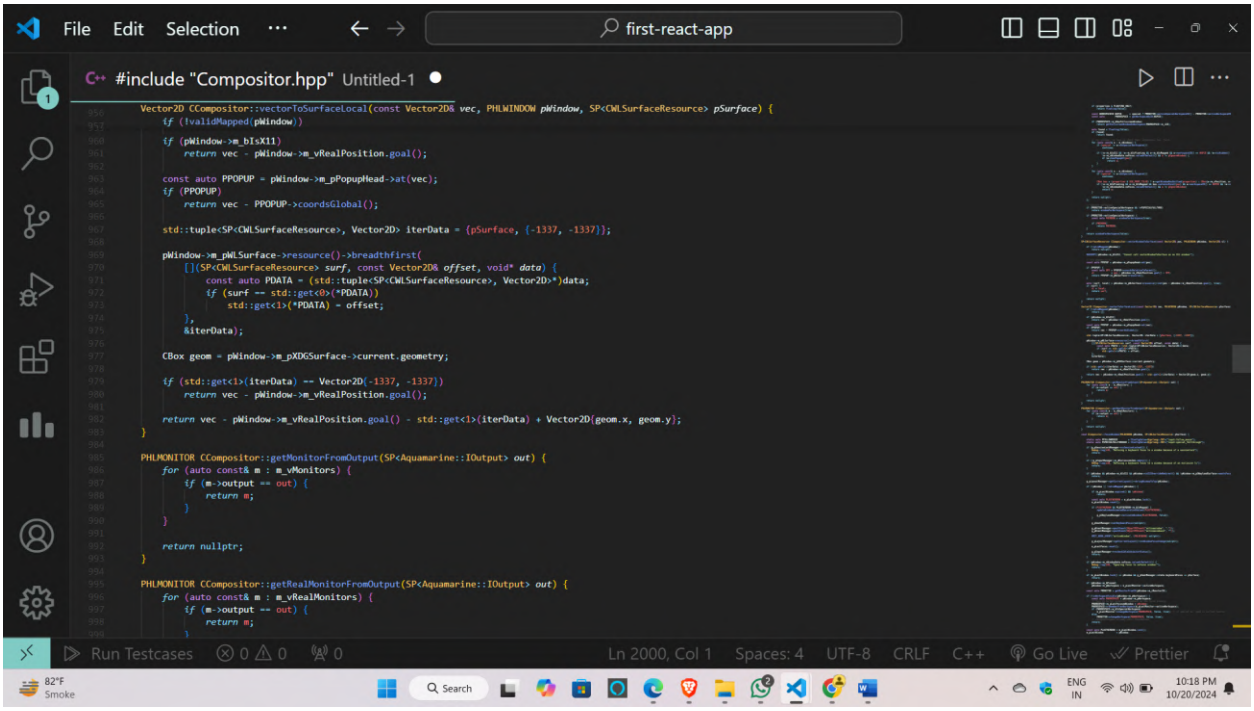

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLMONITOR CCompositor::getMonitorFromVector(const Vector2D& point) {
if (!mon) {
SP<CMonitor> pBestMon;
for (auto const& m : m_vMonitors) {
float dist = vecToRectDistanceSquared(point, m->vecPosition, m->vecPosition + m->vecSize);
if (dist < bestDistance || !pBestMon) {
bestDistance = dist;
pBestMon = m;
}
}
if (!pBestMon) { // ?????
Debug::log(WARN, "getMonitorFromVector no close mon???");
return m_vMonitors.front();
}
return pBestMon;
}
return mon;
}
void CCompositor::removeWindowFromVectorSafe(PHLWINDOW pWindow) {
if (!pWindow->m_bFadingOut) {
EMIT_HOOK_EVENT("destroyWindow", pWindow);
std::erase_if(m_vWindows, [&](SP<CWindow>& el) { return el == pWindow; });
std::erase_if(m_vWindowsFadingOut, [&](PHLWINDOWREF el) { return el.lock() == pWindow; });
}
}
bool CCompositor::monitorExists(PHLMONITOR pMonitor) {
for (auto const& m : m_vRealMonitors) {
if (m == pMonitor)
return true;
}
return false;
}
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:16 PM
10/20/2024

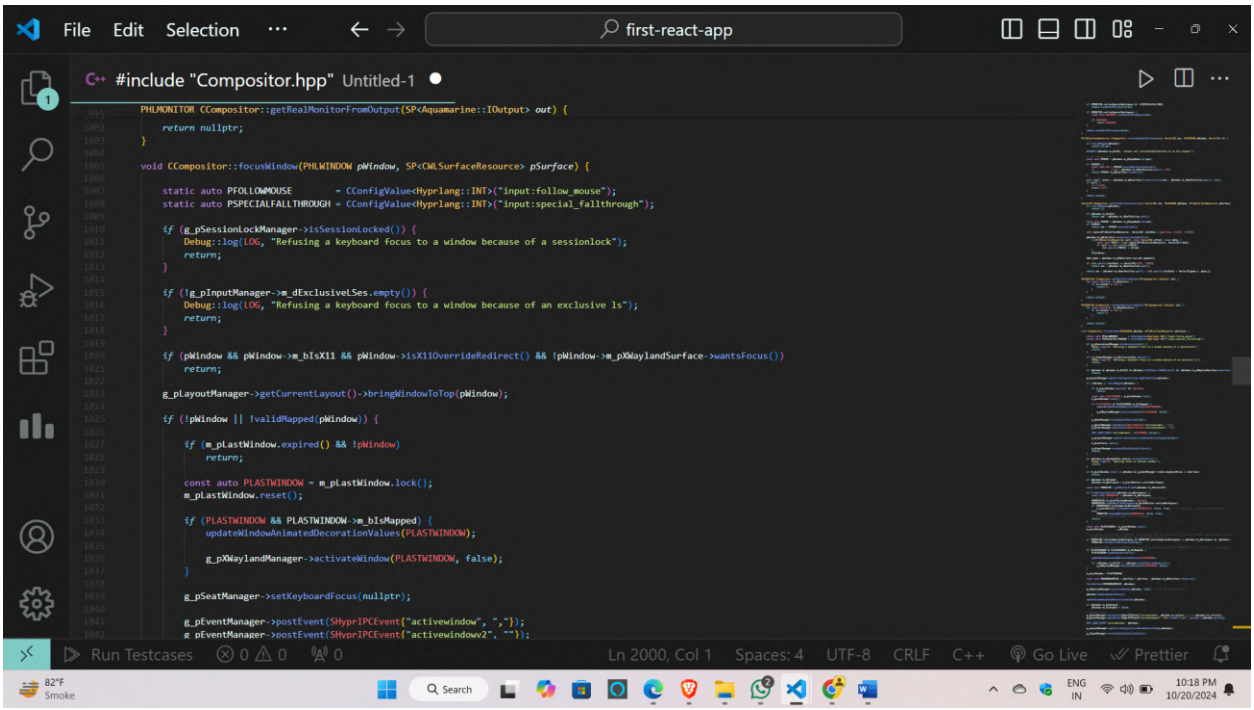

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::vectorToWindowUnified(const Vector2D& pos, uint8_t properties, PHLWINDOW pIgnoreWindow) {
const auto PMONITOR = getMonitorFromVector(pos);
static auto PRESIZEONBORDER = CConfigValue<Hyprlang::INT>("general:resize_on_border");
static auto PBORDERSIZE = CConfigValue<Hyprlang::INT>("general:border_size");
static auto PBORDERGRABEXTEND = CConfigValue<Hyprlang::INT>("general:extend_border_grab_area");
static auto PSPECIALFALLTHRU = CConfigValue<Hyprlang::INT>("input:special_fallthrough");
const auto BORDER_GRAB_AREA = *PRESIZEONBORDER ? *PBORDERSIZE + *PBORDERGRABEXTEND : 0;
// pinned windows on top of floating regardless
if (properties & ALLOW_FLOATING) {
for (auto const& w : m_vWindows | std::views::reverse) {
const auto BB = w->getWindowBoxUnified(properties);
CBox box = BB.copy().expand(!w->isX11OverrideRedirect() ? BORDER_GRAB_AREA : 0);
if (w->m_bIsFloating && w->m_bIsMapped && !w->isHidden() && !w->m_bX11ShouldntFocus && w->m_bPinned && !w->m_sWindowData.noFocus.valueOrDefault() &&
w != pIgnoreWindow) {
if (box.containsPoint(g_pPointerManager->position()))
return w;
if (!w->m_bIsX11) {
if (w->hasPopupAt(pos))
return w;
}
}
}
auto windowForWorkspace = [&](bool special) -> PHLWINDOW {
auto floating = [&](bool aboveFullscreen) -> PHLWINDOW {
for (auto const& w : m_vWindows | std::views::reverse) {
if (special && !w->onSpecialWorkspace()) // because special floating may creep up into regular
continue;
const auto BB = w->getWindowBoxUnified(properties);
const auto PWINDOWMONITOR = getMonitorFromID(w->m_iMonitorID);
// to avoid focusing windows behind special workspaces from other monitors
if (!*PSPECIALFALLTHRU && PWINDOWMONITOR && PWINDOWMONITOR->activeSpecialWorkspace && w->m_pWorkspace != PWINDOWMONITOR->activeSpecialWorkspace &&
BB.x >= PWINDOWMONITOR->vecPosition.x && BB.y >= PWINDOWMONITOR->vecPosition.y &&
BB.x + BB.width <= PWINDOWMONITOR->vecPosition.x + PWINDOWMONITOR->vecSize.x && BB.y + BB.height <= PWINDOWMONITOR->vecPosition.y +
PWINDOWMONITOR->vecSize.y)
continue;
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:16 PM
10/20/2024

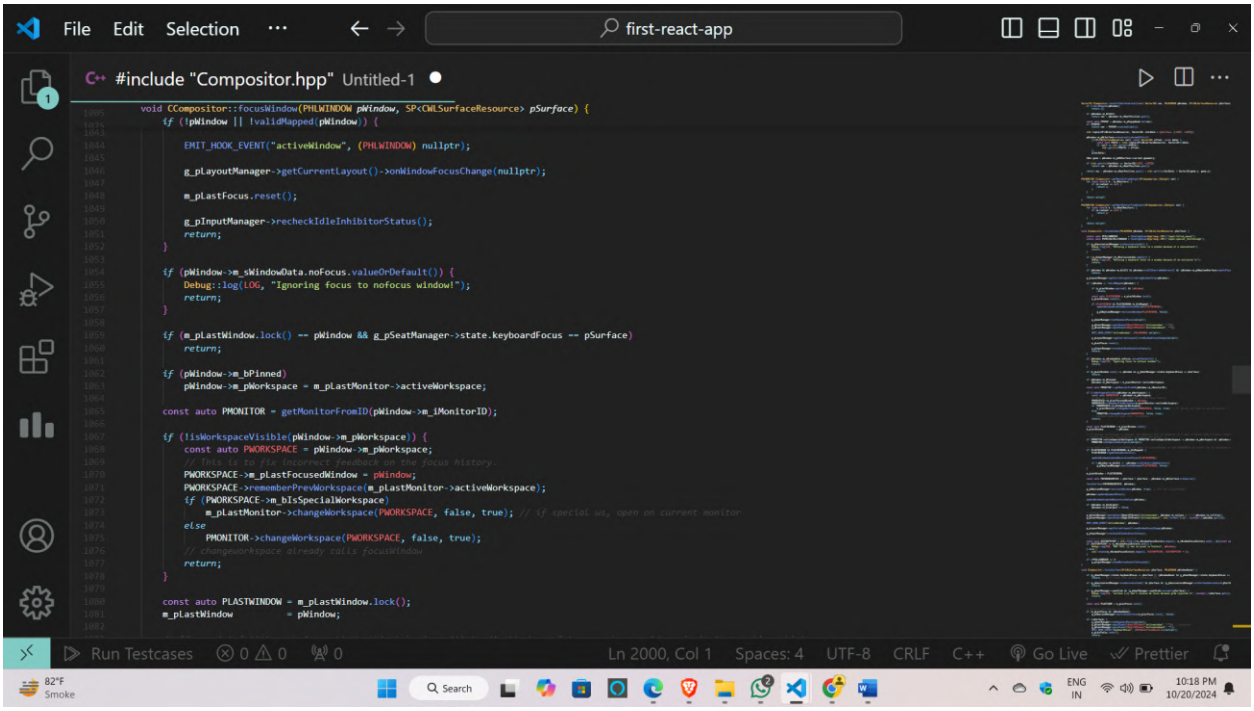

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::vectorToWindowUnified(const Vector2D& pos, uint8_t properties, PHLWINDOW pIgnoreWindow) {
auto windowForWorkspace = [&](bool special) -> PHLWINDOW {
auto floating = [&](bool aboveFullscreen) -> PHLWINDOW {
for (auto const& w : m_vWindows | std::views::reverse) {
if (!*PSPECIALFALLTHRU && PWINDOWMONITOR && PWINDOWMONITOR->activeSpecialWorkspace && w->m_pWorkspace != PWINDOWMONITOR->activeSpecialWorkspace &&
CBox box = BB.copy().expand(!w->isX11OverrideRedirect() ? BORDER_GRAB_AREA : 0);
if (w->m_bIsFloating && w->m_bIsMapped && isWorkspaceVisible(w->m_pWorkspace) && !w->isHidden() && !w->m_bPinned && !w->m_sWindowData.noFocus.valueOrDefault
() &&
w != pIgnoreWindow && (!aboveFullscreen || w->m_bCreatedOverFullscreen)) {
// OR windows should add focus to parent
if (w->m_bX11ShouldntFocus && !w->isX11OverrideRedirect())
continue;
if (box.containsPoint(g_pPointerManager->position())) {
if (w->m_bIsX11 && w->isX11OverrideRedirect() && !w->m_pXWaylandSurface->wantsFocus()) {
return g_pCompositor->m_pLastWindow.lock(); // we kinda trick everything here.
// TODO: this is wrong, we should focus the parent, but idk how to get it considering it's nullptr in most cases.
}
return w;
}
if (!w->m_bIsX11) {
if (w->hasPopupAt(pos))
return w;
}
}
}
return nullptr;
};
if (properties & ALLOW_FLOATING) {
// first loop over floating cuz they're above, m_lWindows should be sorted bottom->top, for tiled it doesn't matter.
auto found = floating(true);
if (found)
return found;
}
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:17 PM
10/20/2024

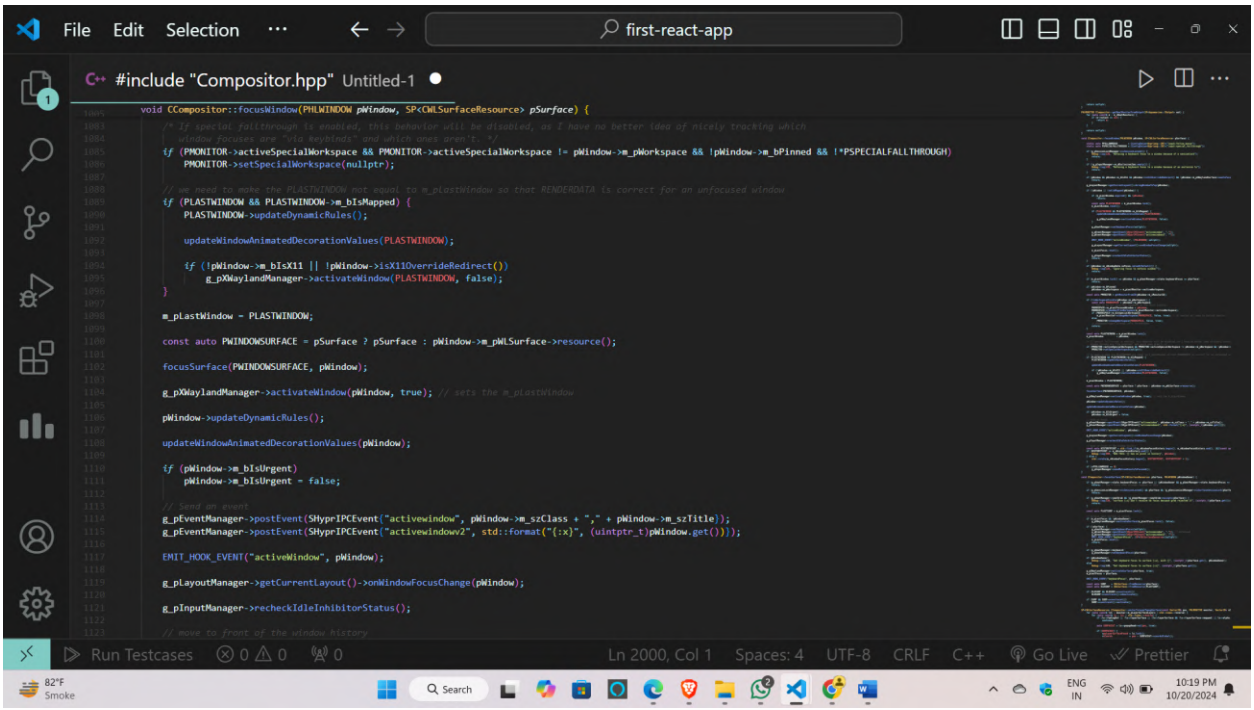

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::vectorToWindowUnified(const Vector2D& pos, uint8_t properties, PHLWINDOW pIgnoreWindow) {
auto windowForWorkspace = [&](bool special) -> PHLWINDOW {
if (properties & FLOATING_ONLY)
return floating(false);
const WORKSPACEID WSPID = special ? PMONITOR->activeSpecialWorkspaceID() : PMONITOR->activeWorkspaceID();
const auto PWORKSPACE = getWorkspaceByID(WSPID);
if (PWORKSPACE->m_bHasFullscreenWindow)
return getFullscreenWindowOnWorkspace(PWORKSPACE->m_iID);
auto found = floating(false);
if (found)
return found;
// for windows we need to check their extensions too, first.
for (auto const& w : m_vWindows) {
if (special != w->onSpecialWorkspace())
continue;
if (!w->m_bIsX11 && !w->m_bIsFloating && w->m_bIsMapped && w->workspaceID() == WSPID && !w->isHidden() && !w->m_bX11ShouldntFocus &&
!w->m_sWindowData.noFocus.valueOrDefault() && w != pIgnoreWindow) {
if (w->hasPopupAt(pos))
return w;
}
}
for (auto const& w : m_vWindows) {
if (special != w->onSpecialWorkspace())
continue;
CBox box = (properties & USE_PROP_TILED) ? w->getWindowBoxUnified(properties) : CBox(w->m_vPosition, w->m_vSize);
if (!w->m_bIsFloating && w->m_bIsMapped && box.containsPoint(pos) && w->workspaceID() == WSPID && !w->isHidden() && !w->m_bX11ShouldntFocus &&
!w->m_sWindowData.noFocus.valueOrDefault() && w != pIgnoreWindow)
return w;
}
return nullptr;
};
// special workspace
if (PMONITOR->activeSpecialWorkspace && !*PSPECIALFALLTHRU)
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:17 PM 10/20/2024

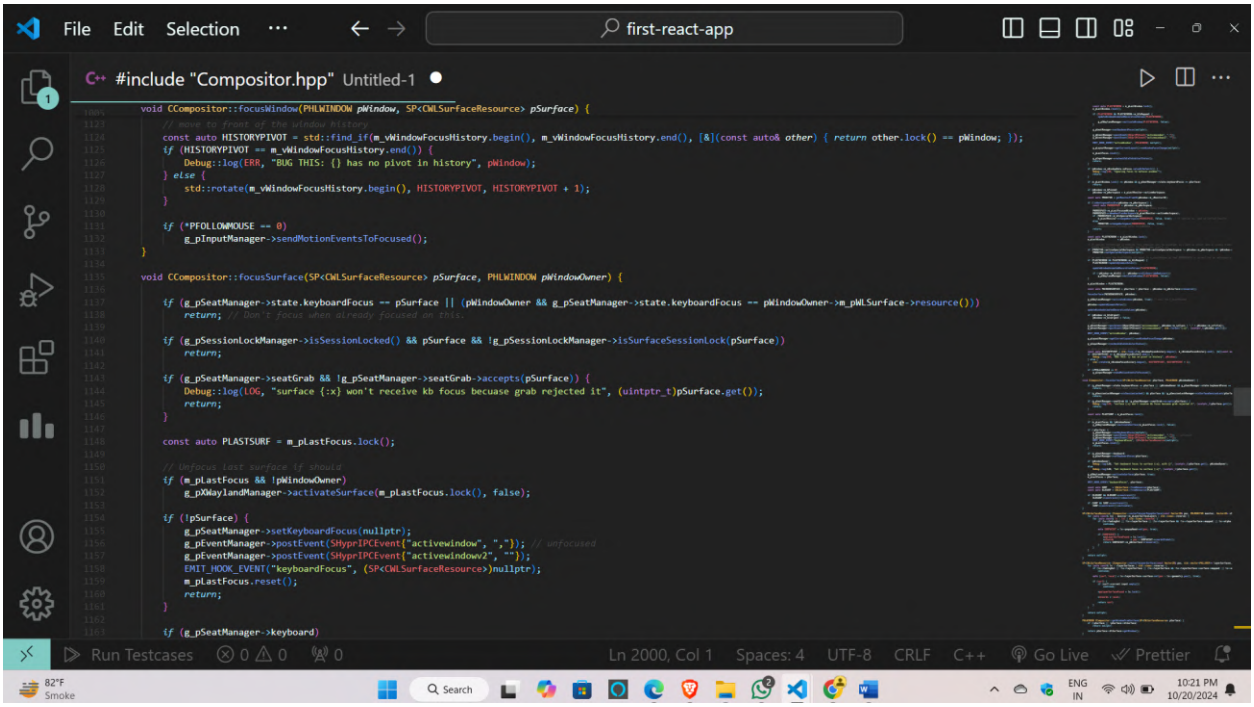

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::vectorToWindowUnified(const Vector2D& pos, uint8_t properties, PHLWINDOW pIgnoreWindow) {
if (PMONITOR->activeSpecialWorkspace && !*PSPECIALFALLTHRU)
return windowForWorkspace(true);
if (PMONITOR->activeSpecialWorkspace) {
const auto PWINDOW = windowForWorkspace(true);
if (PWINDOW)
return PWINDOW;
}
return windowForWorkspace(false);
}
SP<CWLSurfaceResource> CCompositor::vectorWindowToSurface(const Vector2D& pos, PHLWINDOW pWindow, Vector2D& sl) {
if (!validMapped(pWindow))
return nullptr;
RASSERT(!pWindow->m_bIsX11, "Cannot call vectorWindowToSurface on an X11 window!");
// try popups first
const auto PPOPUP = pWindow->m_pPopupHead->at(pos);
if (PPOPUP) {
const auto OFF = PPOPUP->coordsRelativeToParent();
sl = pos - pWindow->m_vRealPosition.goal() - OFF;
return PPOPUP->m_pWLSurface->resource();
}
auto [surf, local] = pWindow->m_pWLSurface->resource()->at(pos - pWindow->m_vRealPosition.goal(), true);
if (surf) {
sl = local;
return surf;
}
return nullptr;
}
Vector2D CCompositor::vectorToSurfaceLocal(const Vector2D& vec, PHLWINDOW pWindow, SP<CWLSurfaceResource> pSurface) {
if (!validMapped(pWindow))
return {};
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:17 PM 10/20/2024

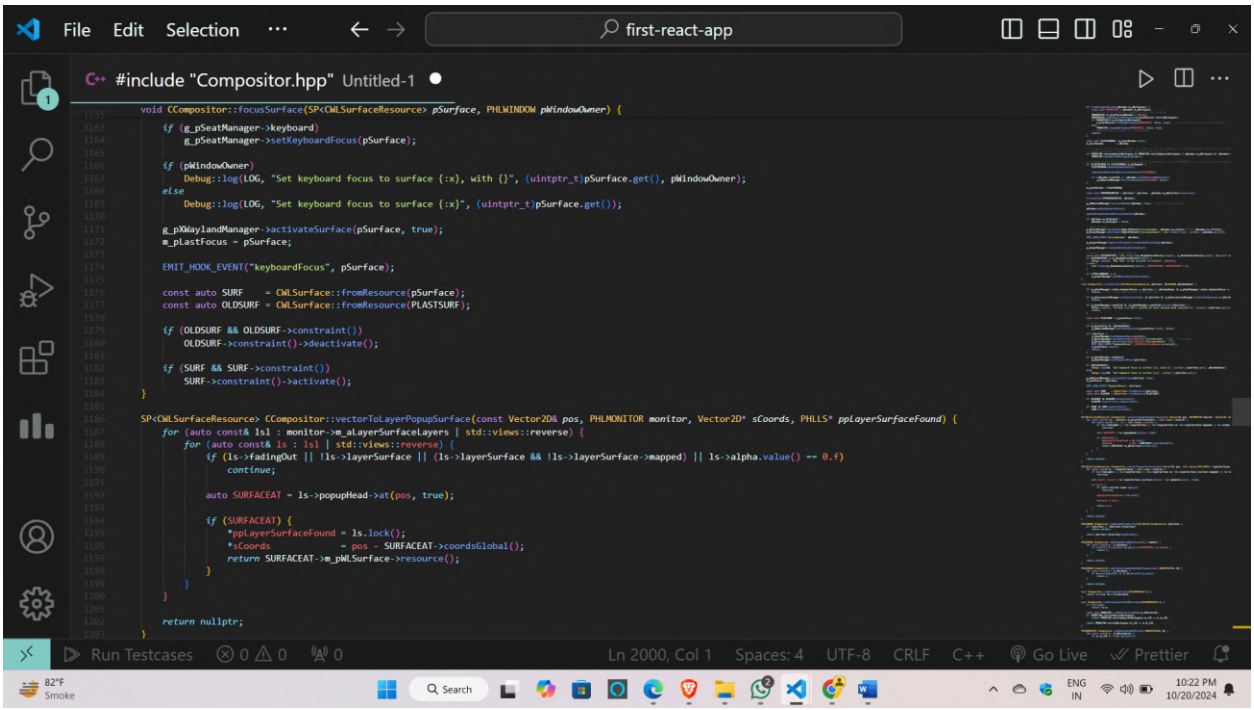

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
Vector2D CCompositor::vectorToSurfaceLocal(const Vector2D& vec, PHLWINDOW pWindow, SP<CWLSurfaceResource> pSurface) {
if (!validMapped(pWindow))
if (pWindow->m_bIsX11)
return vec - pWindow->m_vRealPosition.goal();
const auto PPOPUP = pWindow->m_pPopupHead->at(vec);
if (PPOPUP)
return vec - PPOPUP->coordsGlobal();
std::tuple<SP<CWLSurfaceResource>, Vector2D> iterData = {pSurface, {-1337, -1337}};
pWindow->m_pWLSurface->resource()->breadthfirst(
[](SP<CWLSurfaceResource> surf, const Vector2D& offset, void* data) {
const auto PDATA = (std::tuple<SP<CWLSurfaceResource>, Vector2D>*)data;
if (surf == std::get<0>(*PDATA))
std::get<1>(*PDATA) = offset;
},
&iterData);
CBox geom = pWindow->m_pXDGSurface->current.geometry;
if (std::get<1>(iterData) == Vector2D(-1337, -1337))
return vec - pWindow->m_vRealPosition.goal();
return vec - pWindow->m_vRealPosition.goal() - std::get<1>(iterData) + Vector2D(geom.x, geom.y);
}
PHLMONITOR CCompositor::getMonitorFromOutput(SP<Aquamarine::IOutput> out) {
for (auto const& m : m_vMonitors) {
if (m->output == out) {
return m;
}
}
return nullptr;
}
PHLMONITOR CCompositor::getRealMonitorFromOutput(SP<Aquamarine::IOutput> out) {
for (auto const& m : m_vRealMonitors) {
if (m->output == out) {
return m;
}
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:18 PM
10/20/2024

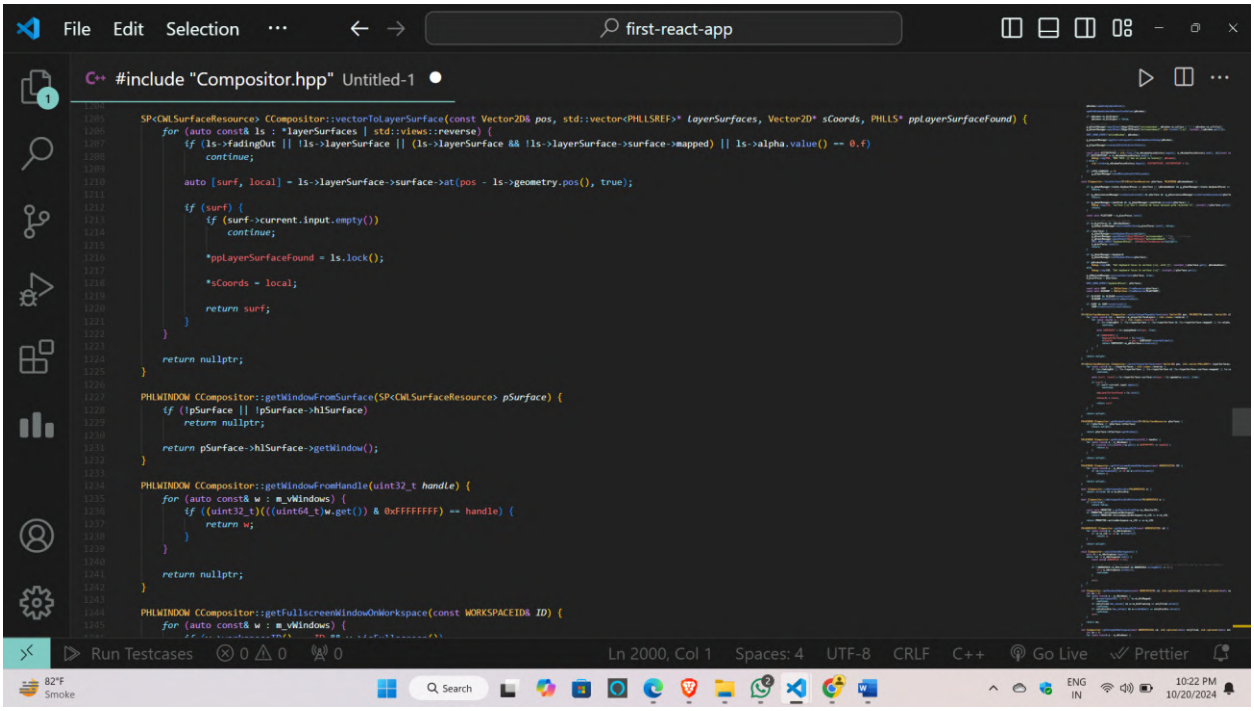

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLMONITOR CCompositor::getRealMonitorFromOutput(SP<Aquamarine::IOutput> out) {
return nullptr;
}
void CCompositor::focusWindow(PHLWINDOW pWindow, SP<CWLSurfaceResource> pSurface) {
static auto PFOLLOWMOUSE = CConfigValue<Hyprlang::INT>("input:follow_mouse");
static auto PSPECIALFALLTHROUGH = CConfigValue<Hyprlang::INT>("input:special_fallthrough");
if (g_pSessionLockManager->isSessionLocked()) {
Debug::log(LOG, "Refusing a keyboard focus to a window because of a sessionlock");
return;
}
if (!g_pInputManager->m_dExclusiveLSes.empty()) {
Debug::log(LOG, "Refusing a keyboard focus to a window because of an exclusive ls");
return;
}
if (pWindow && pWindow->m_bIsX11 && pWindow->isX11OverrideRedirect() && !pWindow->m_pXWaylandSurface->wantsFocus())
return;
g_pLayoutManager->getCurrentLayout()->bringWindowToTop(pWindow);
if (!pWindow || !validMapped(pWindow)) {
if (m_pLastWindow.expired() && !pWindow)
return;
const auto PLASTWINDOW = m_pLastWindow.lock();
m_pLastWindow.reset();
if (PLASTWINDOW && PLASTWINDOW->m_bIsMapped) {
updateWindowAnimatedDecorationValues(PLASTWINDOW);
g_pXWaylandManager->activateWindow(PLASTWINDOW, false);
}
g_pSeatManager->setKeyboardFocus(nullptr);
g_pEventManager->postEvent(SHyprIPCEvent{"activewindow", ","});
g_pEventManager->postEvent(SHyprIPCEvent{"activewindowv2", ""});
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:18 PM
10/20/2024

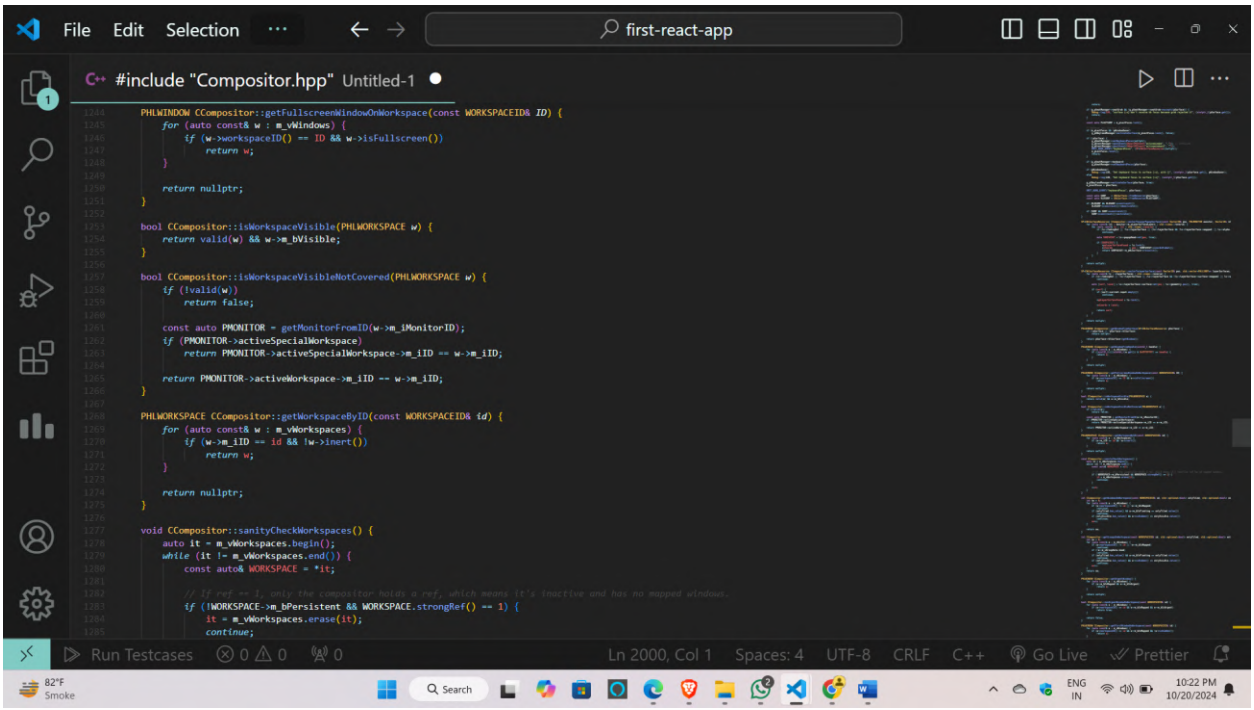

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::focusWindow(PHLWINDOW pWindow, SP<CWLSurfaceResource> pSurface) {
if (!pWindow || !validMapped(pWindow)) {
EMIT_HOOK_EVENT("activeWindow", (PHLWINDOW) nullptr);
g_pLayoutManager->getCurrentLayout()->onWindowFocusChange(nullptr);
m_pLastFocus.reset();
g_pInputManager->recheckIdleInhibitorStatus();
return;
}
if (pWindow->m_sWindowData.noFocus.valueOrDefault()) {
Debug::log(LOG, "Ignoring focus to nofocus window!");
return;
}
if (m_pLastWindow.lock() == pWindow && g_pSeatManager->state.keyboardFocus == pSurface)
return;
if (pWindow->m_bPinned)
pWindow->m_pWorkspace = m_pLastMonitor->activeWorkspace;
const auto PMONITOR = getMonitorFromID(pWindow->m_iMonitorID);
if (!isWorkspaceVisible(pWindow->m_pWorkspace)) {
const auto PWORKSPACE = pWindow->m_pWorkspace;
// This is to fix incorrect feedback on the focus history.
PWORKSPACE->m_pLastFocusedWindow = pWindow;
PWORKSPACE->rememberPrevWorkspace(m_pLastMonitor->activeWorkspace);
if (PWORKSPACE->m_bIsSpecialWorkspace)
m_pLastMonitor->changeWorkspace(PWORKSPACE, false, true); // if special ws, open on current monitor
else
PMONITOR->changeWorkspace(PWORKSPACE, false, true);
// changeWorkspace already calls focusWindow
return;
}
const auto PLASTWINDOW = m_pLastWindow.lock();
m_pLastWindow = pWindow;
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:18 PM 10/20/2024

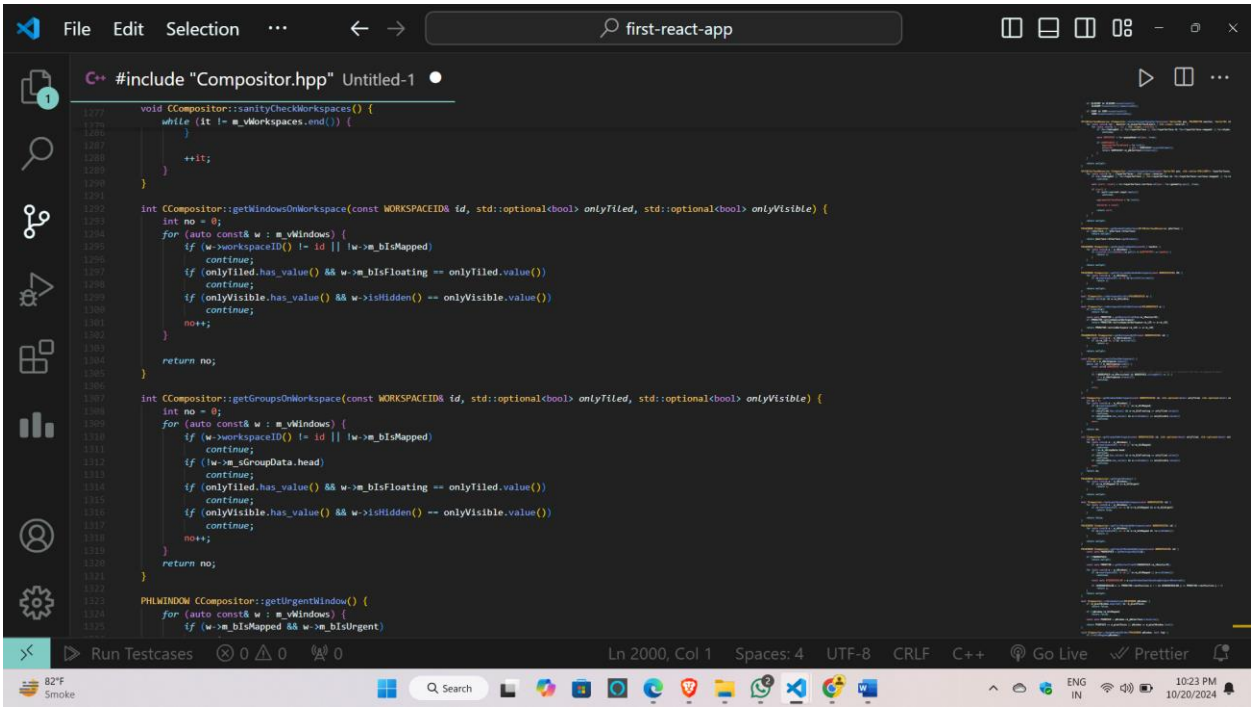

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::focusWindow(PHLWINDOW pWindow, SP<CWLSurfaceResource> pSurface) {
/* If special fallthrough is enabled, this behavior will be disabled, as I have no better idea of nicely tracking which
window focused from "old workspace" and which from event. */
if (PMONITOR->activeSpecialWorkspace && PMONITOR->activeSpecialWorkspace != pWindow->m_pWorkspace && !pWindow->m_bPinned && !*PSPECIALFALLTHROUGH)
PMONITOR->setSpecialWorkspace(nullptr);
// we need to make the PLASTWINDOW not equal to m_pLastWindow so that RENDERDATA is correct for an unfocused window
if (PLASTWINDOW && PLASTWINDOW->m_bIsMapped) {
PLASTWINDOW->updateDynamicRules();
updateWindowAnimatedDecorationValues(PLASTWINDOW);
if (!pWindow->m_bIsX11 || !pWindow->isX11OverrideRedirect())
g_pXWaylandManager->activateWindow(PLASTWINDOW, false);
}
m_pLastWindow = PLASTWINDOW;
const auto PWINDOWSURFACE = pSurface ? pSurface : pWindow->m_pWLSurface->resource();
focusSurface(PWINDOWSURFACE, pWindow);
g_pXWaylandManager->activateWindow(pWindow, true); // sets the m_pLastWindow
pWindow->updateDynamicRules();
updateWindowAnimatedDecorationValues(pWindow);
if (pWindow->m_bIsUrgent)
pWindow->m_bIsUrgent = false;
// Send an event
g_pEventManager->postEvent(SHyprIPCEvent{"activewindow", pWindow->m_szClass + "," + pWindow->m_szTitle});
g_pEventManager->postEvent(SHyprIPCEvent{"activewindowv2", std::format("{:x}", (uintptr_t)pWindow.get())});
EMIT_HOOK_EVENT("activeWindow", pWindow);
g_pLayoutManager->getCurrentLayout()->onWindowFocusChange(pWindow);
g_pInputManager->recheckIdleInhibitorStatus();
// move to front of the window history
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:19 PM 10/20/2024

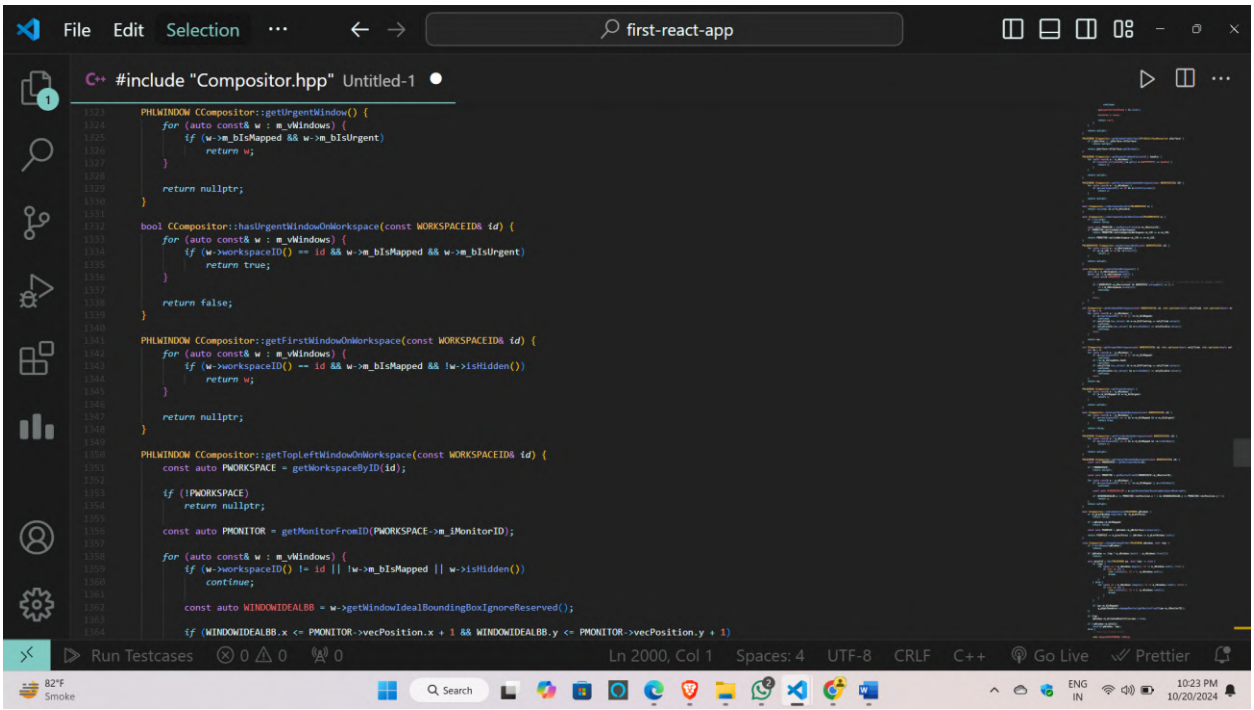

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::focusWindow(PHLWINDOW pWindow, SP<CWLSurfaceResource> pSurface) {
// move to front of the window history
const auto HISTORYPIVOT = std::find_if(m_vWindowFocusHistory.begin(), m_vWindowFocusHistory.end(), [&](const auto& other) { return other.lock() == pWindow; });
if (HISTORYPIVOT == m_vWindowFocusHistory.end()) {
Debug::log(ERR, "BUG THIS: {} has no pivot in history", pWindow);
} else {
std::rotate(m_vWindowFocusHistory.begin(), HISTORYPIVOT, HISTORYPIVOT + 1);
}
if (*PFOLLOWMOUSE == 0)
g_pInputManager->sendMotionEventsToFocused();
}
void CCompositor::focusSurface(SP<CWLSurfaceResource> pSurface, PHLWINDOW pWindowOwner) {
if (g_pSeatManager->state.keyboardFocus == pSurface || (pWindowOwner && g_pSeatManager->state.keyboardFocus == pWindowOwner->m_pWLSurface->resource()))
return; // don't focus when already focused on this.
if (g_pSessionLockManager->isSessionLocked() && pSurface && !g_pSessionLockManager->isSurfaceSessionLock(pSurface))
return;
if (g_pSeatManager->seatGrab && !g_pSeatManager->seatGrab->accepts(pSurface)) {
Debug::log(LOG, "surface {:x} won't receive kb focus becuase grab rejected it", (uintptr_t)pSurface.get());
return;
}
const auto PLASTSURF = m_pLastFocus.lock();
// Unfocus last surface if should
if (m_pLastFocus && !pWindowOwner)
g_pXWaylandManager->activateSurface(m_pLastFocus.lock(), false);
if (!pSurface) {
g_pSeatManager->setKeyboardFocus(nullptr);
g_pEventManager->postEvent(SHyprIPCEvent{"activewindow", ","}); // unfocused
g_pEventManager->postEvent(SHyprIPCEvent{"activewindowv2", ""});
EMIT_HOOK_EVENT("keyboardFocus", (SP<CWLSurfaceResource>)nullptr);
m_pLastFocus.reset();
return;
}
if (g_pSeatManager->keyboard)
Run Testcases ⊗ 0 ⚠ 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:21 PM 10/20/2024

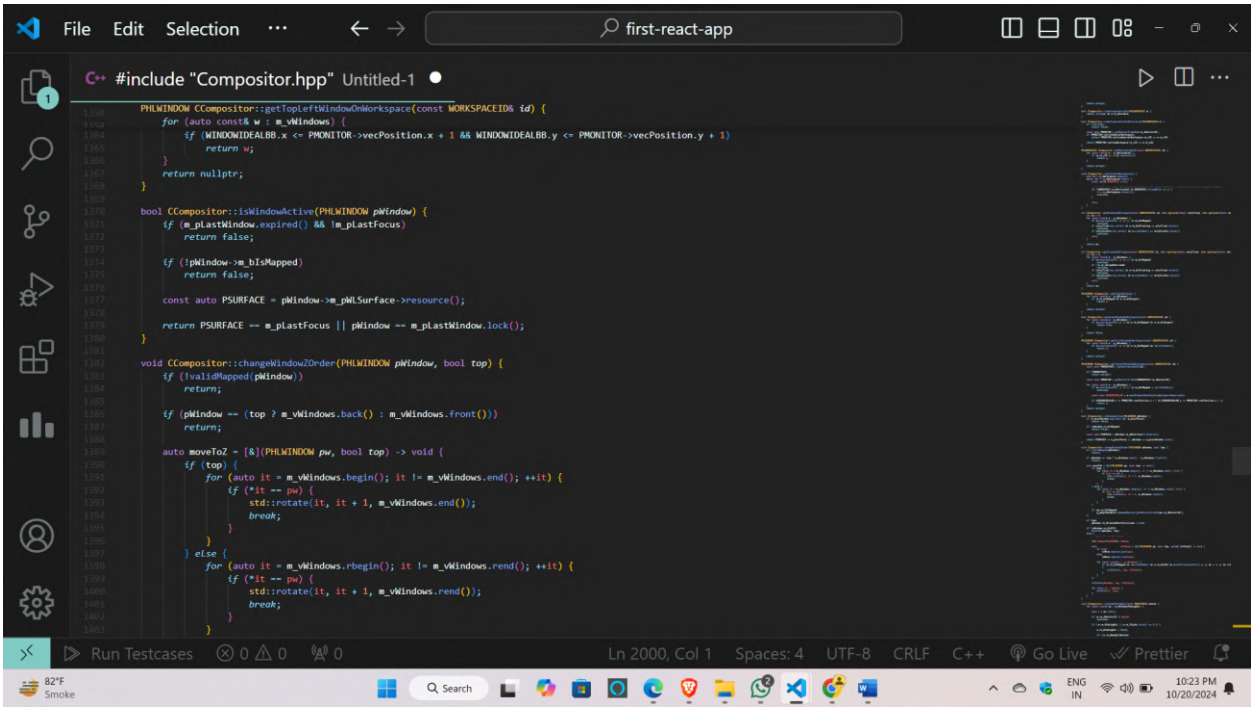

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::focusSurface(SP<CWLSurfaceResource> pSurface, PHLWINDOW pWindowOwner) {
if (g_pSeatManager->keyboard)
g_pSeatManager->setKeyboardFocus(pSurface);
if (pWindowOwner)
Debug::log(LOG, "Set keyboard focus to surface {:x}, with {}", (uintptr_t)pSurface.get(), pWindowOwner);
else
Debug::log(LOG, "Set keyboard focus to surface {:x}", (uintptr_t)pSurface.get());
g_pXWaylandManager->activateSurface(pSurface, true);
m_pLastFocus = pSurface;
EMIT_HOOK_EVENT("keyboardFocus", pSurface);
const auto SURF = CWLSurface::fromResource(pSurface);
const auto OLDSURF = CWLSurface::fromResource(PLASTSURF);
if (OLDSURF && OLDSURF->constraint())
OLDSURF->constraint()->deactivate();
if (SURF && SURF->constraint())
SURF->constraint()->activate();
}
SP<CWLSurfaceResource> CCompositor::vectorToLayerPopupSurface(const Vector2D& pos, PHLMONITOR monitor, Vector2D* sCoords, PHLLS* ppLayerSurfaceFound) {
for (auto const& lsl : monitor->m_aLayerSurfaceLayers | std::views::reverse) {
for (auto const& ls : lsl | std::views::reverse) {
if (ls->fadingOut || !ls->layerSurface || (ls->layerSurface && !ls->layerSurface->mapped) || ls->alpha.value() == 0.f)
continue;
auto SURFACEAT = ls->popupHead->at(pos, true);
if (SURFACEAT) {
*ppLayerSurfaceFound = ls.lock();
*sCoords = pos - SURFACEAT->coordsGlobal();
return SURFACEAT->m_pWLSurface->resource();
}
}
}
return nullptr;
}
Run Testcases ⊗ 0 ⚠ 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:22 PM 10/20/2024

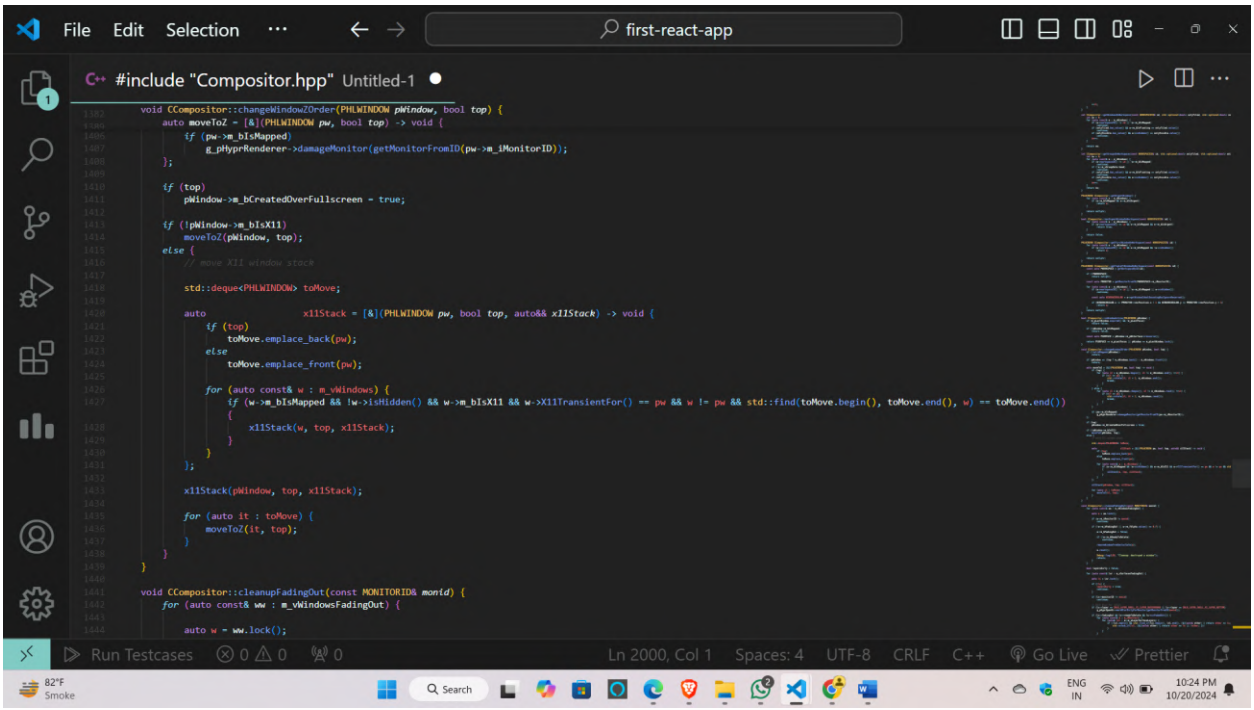

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
SP<CWLSurfaceResource> CCompositor::vectorToLayerSurface(const Vector2D& pos, std::vector<PHLLSREF>* layerSurfaces, Vector2D* sCoords, PHLLS* ppLayerSurfaceFound) {
for (auto const& ls : *layerSurfaces | std::views::reverse) {
if (ls->fadingOut || !ls->layerSurface || (ls->layerSurface && !ls->layerSurface->surface->mapped) || ls->alpha.value() == 0.f)
continue;
auto [surf, local] = ls->layerSurface->surface->at(pos - ls->geometry.pos(), true);
if (surf) {
if (surf->current.input.empty())
continue;
*ppLayerSurfaceFound = ls.lock();
*sCoords = local;
return surf;
}
}
return nullptr;
}
PHLWINDOW CCompositor::getWindowFromSurface(SP<CWLSurfaceResource> pSurface) {
if (!pSurface || !pSurface->hlSurface)
return nullptr;
return pSurface->hlSurface->getWindow();
}
PHLWINDOW CCompositor::getWindowFromHandle(uint32_t handle) {
for (auto const& w : m_vWindows) {
if ((uint32_t)((uint64_t)w.get()) & 0xFFFFFFFF) == handle) {
return w;
}
}
return nullptr;
}
PHLWINDOW CCompositor::getFullscreenWindowOnWorkspace(const WORKSPACEID& ID) {
for (auto const& w : m_vWindows) {
Run Testcases 0 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:22 PM 10/20/2024

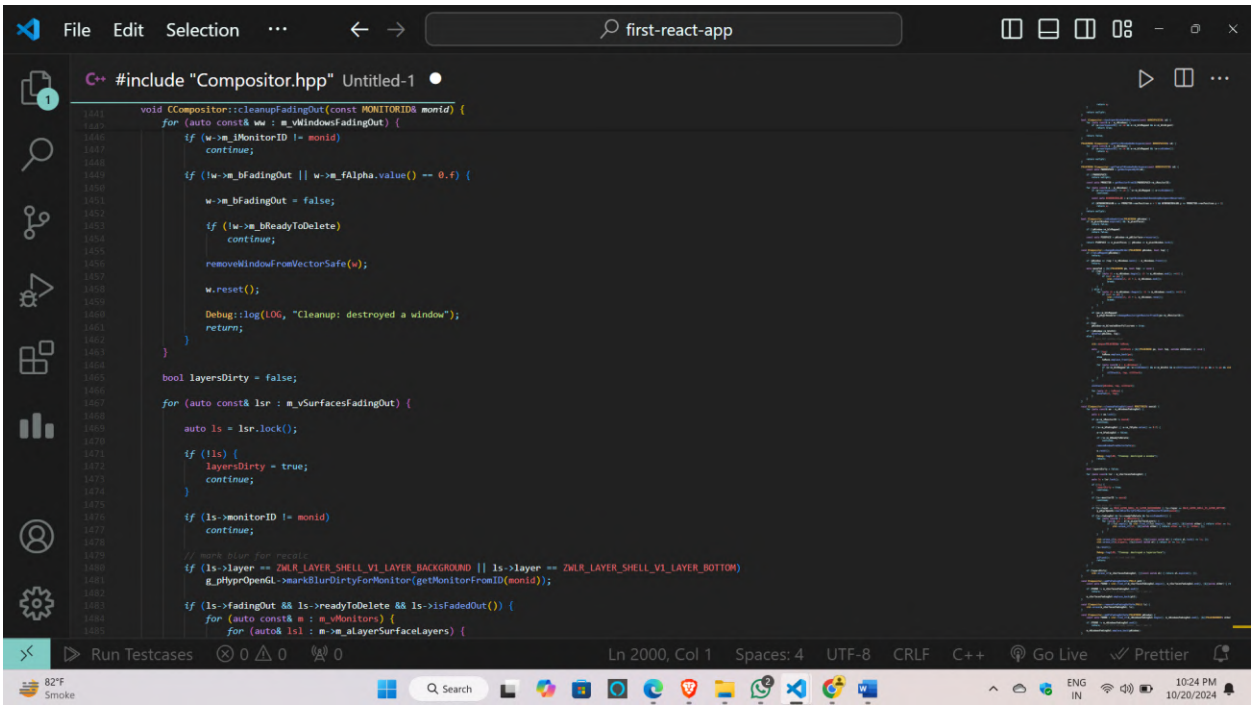

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::getFullscreenWindowOnWorkspace(const WORKSPACEID& ID) {
for (auto const& w : m_vWindows) {
if (w->workspaceID() == ID && w->isFullscreen())
return w;
}
return nullptr;
}
bool CCompositor::isWorkspaceVisible(PHLWORKSPACE w) {
return valid(w) && w->m_bVisible;
}
bool CCompositor::isWorkspaceVisibleNotCovered(PHLWORKSPACE w) {
if (!valid(w))
return false;
const auto PMONITOR = getMonitorFromID(w->m_iMonitorID);
if (PMONITOR->activeSpecialWorkspace)
return PMONITOR->activeSpecialWorkspace->m_iID == w->m_iID;
return PMONITOR->activeWorkspace->m_iID == w->m_iID;
}
PHLWORKSPACE CCompositor::getWorkspaceByID(const WORKSPACEID& id) {
for (auto const& w : m_vWorkspaces) {
if (w->m_iID == id && !w->inert())
return w;
}
return nullptr;
}
void CCompositor::sanityCheckWorkspaces() {
auto it = m_vWorkspaces.begin();
while (it != m_vWorkspaces.end()) {
const auto& WORKSPACE = *it;
// If ref == 1, only the compositor holds a ref, which means it's inactive and has no mapped windows.
if (!WORKSPACE->m_bPersistent && WORKSPACE.strongRef() == 1) {
it = m_vWorkspaces.erase(it);
continue;
Run Testcases 0 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:22 PM 10/20/2024

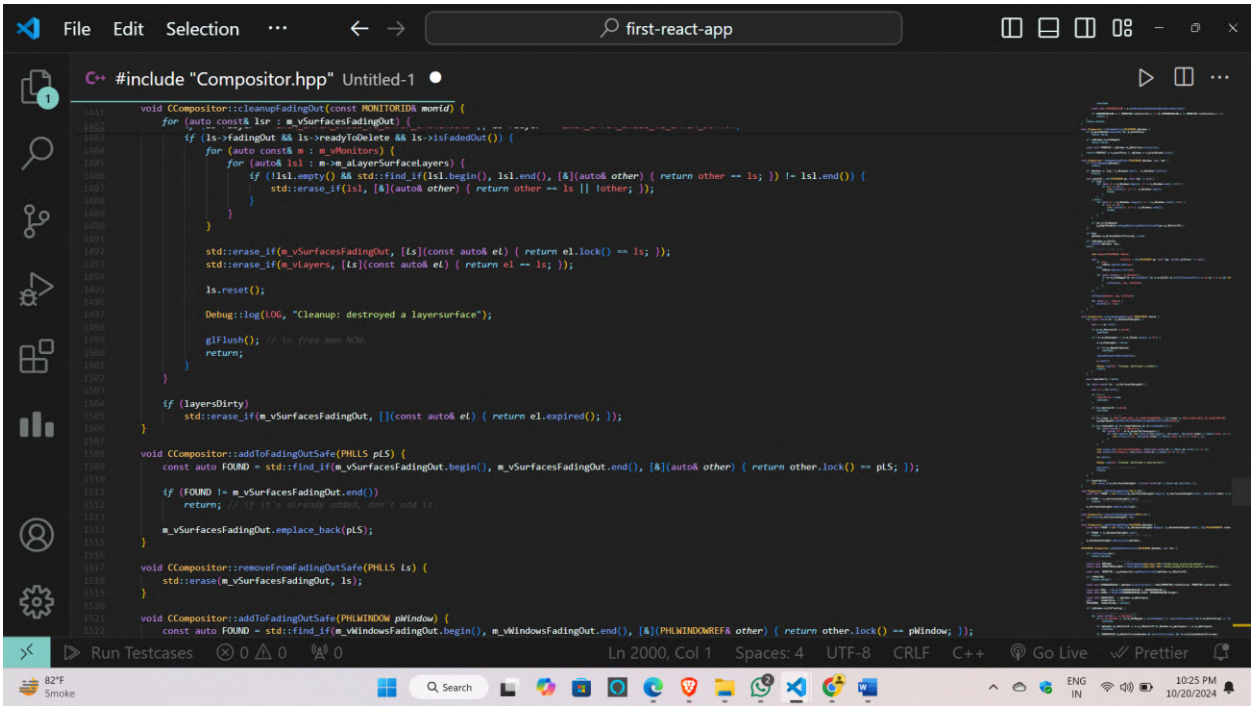

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::sanityCheckWorkspaces() {
while (it != m_vWorkspaces.end()) {
++it;
}
}
int CCompositor::getWindowsOnWorkspace(const WORKSPACEID& id, std::optional<bool> onlyTiled, std::optional<bool> onlyVisible) {
int no = 0;
for (auto const& w : m_vWindows) {
if (w->workspaceID() != id || !w->m_bIsMapped)
continue;
if (onlyTiled.has_value() && w->m_bIsFloating == onlyTiled.value())
continue;
if (onlyVisible.has_value() && w->isHidden() == onlyVisible.value())
continue;
no++;
}
return no;
}
int CCompositor::getGroupsOnWorkspace(const WORKSPACEID& id, std::optional<bool> onlyTiled, std::optional<bool> onlyVisible) {
int no = 0;
for (auto const& w : m_vWindows) {
if (w->workspaceID() != id || !w->m_bIsMapped)
continue;
if (!w->m_sGroupData.head)
continue;
if (onlyTiled.has_value() && w->m_bIsFloating == onlyTiled.value())
continue;
if (onlyVisible.has_value() && w->isHidden() == onlyVisible.value())
continue;
no++;
}
return no;
}
PHLWINDOW CCompositor::getUrgentWindow() {
for (auto const& w : m_vWindows) {
if (w->m_bIsMapped && w->m_bIsUrgent)
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:23 PM
10/20/2024

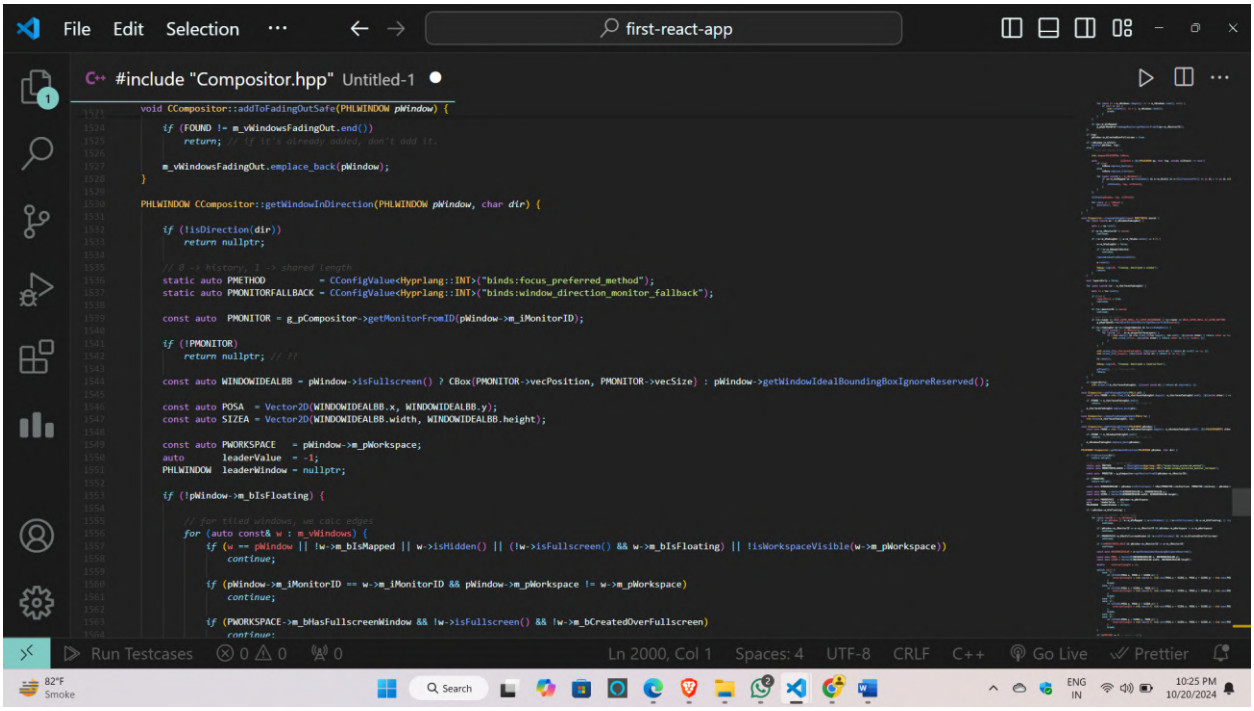

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::getUrgentWindow() {
for (auto const& w : m_vWindows) {
if (w->m_bIsMapped && w->m_bIsUrgent)
return w;
}
return nullptr;
}
bool CCompositor::hasUrgentWindowOnWorkspace(const WORKSPACEID& id) {
for (auto const& w : m_vWindows) {
if (w->workspaceID() == id && w->m_bIsMapped && w->m_bIsUrgent)
return true;
}
return false;
}
PHLWINDOW CCompositor::getFirstWindowOnWorkspace(const WORKSPACEID& id) {
for (auto const& w : m_vWindows) {
if (w->workspaceID() == id && w->m_bIsMapped && !w->isHidden())
return w;
}
return nullptr;
}
PHLWINDOW CCompositor::getTopLeftWindowOnWorkspace(const WORKSPACEID& id) {
const auto PWORKSPACE = getWorkspaceByID(id);
if (!PWORKSPACE)
return nullptr;
const auto PMONITOR = getMonitorFromID(PWORKSPACE->m_iMonitorID);
for (auto const& w : m_vWindows) {
if (w->workspaceID() != id || !w->m_bIsMapped || w->isHidden())
continue;
const auto WINDOWIDEALBB = w->getWindowIdealBoundingBoxIgnoreReserved();
if (WINDOWIDEALBB.x <= PMONITOR->vecPosition.x + 1 && WINDOWIDEALBB.y <= PMONITOR->vecPosition.y + 1)
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:23 PM
10/20/2024

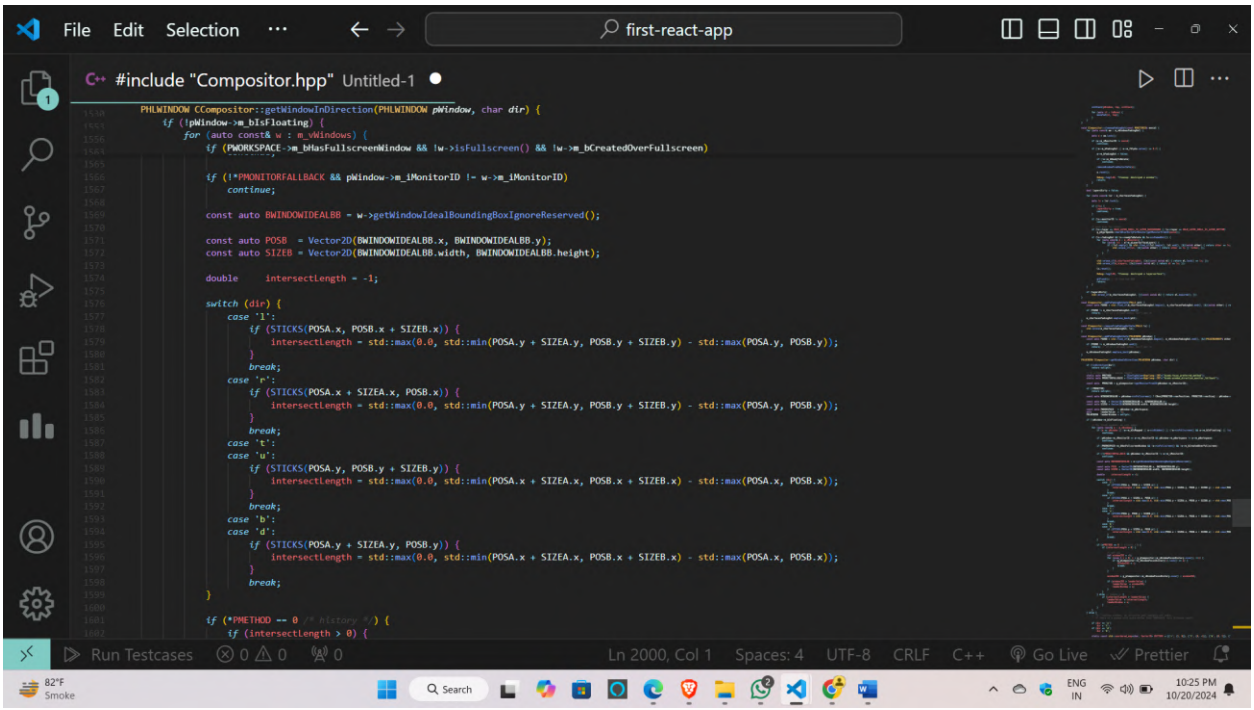

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::getTopLeftWindowOnWorkspace(const WORKSPACEID& id) {
for (auto const& w : m_vWindows) {
if (WINDOWIDEALBB.x <= PMONITOR->vecPosition.x + 1 && WINDOWIDEALBB.y <= PMONITOR->vecPosition.y + 1)
return w;
}
return nullptr;
}
bool CCompositor::isWindowActive(PHLWINDOW pWindow) {
if (m_pLastWindow.expired() && !m_pLastFocus)
return false;
if (!pWindow->m_bIsMapped)
return false;
const auto PSURFACE = pWindow->m_pWLSurface->resource();
return PSURFACE == m_pLastFocus || pWindow == m_pLastWindow.lock();
}
void CCompositor::changeWindowZOrder(PHLWINDOW pWindow, bool top) {
if (!validMapped(pWindow))
return;
if (pWindow == (top ? m_vWindows.back() : m_vWindows.front()))
return;
auto moveToZ = [&](PHLWINDOW pw, bool top) -> void {
if (top) {
for (auto it = m_vWindows.begin(); it != m_vWindows.end(); ++it) {
if (*it == pw) {
std::rotate(it, it + 1, m_vWindows.end());
break;
}
}
} else {
for (auto it = m_vWindows.rbegin(); it != m_vWindows.rend(); ++it) {
if (*it == pw) {
std::rotate(it, it + 1, m_vWindows.rend());
break;
}
}
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:23 PM 10/20/2024

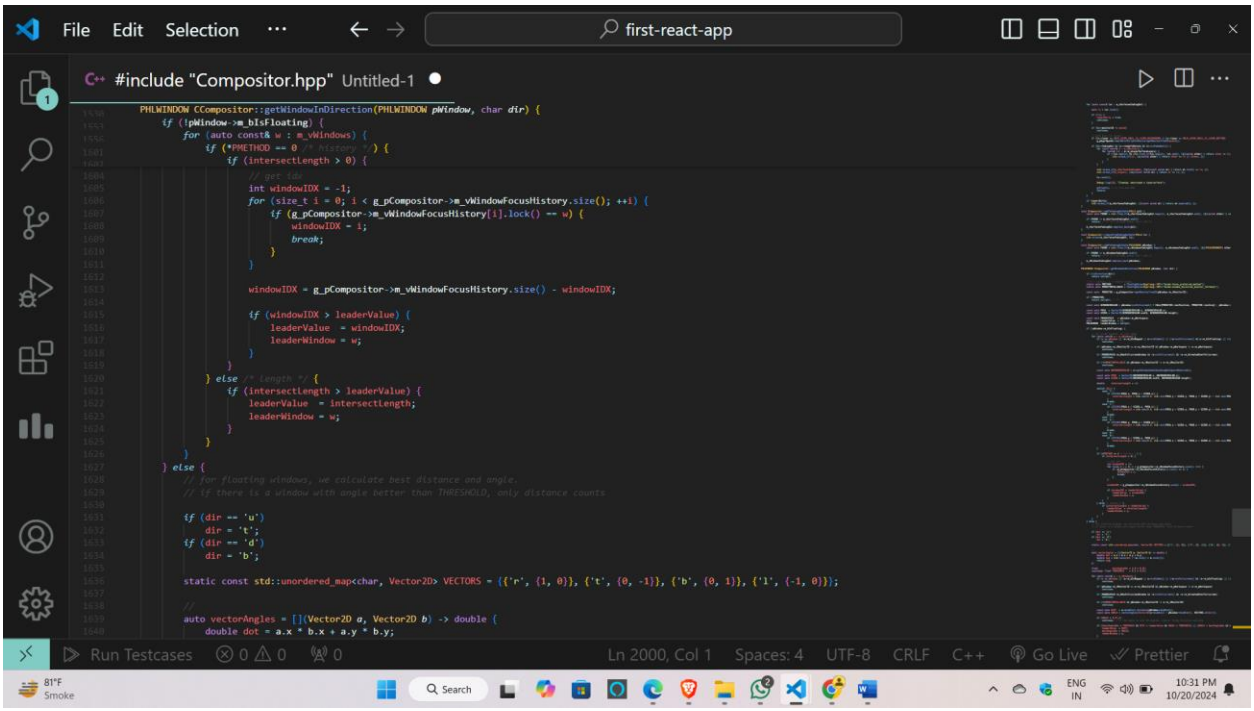

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::changeWindowZOrder(PHLWINDOW pWindow, bool top) {
auto moveToZ = [&](PHLWINDOW pw, bool top) -> void {
if (pw->m_bIsMapped)
g_pHyprRenderer->damageMonitor(getMonitorFromID(pw->m_iMonitorID));
};
if (top)
pWindow->m_bCreatedOverFullscreen = true;
if (!pWindow->m_bIsX11)
moveToZ(pWindow, top);
else {
// move X11 window stack
std::deque<PHLWINDOW> toMove;
auto x11Stack = [&](PHLWINDOW pw, bool top, auto&& x11Stack) -> void {
if (top)
toMove.emplace_back(pw);
else
toMove.emplace_front(pw);
for (auto const& w : m_vWindows) {
if (w->m_bIsMapped && !w->isHidden() && w->m_bIsX11 && w->X11TransientFor() == pw && w != pw && std::find(toMove.begin(), toMove.end(), w) == toMove.end())
{
x11Stack(w, top, x11Stack);
}
}
};
x11Stack(pWindow, top, x11Stack);
for (auto it : toMove) {
moveToZ(it, top);
}
}
}
void CCompositor::cleanupFadingOut(const MONITORID& monid) {
for (auto const& ww : m_vWindowsFadingOut) {
auto w = ww.lock();
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:24 PM 10/20/2024

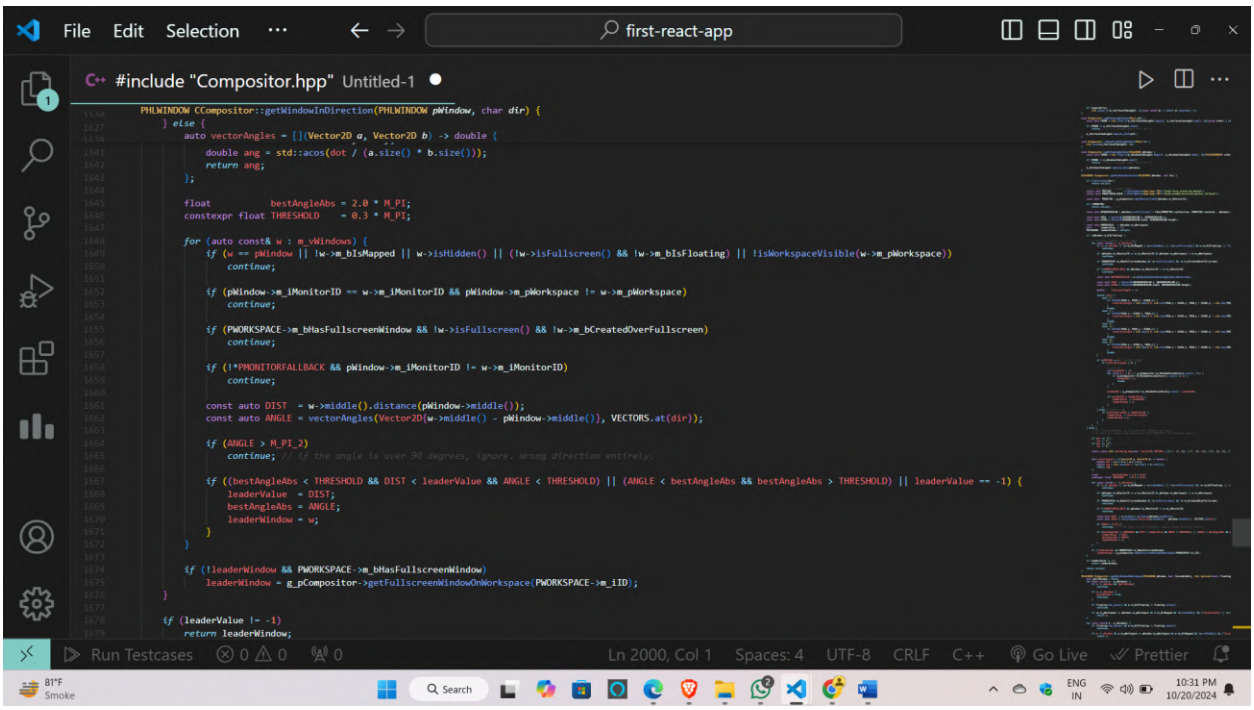

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::cleanupFadingOut(const MONITORID& monid) {
for (auto const& ww : m_vWindowsFadingOut) {
if (w->m_iMonitorID != monid)
continue;
if (!w->m_bFadingOut || w->m_fAlpha.value() == 0.f) {
w->m_bFadingOut = false;
if (!w->m_bReadyToDelete)
continue;
removeWindowFromVectorSafe(w);
w.reset();
Debug::log(LOG, "Cleanup: destroyed a window");
return;
}
}
bool layersDirty = false;
for (auto const& lsr : m_vSurfacesFadingOut) {
auto ls = lsr.lock();
if (!ls) {
layersDirty = true;
continue;
}
if (ls->monitorID != monid)
continue;
// mark blur for recalc
if (ls->layer == ZWLR_LAYER_SHELL_V1_LAYER_BACKGROUND || ls->layer == ZWLR_LAYER_SHELL_V1_LAYER_BOTTOM)
g_pHyprOpenGL->markBlurDirtyForMonitor(getMonitorFromID(monid));
if (ls->fadingOut && ls->readyToDelete && ls->isFadedOut()) {
for (auto const& m : m_vMonitors) {
for (auto& lsl : m->m_aLayerSurfaceLayers) {
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:24 PM
10/20/2024

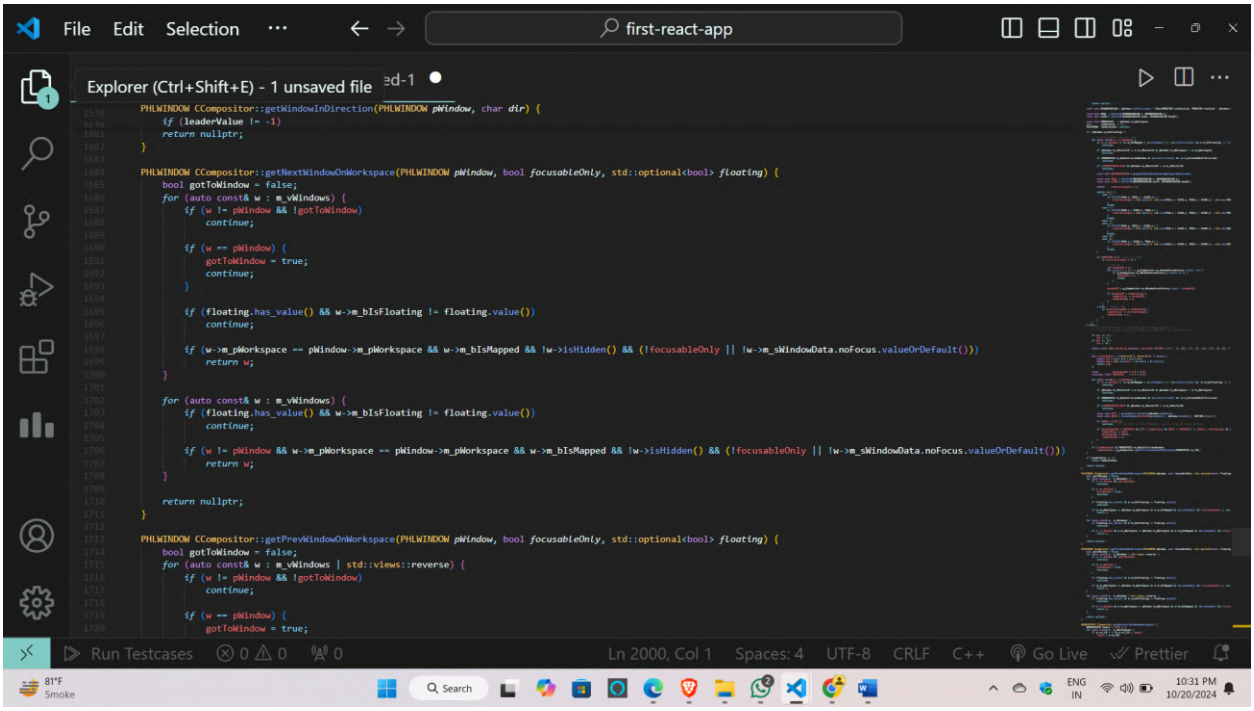

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::cleanupFadingOut(const MONITORID& monid) {
for (auto const& lsr : m_vSurfacesFadingOut) {
if (ls->fadingOut && ls->readyToDelete && ls->isFadedOut()) {
for (auto const& m : m_vMonitors) {
for (auto& lsl : m->m_aLayerSurfaceLayers) {
if (!lsl.empty() && std::find_if(lsl.begin(), lsl.end(), [&](auto& other) { return other == ls; }) != lsl.end()) {
std::erase_if(lsl, [&](auto& other) { return other == ls || !other; });
}
}
}
std::erase_if(m_vSurfacesFadingOut, [ls](const auto& el) { return el.lock() == ls; });
std::erase_if(m_vLayers, [ls](const auto& el) { return el == ls; });
ls.reset();
Debug::log(LOG, "Cleanup: destroyed a layersurface");
glFlush(); // to free mem NOW.
return;
}
}
if (layersDirty)
std::erase_if(m_vSurfacesFadingOut, [](const auto& el) { return el.expired(); });
}
void CCompositor::addToFadingOutSafe(PHLLS pLS) {
const auto FOUND = std::find_if(m_vSurfacesFadingOut.begin(), m_vSurfacesFadingOut.end(), [&](auto& other) { return other.lock() == pLS; });
if (FOUND != m_vSurfacesFadingOut.end())
return; // if it's already added, don't add it.
m_vSurfacesFadingOut.emplace_back(pLS);
}
void CCompositor::removeFromFadingOutSafe(PHLLS ls) {
std::erase(m_vSurfacesFadingOut, ls);
}
void CCompositor::addToFadingOutSafe(PHLWINDOW pWindow) {
const auto FOUND = std::find_if(m_vWindowsFadingOut.begin(), m_vWindowsFadingOut.end(), [&](PHLWINDOWREF& other) { return other.lock() == pWindow; });
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
82°F
Smoke
Search
ENG
IN
10:25 PM
10/20/2024

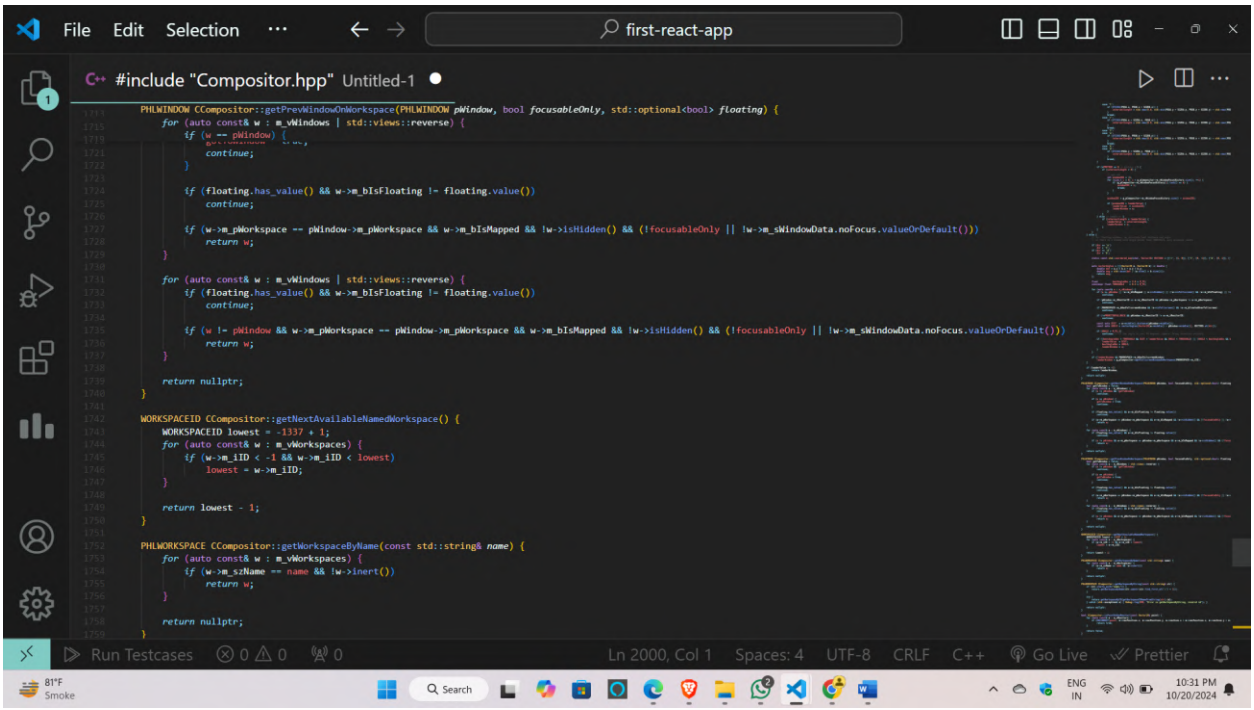

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::addToFadingOutSafe(PHLWINDOW pWindow) {
if (FOUND != m_vWindowsFadingOut.end())
return; // if it's already added, don't add it.
m_vWindowsFadingOut.emplace_back(pWindow);
}
PHLWINDOW CCompositor::getWindowInDirection(PHLWINDOW pWindow, char dir) {
if (!isDirection(dir))
return nullptr;
// 0 -> history, 1 -> shared length
static auto PMETHOD = CConfigValue<Hyprlang::INT>("binds:focus_preferred_method");
static auto PMONITORFALLBACK = CConfigValue<Hyprlang::INT>("binds:window_direction_monitor_fallback");
const auto PMONITOR = g_pCompositor->getMonitorFromID(pWindow->m_iMonitorID);
if (!PMONITOR)
return nullptr; // ??
const auto WINDOWIDEALBB = pWindow->isFullscreen() ? CBox(PMONITOR->vecPosition, PMONITOR->vecSize) : pWindow->getWindowIdealBoundingBoxIgnoreReserved();
const auto POSA = Vector2D(WINDOWIDEALBB.x, WINDOWIDEALBB.y);
const auto SIZEA = Vector2D(WINDOWIDEALBB.width, WINDOWIDEALBB.height);
const auto PWORKSPACE = pWindow->m_pWorkspace;
auto leaderValue = -1;
PHLWINDOW leaderWindow = nullptr;
if (!pWindow->m_bIsFloating) {
// for tiled windows, we calc edges
for (auto const& w : m_vWindows) {
if (w == pWindow || !w->m_bIsMapped || w->isHidden() || (!w->isFullscreen() && w->m_bIsFloating) || !isWorkspaceVisible(w->m_pWorkspace))
continue;
if (pWindow->m_iMonitorID == w->m_iMonitorID && pWindow->m_pWorkspace != w->m_pWorkspace)
continue;
if (PWORKSPACE->m_bHasFullscreenWindow && !w->isFullscreen() && !w->m_bCreatedOverFullscreen)
continue;
Run Testcases ⊗ 0 ⚠ 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:25 PM 10/20/2024

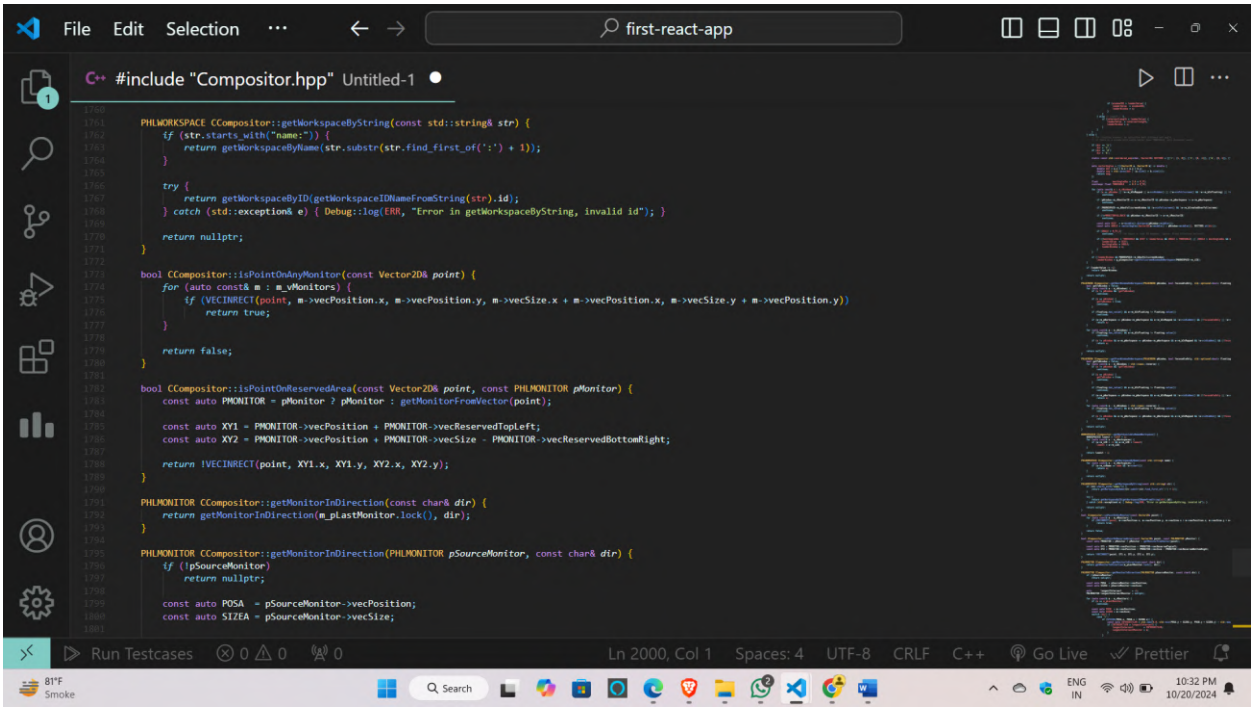

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::getWindowInDirection(PHLWINDOW pWindow, char dir) {
if (!pWindow->m_bIsFloating) {
for (auto const& w : m_vWindows) {
if (PWORKSPACE->m_bHasFullscreenWindow && !w->isFullscreen() && !w->m_bCreatedOverFullscreen)
if (!*PMONITORFALLBACK && pWindow->m_iMonitorID != w->m_iMonitorID)
continue;
const auto BWINDOWIDEALBB = w->getWindowIdealBoundingBoxIgnoreReserved();
const auto POSB = Vector2D(BWINDOWIDEALBB.x, BWINDOWIDEALBB.y);
const auto SIZEB = Vector2D(BWINDOWIDEALBB.width, BWINDOWIDEALBB.height);
double intersectLength = -1;
switch (dir) {
case 'l':
if (STICKS(POSA.x, POSB.x + SIZEB.x)) {
intersectLength = std::max(0.0, std::min(POSA.y + SIZEA.y, POSB.y + SIZEB.y) - std::max(POSA.y, POSB.y));
}
break;
case 'r':
if (STICKS(POSA.x + SIZEA.x, POSB.x)) {
intersectLength = std::max(0.0, std::min(POSA.y + SIZEA.y, POSB.y + SIZEB.y) - std::max(POSA.y, POSB.y));
}
break;
case 't':
case 'u':
if (STICKS(POSA.y, POSB.y + SIZEB.y)) {
intersectLength = std::max(0.0, std::min(POSA.x + SIZEA.x, POSB.x + SIZEB.x) - std::max(POSA.x, POSB.x));
}
break;
case 'b':
case 'd':
if (STICKS(POSA.y + SIZEA.y, POSB.y)) {
intersectLength = std::max(0.0, std::min(POSA.x + SIZEA.x, POSB.x + SIZEB.x) - std::max(POSA.x, POSB.x));
}
break;
}
if (*PMETHOD == 0 /* history */) {
if (intersectLength > 0) {
Run Testcases ⊗ 0 ⚠ 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
82°F Smoke
Search
ENG IN
10:25 PM 10/20/2024

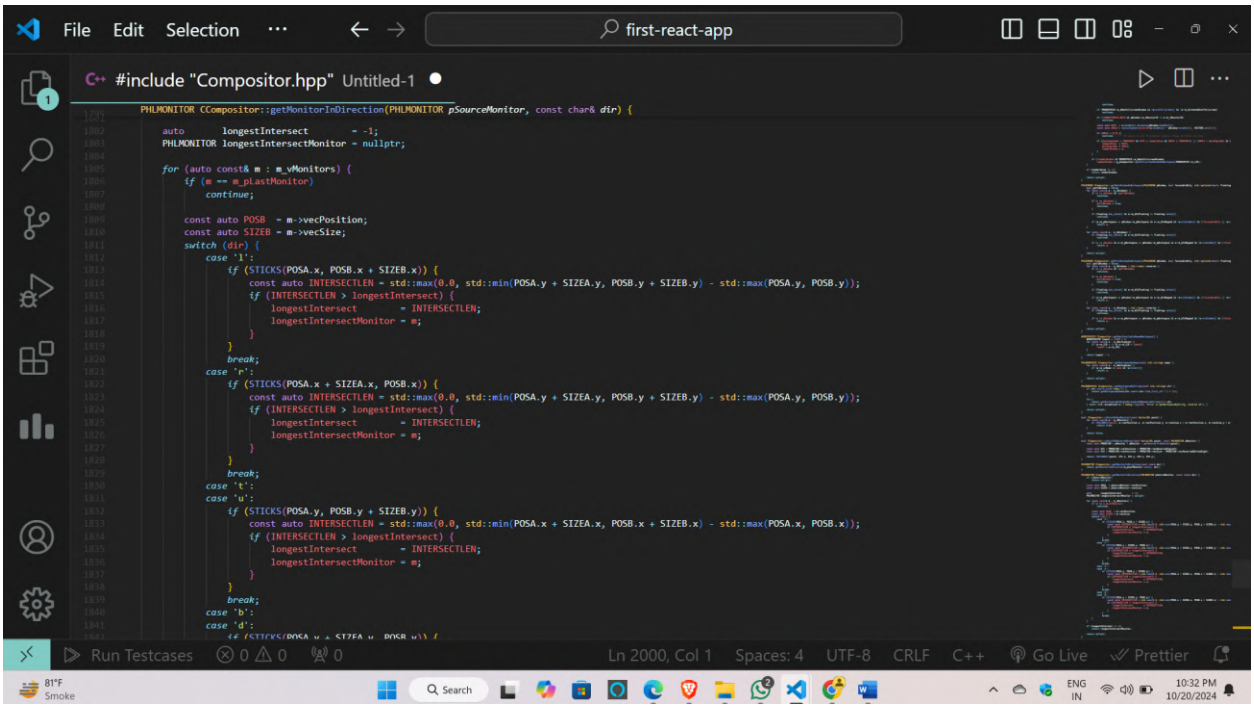

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::getWindowInDirection(PHLWINDOW pWindow, char dir) {
if (!pWindow->m_bIsFloating) {
for (auto const& w : m_vWindows) {
if (*PMETHOD == 0 /* history */) {
if (intersectLength > 0) {
// get id
int windowIDX = -1;
for (size_t i = 0; i < g_pCompositor->m_vWindowFocusHistory.size(); ++i) {
if (g_pCompositor->m_vWindowFocusHistory[i].lock() == w) {
windowIDX = i;
break;
}
}
windowIDX = g_pCompositor->m_vWindowFocusHistory.size() - windowIDX;
if (windowIDX > leaderValue) {
leaderValue = windowIDX;
leaderWindow = w;
}
}
} else /* length */ {
if (intersectLength > leaderValue) {
leaderValue = intersectLength;
leaderWindow = w;
}
}
}
} else {
// for floating windows, we calculate best distance and angle.
// if there is a window with angle better than THRESHOLD, only distance counts
if (dir == 'u')
dir = 't';
if (dir == 'd')
dir = 'b';
static const std::unordered_map<char, Vector2D> VECTORS = {{'r', {1, 0}}, {'t', {0, -1}}, {'b', {0, 1}}, {'l', {-1, 0}}};
//
auto vectorAngles = [](Vector2D a, Vector2D b) -> double {
double dot = a.x * b.x + a.y * b.y;
Run Testcases 0 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
81°F Smoke
Search
ENG IN
10:31 PM 10/20/2024

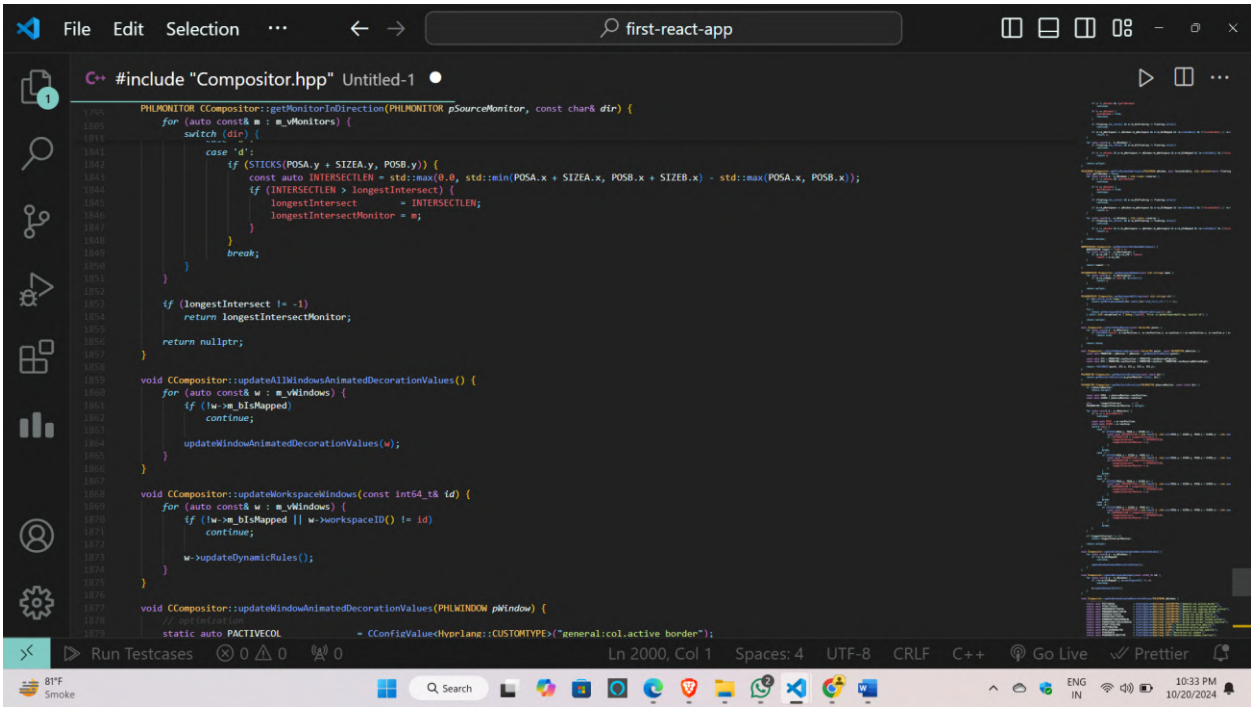

File Edit Selection ···
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::getWindowInDirection(PHLWINDOW pWindow, char dir) {
} else {
auto vectorAngles = [](Vector2D a, Vector2D b) -> double {
double ang = std::acos(dot / (a.size() * b.size()));
return ang;
};
float bestAngleAbs = 2.0 * M_PI;
constexpr float THRESHOLD = 0.3 * M_PI;
for (auto const& w : m_vWindows) {
if (w == pWindow || !w->m_bIsMapped || w->isHidden() || (!w->isFullscreen() && !w->m_bIsFloating) || !isWorkspaceVisible(w->m_pWorkspace))
continue;
if (pWindow->m_iMonitorID == w->m_iMonitorID && pWindow->m_pWorkspace != w->m_pWorkspace)
continue;
if (PWORKSPACE->m_bHasFullscreenWindow && !w->isFullscreen() && !w->m_bCreatedOverFullscreen)
continue;
if (!*PMONITORFALLBACK && pWindow->m_iMonitorID != w->m_iMonitorID)
continue;
const auto DIST = w->middle().distance(pWindow->middle());
const auto ANGLE = vectorAngles(Vector2D(w->middle() - pWindow->middle()), VECTORS.at(dir));
if (ANGLE > M_PI_2)
continue; // if the angle is over 90 degrees, ignore. Wrong direction entirely.
if ((bestAngleAbs < THRESHOLD && DIST < leaderValue && ANGLE < THRESHOLD) || (ANGLE < bestAngleAbs && bestAngleAbs > THRESHOLD) || leaderValue == -1) {
leaderValue = DIST;
bestAngleAbs = ANGLE;
leaderWindow = w;
}
}
if (!leaderWindow && PWORKSPACE->m_bHasFullscreenWindow)
leaderWindow = g_pCompositor->getFullscreenWindowOnWorkspace(PWORKSPACE->m_iID);
}
if (leaderValue != -1)
return leaderWindow;
Run Testcases 0 0 0
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
81°F Smoke
Search
ENG IN
10:31 PM 10/20/2024

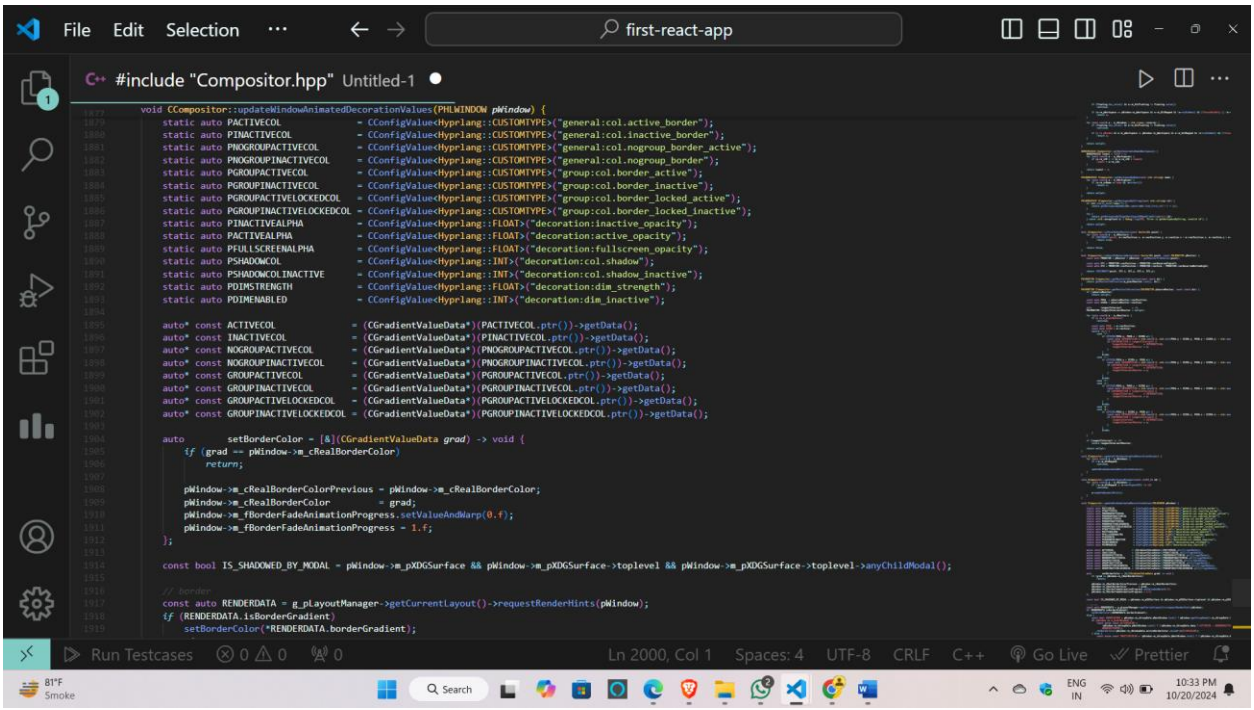

File Edit Selection
first-react-app
Explorer (Ctrl+Shift+E) - 1 unsaved file
PHLWINDOW CCompositor::getWindowInDirection(PHLWINDOW pWindow, char dir) {
if (leaderValue != -1)
return nullptr;
}
PHLWINDOW CCompositor::getNextWindowOnWorkspace(PHLWINDOW pWindow, bool focusableOnly, std::optional<bool> floating) {
bool gotToWindow = false;
for (auto const& w : m_vWindows) {
if (w != pWindow && !gotToWindow)
continue;
if (w == pWindow) {
gotToWindow = true;
continue;
}
if (floating.has_value() && w->m_bIsFloating != floating.value())
continue;
if (w->m_pWorkspace == pWindow->m_pWorkspace && w->m_bIsMapped && !w->isHidden() && (!focusableOnly || !w->m_sWindowData.noFocus.valueOrDefault()))
return w;
}
for (auto const& w : m_vWindows) {
if (floating.has_value() && w->m_bIsFloating != floating.value())
continue;
if (w != pWindow && w->m_pWorkspace == pWindow->m_pWorkspace && w->m_bIsMapped && !w->isHidden() && (!focusableOnly || !w->m_sWindowData.noFocus.valueOrDefault()))
return w;
}
return nullptr;
}
PHLWINDOW CCompositor::getPrevWindowOnWorkspace(PHLWINDOW pWindow, bool focusableOnly, std::optional<bool> floating) {
bool gotToWindow = false;
for (auto const& w : m_vWindows | std::views::reverse) {
if (w != pWindow && !gotToWindow)
continue;
if (w == pWindow) {
gotToWindow = true;
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
81°F
Smoke
Search
ENG
IN
10:31 PM
10/20/2024

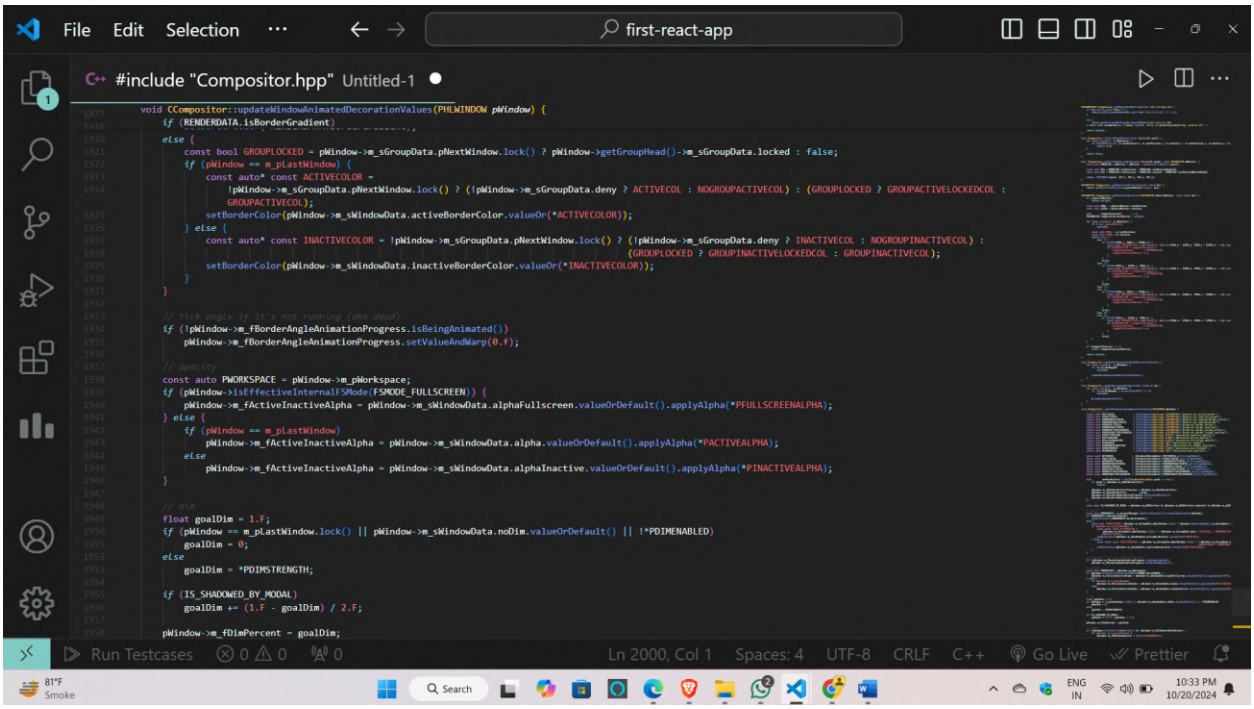

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWINDOW CCompositor::getPrevWindowOnWorkspace(PHLWINDOW pWindow, bool focusableOnly, std::optional<bool> floating) {
for (auto const& w : m_vWindows | std::views::reverse) {
if (w == pWindow) {
continue;
}
if (floating.has_value() && w->m_bIsFloating != floating.value())
continue;
if (w->m_pWorkspace == pWindow->m_pWorkspace && w->m_bIsMapped && !w->isHidden() && (!focusableOnly || !w->m_sWindowData.noFocus.valueOrDefault()))
return w;
}
for (auto const& w : m_vWindows | std::views::reverse) {
if (floating.has_value() && w->m_bIsFloating != floating.value())
continue;
if (w != pWindow && w->m_pWorkspace == pWindow->m_pWorkspace && w->m_bIsMapped && !w->isHidden() && (!focusableOnly || !w->m_sWindowData.noFocus.valueOrDefault()))
return w;
}
return nullptr;
}
WORKSPACEID CCompositor::getNextAvailableNamedWorkspace() {
WORKSPACEID lowest = -1337 + 1;
for (auto const& w : m_vWorkspaces) {
if (w->m_iID < -1 && w->m_iID < lowest)
lowest = w->m_iID;
}
return lowest - 1;
}
PHLWORKSPACE CCompositor::getWorkspaceByName(const std::string& name) {
for (auto const& w : m_vWorkspaces) {
if (w->m_szName == name && !w->inert())
return w;
}
return nullptr;
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
81°F
Smoke
Search
ENG
IN
10:31 PM
10/20/2024

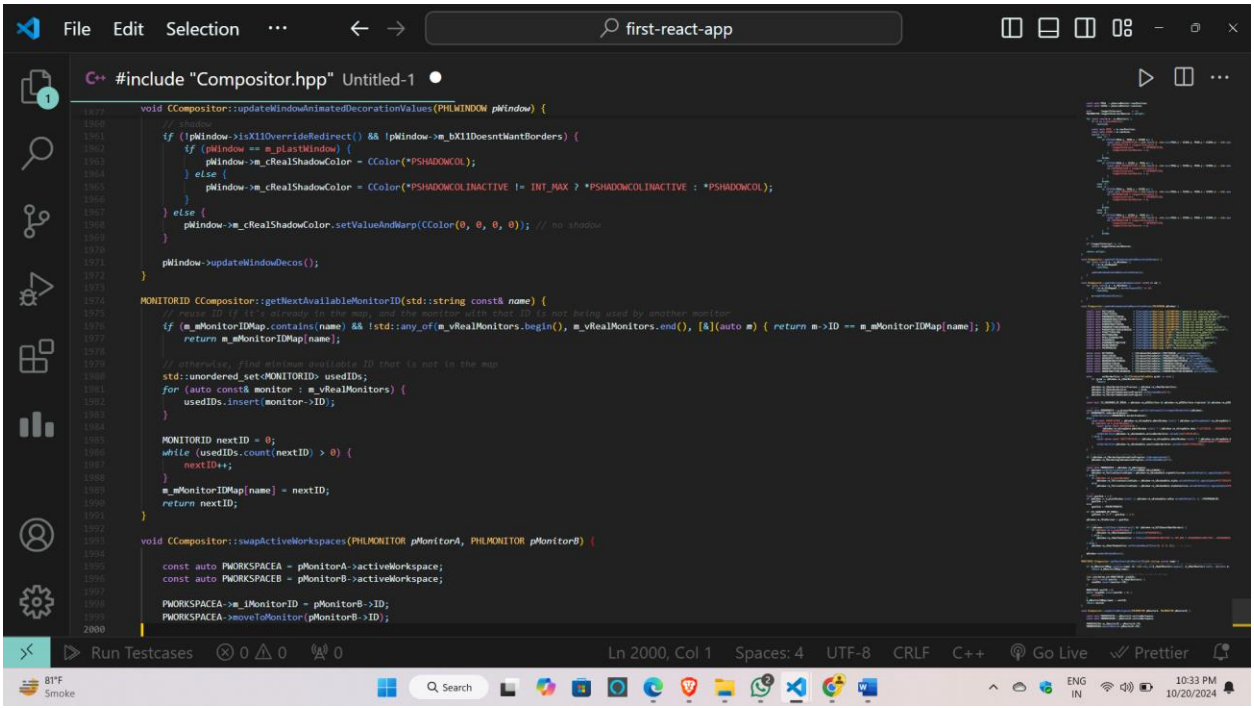

File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLWORKSPACE CCompositor::getWorkspaceByString(const std::string& str) {
if (str.starts_with("name:")) {
return getWorkspaceByName(str.substr(str.find_first_of(':') + 1));
}
try {
return getWorkspaceByID(getWorkspaceIDNameFromString(str).id);
} catch (std::exception& e) { Debug::log(ERR, "Error in getWorkspaceByString, invalid id"); }
return nullptr;
}
bool CCompositor::isPointOnAnyMonitor(const Vector2D& point) {
for (auto const& m : m_vMonitors) {
if (VECINRECT(point, m->vecPosition.x, m->vecPosition.y, m->vecSize.x + m->vecPosition.x, m->vecSize.y + m->vecPosition.y))
return true;
}
return false;
}
bool CCompositor::isPointOnReservedArea(const Vector2D& point, const PHLMONITOR pMonitor) {
const auto PMONITOR = pMonitor ? pMonitor : getMonitorFromVector(point);
const auto XY1 = PMONITOR->vecPosition + PMONITOR->vecReservedTopLeft;
const auto XY2 = PMONITOR->vecPosition + PMONITOR->vecSize - PMONITOR->vecReservedBottomRight;
return !VECINRECT(point, XY1.x, XY1.y, XY2.x, XY2.y);
}
PHLMONITOR CCompositor::getMonitorInDirection(const char& dir) {
return getMonitorInDirection(m_pLastMonitor.lock(), dir);
}
PHLMONITOR CCompositor::getMonitorInDirection(PHLMONITOR pSourceMonitor, const char& dir) {
if (!pSourceMonitor)
return nullptr;
const auto POSA = pSourceMonitor->vecPosition;
const auto SIZEA = pSourceMonitor->vecSize;
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
81°F Smoke
Search
ENG IN
10:32 PM 10/20/2024


File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLMONITOR CCompositor::getMonitorInDirection(PHLMONITOR pSourceMonitor, const char& dir) {
auto longestIntersect = -1;
PHLMONITOR longestIntersectMonitor = nullptr;
for (auto const& m : m_vMonitors) {
if (m == m_pLastMonitor)
continue;
const auto POSB = m->vecPosition;
const auto SIZEB = m->vecSize;
switch (dir) {
case 'l':
if (STICKS(POSA.x, POSB.x + SIZEB.x)) {
const auto INTERSECTLEN = std::max(0.0, std::min(POSA.y + SIZEA.y, POSB.y + SIZEB.y) - std::max(POSA.y, POSB.y));
if (INTERSECTLEN > longestIntersect) {
longestIntersect = INTERSECTLEN;
longestIntersectMonitor = m;
}
}
break;
case 'r':
if (STICKS(POSA.x + SIZEA.x, POSB.x)) {
const auto INTERSECTLEN = std::max(0.0, std::min(POSA.y + SIZEA.y, POSB.y + SIZEB.y) - std::max(POSA.y, POSB.y));
if (INTERSECTLEN > longestIntersect) {
longestIntersect = INTERSECTLEN;
longestIntersectMonitor = m;
}
}
break;
case 't':
case 'u':
if (STICKS(POSA.y, POSB.y + SIZEB.y)) {
const auto INTERSECTLEN = std::max(0.0, std::min(POSA.x + SIZEA.x, POSB.x + SIZEB.x) - std::max(POSA.x, POSB.x));
if (INTERSECTLEN > longestIntersect) {
longestIntersect = INTERSECTLEN;
longestIntersectMonitor = m;
}
}
break;
case 'b':
case 'd':
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
81°F Smoke
Search
ENG IN
10:32 PM 10/20/2024


File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
PHLMONITOR CCompositor::getMonitorInDirection(PHLMONITOR pSourceMonitor, const char& dir) {
for (auto const& m : m_vMonitors) {
switch (dir) {
case 'd':
if (STICKS(POSA.y + SIZEA.y, POSB.y)) {
const auto INTERSECTLEN = std::max(0.0, std::min(POSA.x + SIZEA.x, POSB.x + SIZEB.x) - std::max(POSA.x, POSB.x));
if (INTERSECTLEN > longestIntersect) {
longestIntersect = INTERSECTLEN;
longestIntersectMonitor = m;
}
}
break;
}
}
if (longestIntersect != -1)
return longestIntersectMonitor;
return nullptr;
}
void CCompositor::updateAllWindowsAnimatedDecorationValues() {
for (auto const& w : m_vWindows) {
if (!w->m_bIsMapped)
continue;
updateWindowAnimatedDecorationValues(w);
}
}
void CCompositor::updateWorkspaceWindows(const int64_t& id) {
for (auto const& w : m_vWindows) {
if (!w->m_bIsMapped || w->workspaceID() != id)
continue;
w->updateDynamicRules();
}
}
void CCompositor::updateWindowAnimatedDecorationValues(PHLWINDOW pWindow) {
// optimization
static auto PACTIVECOL = CConfigValue<Hyprlang::CUSTOMTYPE>("general:col.active_border");
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
81°F
Smoke
Search
ENG
IN
10:33 PM
10/20/2024


File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::updateWindowAnimatedDecorationValues(PHLWINDOW pWindow) {
static auto PACTIVECOL = CConfigValue<Hyprlang::CUSTOMTYPE>("general:col.active_border");
static auto PINACTIVECOL = CConfigValue<Hyprlang::CUSTOMTYPE>("general:col.inactive_border");
static auto PNOGROUPACTIVECOL = CConfigValue<Hyprlang::CUSTOMTYPE>("general:col.nogroup_border_active");
static auto PNOGROUPINACTIVECOL = CConfigValue<Hyprlang::CUSTOMTYPE>("general:col.nogroup_border");
static auto PGROUPACTIVECOL = CConfigValue<Hyprlang::CUSTOMTYPE>("group:col.border_active");
static auto PGROUPINACTIVECOL = CConfigValue<Hyprlang::CUSTOMTYPE>("group:col.border_inactive");
static auto PGROUPACTIVELOCKEDCOL = CConfigValue<Hyprlang::CUSTOMTYPE>("group:col.border_locked_active");
static auto PGROUPINACTIVELOCKEDCOL = CConfigValue<Hyprlang::CUSTOMTYPE>("group:col.border_locked_inactive");
static auto PINACTIVEALPHA = CConfigValue<Hyprlang::FLOAT>("decoration:inactive_opacity");
static auto PACTIVEALPHA = CConfigValue<Hyprlang::FLOAT>("decoration:active_opacity");
static auto PFULLSCREENALPHA = CConfigValue<Hyprlang::FLOAT>("decoration:fullscreen_opacity");
static auto PSHADOWCOL = CConfigValue<Hyprlang::INT>("decoration:col.shadow");
static auto PSHADOWCOLINACTIVE = CConfigValue<Hyprlang::INT>("decoration:col.shadow_inactive");
static auto PDIMSTRENGTH = CConfigValue<Hyprlang::FLOAT>("decoration:dim_strength");
static auto PDIMENABLED = CConfigValue<Hyprlang::INT>("decoration:dim_inactive");
auto* const ACTIVECOL = (CGradientValueData*)(PACTIVECOL.ptr())->getData();
auto* const INACTIVECOL = (CGradientValueData*)(PINACTIVECOL.ptr())->getData();
auto* const NOGROUPACTIVECOL = (CGradientValueData*)(PNOGROUPACTIVECOL.ptr())->getData();
auto* const NOGROUPINACTIVECOL = (CGradientValueData*)(PNOGROUPINACTIVECOL.ptr())->getData();
auto* const GROUPACTIVECOL = (CGradientValueData*)(PGROUPACTIVECOL.ptr())->getData();
auto* const GROUPINACTIVECOL = (CGradientValueData*)(PGROUPINACTIVECOL.ptr())->getData();
auto* const GROUPACTIVELOCKEDCOL = (CGradientValueData*)(PGROUPACTIVELOCKEDCOL.ptr())->getData();
auto* const GROUPINACTIVELOCKEDCOL = (CGradientValueData*)(PGROUPINACTIVELOCKEDCOL.ptr())->getData();
auto setBorderColor = [&](CGradientValueData grad) -> void {
if (grad == pWindow->m_cRealBorderColor)
return;
pWindow->m_cRealBorderColorPrevious = pWindow->m_cRealBorderColor;
pWindow->m_cRealBorderColor = grad;
pWindow->m_fBorderFadeAnimationProgress.setValueAndWarp(0.f);
pWindow->m_fBorderFadeAnimationProgress = 1.f;
};
const bool IS_SHADOWED_BY_MODAL = pWindow->m_pXDGSurface && pWindow->m_pXDGSurface->toplevel && pWindow->m_pXDGSurface->toplevel->anyChildModal();
// Border
const auto RENDERDATA = g_pLayoutManager->getCurrentLayout()->requestRenderHints(pWindow);
if (RENDERDATA.isBorderGradient)
setBorderColor(*RENDERDATA.borderGradient);
Run Testcases
Ln 2000, Col 1
Spaces: 4
UTF-8
CRLF
C++
Go Live
Prettier
81°F
Smoke
Search
ENG
IN
10:33 PM
10/20/2024


File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::updateWindowAnimatedDecorationValues(PHLWINDOW pWindow) {
if (RENDERDATA.isBorderGradient)
else {
const bool GROUPLOCKED = pWindow->m_sGroupData.pNextWindow.lock() ? pWindow->getGroupHead()->m_sGroupData.locked : false;
if (pWindow == m_pLastWindow) {
const auto* const ACTIVECOLOR =
!pWindow->m_sGroupData.pNextWindow.lock() ? (!pWindow->m_sGroupData.deny ? ACTIVECOL : NOGROUPACTIVECOL) : (GROUPLOCKED ? GROUPACTIVELOCKEDCOL :
GROUPACTIVECOL);
setBorderColor(pWindow->m_sWindowData.activeBorderColor.valueOr(*ACTIVECOLOR));
} else {
const auto* const INACTIVECOLOR = !pWindow->m_sGroupData.pNextWindow.lock() ? (!pWindow->m_sGroupData.deny ? INACTIVECOL : NOGROUPINACTIVECOL) :
(GROUPLOCKED ? GROUPINACTIVELOCKEDCOL : GROUPINACTIVECOL);
setBorderColor(pWindow->m_sWindowData.inactiveBorderColor.valueOr(*INACTIVECOLOR));
}
}
// rick angle if it's not running (aka dead)
if (!pWindow->m_fBorderAngleAnimationProgress.isBeingAnimated())
pWindow->m_fBorderAngleAnimationProgress.setValueAndWarp(0.f);
// opacity
const auto PWORKSPACE = pWindow->m_pWorkspace;
if (pWindow->isEffectiveInternalFSMode(FSMODE_FULLSCREEN)) {
pWindow->m_fActiveInactiveAlpha = pWindow->m_sWindowData.alphaFullscreen.valueOrDefault().applyAlpha(*PFULLSCREENALPHA);
} else {
if (pWindow == m_pLastWindow)
pWindow->m_fActiveInactiveAlpha = pWindow->m_sWindowData.alpha.valueOrDefault().applyAlpha(*PACTIVEALPHA);
else
pWindow->m_fActiveInactiveAlpha = pWindow->m_sWindowData.alphaInactive.valueOrDefault().applyAlpha(*PINACTIVEALPHA);
}
// dim
float goalDim = 1.F;
if (pWindow == m_pLastWindow.lock() || pWindow->m_sWindowData.noDim.valueOrDefault() || !*PDIMENABLED)
goalDim = 0;
else
goalDim = *PDIMSTRENGTH;
if (IS_SHADOWED_BY_MODAL)
goalDim += (1.F - goalDim) / 2.F;
pWindow->m_fDimPercent = goalDim;
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
81°F Smoke
Search
ENG IN
10:33 PM 10/20/2024


File Edit Selection
first-react-app
#include "Compositor.hpp" Untitled-1
void CCompositor::updateWindowAnimatedDecorationValues(PHLWINDOW pWindow) {
if (!pWindow->isX11OverrideRedirect() && !pWindow->m_bX11DoesntWantBorders) {
if (pWindow == m_pLastWindow) {
pWindow->m_cRealShadowColor = CColor(*PSHADOWCOL);
} else {
pWindow->m_cRealShadowColor = CColor(*PSHADOWCOLINACTIVE != INT_MAX ? *PSHADOWCOLINACTIVE : *PSHADOWCOL);
}
} else {
pWindow->m_cRealShadowColor.setValueAndWarp(CColor(0, 0, 0, 0)); // no shadow
}
pWindow->updateWindowDecos();
}
MONITORID CCompositor::getNextAvailableMonitorID(std::string const& name) {
// reuse ID if it's already in the map, and the monitor with that ID is not being used by another monitor
if (m_mMonitorIDMap.contains(name) && !std::any_of(m_vRealMonitors.begin(), m_vRealMonitors.end(), [&](auto m) { return m->ID == m_mMonitorIDMap[name]; }))
return m_mMonitorIDMap[name];
// otherwise, find the next available ID that is not in the map
std::unordered_set<MONITORID> usedIDs;
for (auto const& monitor : m_vRealMonitors) {
usedIDs.insert(monitor->ID);
}
MONITORID nextID = 0;
while (usedIDs.count(nextID) > 0) {
nextID++;
}
m_mMonitorIDMap[name] = nextID;
return nextID;
}
void CCompositor::swapActiveWorkspaces(PHLMONITOR pMonitorA, PHLMONITOR pMonitorB) {
const auto PWORKSPACEA = pMonitorA->activeWorkspace;
const auto PWORKSPACEB = pMonitorB->activeWorkspace;
PWORKSPACEA->m_iMonitorID = pMonitorB->ID;
PWORKSPACEA->moveToMonitor(pMonitorB->ID);
Run Testcases
Ln 2000, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Prettier
81°F Smoke
Search
ENG IN
10:33 PM 10/20/2024

CODE : 1 (1-550 lines)

1. How many errors can you identify in the program?

Category A: Data Reference Errors

- Uninitialized or unset variables: In the constructor CCompositor(), the members like m_szHyprTempDataRoot and m_szInstanceSignature are initialized properly, but there is no check to handle invalid environment variables (like if XDG_RUNTIME_DIR is unset).

Category B: Syntax Errors

- None found**.**

Category C: Computation Errors

- No computation errors are apparent, but certain functions like bumpNofile() handle resource limits (rlimit) which may fail under specific system configurations without proper checks.

Category D: Comparison Errors

- There doesn't appear to be faulty comparison logic directly related to this section of the code.

Category E: Control-Flow Errors

- Inconsistent error handling: If any directory operation (mkdir) fails, there is no specific clean-up routine to undo previously successful operations. This could leave residual files or folders.
- Off-by-one mistake: No clear off-by-one errors detected in the range-based loops used here.

Category F: Interface Errors

- Incorrect handling of parameters: Some environment variables like WAYLAND_DISPLAY and XDG_RUNTIME_DIR are assumed to be valid, but there's no strict validation before using them, which can lead to failures.

Category G: Input/Output Errors

- Missing error handling for input data: When creating or setting up directories (like m_szHyprTempDataRoot), error-handling mechanisms are basic and don't provide detailed feedback on failures.

Category H: Other Checks

- Missing libraries: No missing libraries, but the program assumes certain system environment variables are set. If these are not set, it may fail.

---

2. Which category of inspection would you consider more useful?

- **Category A: Data Reference Errors:** Ensuring the environment variables like XDG_RUNTIME_DIR are correctly set and handling cases where these variables are invalid would prevent crashes and runtime issues.
- **Category E: Control-Flow Errors: I**mproving error handling (especially directory creation and clean-up) would prevent inconsistent system states, like partially created directories if an error occurs midway.

---

3. Which type of error are you not able to identify using the program inspection?

- **Logic Errors:** The program inspection might not identify issues like whether the compositor correctly manages all connected devices (e.g., handling new outputs and inputs). These errors would only show up during actual execution, especially under specific runtime conditions (e.g., missing devices or unusual environment setups).

---

4. Is the program inspection technique worth applying?

Yes, the inspection technique helps in identifying common errors like missing initialization, incorrect assumptions about environment variables, and control flow issues in directory creation and resource management. However, it should be complemented with real-world testing, especially to detect issues like incorrect input handling, environmental dependencies, and runtime performance under various conditions.

CODE : 2 (551-1004 lines)

1. How many errors can you identify in the program?

Category A: Data Reference Errors

- Unset variables or values: The program handles a few environmental variables like WAYLAND_DISPLAY and XDG_RUNTIME_DIR but does not explicitly check if they are set before using them.

Category B: Syntax Errors

- None found: The code appears syntactically correct.

Category C: Computation Errors

- No obvious computation errors: The code doesn't perform complex mathematical operations that could be prone to errors. However, the behavior of certain functions (like creating and removing lock files or handling monitors) might have subtle side effects that require runtime testing.

Category D: Comparison Errors

- None found: The comparison logic in the loops and conditionals seems correct.

Category E: Control-Flow Errors

- Inconsistent error handling: The function createLockFile() writes a file but does not handle potential I/O errors (like disk write failure). Similarly, prepareFallbackOutput() does not fully manage cases where the output backend may fail to initialize.
- Potential infinite loop: The loop in initManagers() (while (!g_pWatchdog->m_bWatchdogInitialized) { std::this_thread::yield(); }) could potentially result in an infinite loop if m_bWatchdogInitialized never gets set.

Category F: Interface Errors

- Incorrect parameter handling: In startCompositor(), system commands are constructed using environment variables, but there’s no thorough validation or fallback in case these variables are missing or incorrectly set.

Category G: Input/Output Errors

- Missing error handling for I/O operations: Functions such as createLockFile() and removeLockFile() assume that file creation/removal will succeed, but there is no error handling for cases when these operations fail.
- Inadequate validation for environment variables: Before invoking external commands (e.g., systemctl and dbus-update-activation-environment), the environment variables are assumed to be correct without checks.

Category H: Other Checks

- Missing system checks: The program relies heavily on specific system configurations (e.g., presence of Wayland/XWayland, systemd integration) without explicitly verifying whether these dependencies are present before proceeding.

---

2. **Most Useful Inspection Categories:**
    - **Category A: Data Reference Errors:** Verifying the proper initialization and validation of environment variables (like WAYLAND_DISPLAY and XDG_RUNTIME_DIR) is critical to avoid runtime crashes.
    - **Category E: Control-Flow Errors:** Improving error handling, particularly for resource initialization and directory management (such as the prepareFallbackOutput() function), will help prevent inconsistencies like lingering lock files or incomplete setups.

---

3. **Type of Errors Not Identified Through Inspection:**
    - **Logic Errors:** The program inspection might miss errors like the compositor failing to handle dynamic changes in connected devices (e.g., monitors or inputs being added/removed). These issues are typically revealed only through real-world execution under various conditions.

---

4. **Value of Program Inspection:**
    - Yes, the inspection technique is worth applying. It helps in identifying key issues like missing validation of environment variables, unhandled errors in I/O operations, and control flow inefficiencies. However, these findings should be validated with real-world testing to ensure the program handles edge cases like missing system dependencies or runtime failures.

CODE : 3(1003-1551 lines)

1. **How many errors can you identify in the program? List the errors below.**

Category A: Data Reference Errors:

- There are no explicit uninitialized variables in this program, but handling for invalid/missing parameters is limited.

Category C: Computation Errors:

- There are no explicit computation errors like division by zero.

Category D: Comparison Errors:

- In CCompositor::focusWindow, the logic for checking m_pLastWindow.lock() does not account for cases where windows could have identical data in certain edge cases, potentially leading to faulty focus handling.

Category E: Control-Flow Errors:

- The method CCompositor::changeWindowZOrder could face control flow issues if it does not properly account for all conditions of X11 windows.

Category F: Interface Errors:

- Input parameters are not fully validated in terms of null or invalid values for functions like focusWindow or focusSurface.

Category G: Input/Output Errors:

- There is no handling for cases where m_vWindowFocusHistory is empty or fails to find a history pivot, resulting in a potential crash.

Category H: Other Checks:

- Missing header or library imports are not an issue in this context since the code relies on specific internal APIs and classes.

2. **Most Useful Inspection Category:**

- **Category D: Comparison Errors:** The comparison logic for handling window focus, particularly with m_pLastWindow.lock(), may produce unexpected results under certain conditions. Fixing this is critical to ensuring that window focus management operates correctly**.**

3. **Errors not Easily Identified:**

- **Logic Errors:** While the program inspection can catch many structural and reference issues, it may fail to detect subtle logic errors. For example, ensuring that focusWindow and focusSurface correctly handle edge cases like overlapping surfaces or quickly switching between windows might require dynamic runtime testing.

4. **Is Program Inspection Worth Applying?**

- Yes: Program inspection helps identify key categories of errors like data reference issues and control-flow anomalies. However, it should be combined with runtime testing and debugging to ensure full coverage, especially for complex window management logic.

CODE : 4(1552-2000 lines):

1. How many errors are there in the program? Mention the errors you have identified.

Category A: Data Reference Errors:
There are no explicit uninitialized variables in this program, but handling for invalid/missing parameters is limited.

Category C: Computation Errors:
There are no explicit computation errors like division by zero.

Category D: Comparison Errors:
In CCompositor::focusWindow, the logic for checking m_pLastWindow.lock() does not account for cases where windows could have identical data in certain edge cases, potentially leading to faulty focus handling**.**

Category E: Control-Flow Errors:
The method CCompositor::changeWindowZOrder could face control flow issues if it does not properly account for all conditions of X11 windows.

Category F: Interface Errors:
Input parameters are not fully validated in terms of null or invalid values for functions like focusWindow or focusSurface.

Category G: Input/Output Errors:
There is no handling for cases where m_vWindowFocusHistory is empty or fails to find a history pivot, resulting in a potential crash**.**

Category H: Other Checks:
Missing header or library imports are not an issue in this context since the code relies on specific internal APIs and classes.

Most Useful Inspection Category:

Category D: Comparison Errors:
The comparison logic for handling window focus, particularly with m_pLastWindow.lock(), may produce unexpected results under certain conditions. Fixing this is critical to ensuring that window focus management operates correctly.

Errors not Easily Identified:

Logic Errors:
While the program inspection can catch many structural and reference issues, it may fail to detect subtle logic errors. For example, ensuring that focusWindow and focusSurface correctly handle edge cases like overlapping surfaces or quickly switching between windows might require dynamic runtime testing.

---

2. Which category of program inspection would you find more effective?
Computation Errors (Category C): Ensuring that computations are performed correctly is crucial for the accuracy of results, especially in numerical methods like polynomial fitting.

---

3. Which type of error are you not able to identify using the program inspection?
Data-Specific Errors: Certain edge cases with input data (e.g., all y values being the same) may not be identified until the function is executed with specific datasets.

4.Is Program Inspection Worth Applying?

Yes ,Program inspection helps identify key categories of errors like data reference issues and control-flow anomalies. However, it should be combined with runtime testing and debugging to ensure full coverage, especially for complex window management logic.

SECTION _ II

Armstrong Number: Errors and Fixes

1. How many errors are there in the program?
There are **2 errors** in the program.

2. How many breakpoints do you need to fix those errors?
We need **2 breakpoints** to fix these errors.

- **Steps Taken to Fix the Errors:**

Error 1: The division and modulus operations are swapped in the while loop.
Fix: Ensure that the modulus operation retrieves the last digit, while the division operation reduces the number for the next iteration.

Error 2: The check variable is not properly accumulated.
Fix: Correct the logic to ensure that the check variable accurately reflects the sum of each digit raised to the power of the number of digits.

```
class Armstrong {
   public static void main(String args[]) {
      int num = Integer.parseInt(args[0]);
      int n = num; // use to check at last time
      int check = 0, remainder;

      while (num > 0) {
         remainder = num % 10;
         check = check + (int)Math.pow(remainder, 3);
         num = num / 10;
      }

      if (check == n)
         System.out.println(n + " is an Armstrong Number");
      else
         System.out.println(n + " is not an Armstrong Number");
```

```
  }
}
```

GCD and LCM: Errors and Fixes

1. How many errors are there in the program?
There is **1 error** in the program.

2. How many breakpoints do you need to fix this error?
We need **1 breakpoint** to fix this error.

- **Steps Taken to Fix the Error:**

Error: The condition in the while loop of the GCD method is incorrect.
Fix: Change the condition to while (a % b != 0) instead of while (a % b == 0). This ensures the loop continues until the remainder is zero, correctly calculating the GCD.

```
import java.util.Scanner;

public class GCD_LCM {
   static int gcd(int x, int y) {
      int r = 0, a, b;
      a = (x > y) ? x : y; // a is greater number
      b = (x < y) ? x : y; // b is smaller number

      r = b;
      while (a % b != 0) {
         r = a % b;
         a = b;
         b = r;
      }
      return r;
   }

   static int lcm(int x, int y) {
```

```
    int a;
    a = (x > y) ? x : y; // a is greater number
    while (true) {
      if (a % x == 0 && a % y == 0)
        return a;
      ++a;
    }
  }

  public static void main(String args[]) {
    Scanner input = new Scanner(System.in);
    System.out.println("Enter the two numbers: ");
    int x = input.nextInt();
    int y = input.nextInt();

    System.out.println("The GCD of two numbers is: " + gcd(x, y));
    System.out.println("The LCM of two numbers is: " + lcm(x, y));
    input.close();
  }
}
```

Knapsack Problem: Errors and Fixes

1. **How many errors are there in the program?**
   There are **3 errors** in the program.
2. **How many breakpoints do you need to fix these errors?**
   We need **2 breakpoints** to fix these errors.

- **Steps Taken to Fix the Errors:**

  Error: In the "take item n" case, the condition is incorrect.
  Fix: Change if (weight[n] > w) to if (weight[n] <= w) to ensure the profit is calculated when the item can be included.

Error: The profit calculation is incorrect.
Fix: Change profit[n-2] to profit[n] to ensure the correct profit value is used.

Error: In the "don't take item n" case, the indexing is incorrect.
Fix: Change opt[n++][w] to opt[n-1][w] to properly index the items.

```
public class Knapsack {

   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);   // number of items
      int W = Integer.parseInt(args[1]);   // maximum weight of knapsack

      int[] profit = new int[N+1];
      int[] weight = new int[N+1];

      // generate random instance, items 1..N
      for (int n = 1; n <= N; n++) {
         profit[n] = (int) (Math.random() * 1000);
         weight[n] = (int) (Math.random() * W);
      }

      // opt[n][w] = max profit of packing items 1..n with weight limit w
      // sol[n][w] = does opt solution to pack items 1..n with weight limit w include item
n?
      int[][] opt = new int[N+1][W+1];
      boolean[][] sol = new boolean[N+1][W+1];

      for (int n = 1; n <= N; n++) {
         for (int w = 1; w <= W; w++) {

            // don't take item n
            int option1 = opt[n-1][w];
```

```
            // take item n
            int option2 = Integer.MIN_VALUE;
            if (weight[n] <= w) option2 = profit[n] + opt[n-1][w-weight[n]];

            // select better of two options
            opt[n][w] = Math.max(option1, option2);
            sol[n][w] = (option2 > option1);
         }
      }

      // determine which items to take
      boolean[] take = new boolean[N+1];
      for (int n = N, w = W; n > 0; n--) {
         if (sol[n][w]) { take[n] = true;  w = w - weight[n]; }
         else           { take[n] = false;                    }
      }

      // print results
      System.out.println("item" + "\t" + "profit" + "\t" + "weight" + "\t" + "take");
      for (int n = 1; n <= N; n++) {
         System.out.println(n + "\t" + profit[n] + "\t" + weight[n] + "\t" + take[n]);
      }
   }
}
```

Magic Number Check: Errors and Fixes

1. **How many errors are there in the program?**
   There are **3 errors** in the program.
2. **How many breakpoints do you need to fix these errors?**
   We need **1 breakpoint** to fix these errors.

- **Steps Taken to Fix the Errors:**

  Error: The condition in the inner while loop is incorrect.
  Fix: Change while(sum==0) to while(sum!=0) to ensure that the loop processes digits correctly.

  Error: The calculation of s in the inner loop is incorrect.
  Fix: Change s=s*(sum/10) to s=s+(sum%10) to correctly sum the digits.

  Error: The order of operations in the inner while loop is incorrect.
  Fix: Reorder the operations to s=s+(sum%10); sum=sum/10; to correctly accumulate the digit sum.

```
import java.util.*;
public class MagicNumberCheck
{
   public static void main(String args[])
   {
      Scanner ob=new Scanner(System.in);
      System.out.println("Enter the number to be checked.");
      int n=ob.nextInt();
      int sum=0,num=n;
      while(num>9)
      {
         sum=num;
         int s=0;
         while(sum!=0)
         {
            s=s+(sum%10);
            sum=sum/10;
         }
         num=s;
      }
      if(num==1)
      {
         System.out.println(n+" is a Magic Number.");
      }
```

```
      else
      {
         System.out.println(n+" is not a Magic Number.");
      }
   }
}
```

Merge Sort: Errors and Fixes

1. **How many errors are there in the program?**
   There are **3 errors** in the program.
2. **How many breakpoints do you need to fix these errors?**
   We need **2 breakpoints** to fix these errors.

- **Steps Taken to Fix the Errors:**

  Error: Incorrect array indexing when splitting the array in mergeSort.
  Fix: Change int[] left = leftHalf(array+1) to int[] left = leftHalf(array) and int[] right = rightHalf(array-1) to int[] right = rightHalf(array) to pass the array correctly.

  Error: Incorrect increment and decrement in merge.
  Fix: Remove the ++ and -- from merge(array, left++, right--) and instead use merge(array, left, right) to pass the arrays directly.

  Error: The array access in the merge function is incorrectly accessing beyond the array bounds.
  Fix: Ensure the array boundaries are respected by adjusting the indexing in the merging logic.

```
import java.util.*;

public class MergeSort {
   public static void main(String[] args) {
      int[] list = {14, 32, 67, 76, 23, 41, 58, 85};
      System.out.println("before: " + Arrays.toString(list));
      mergeSort(list);
```

```
    System.out.println("after:  " + Arrays.toString(list));
}

public static void mergeSort(int[] array) {
    if (array.length > 1) {
        int[] left = leftHalf(array);
        int[] right = rightHalf(array);

        mergeSort(left);
        mergeSort(right);

        merge(array, left, right);
    }
}

public static int[] leftHalf(int[] array) {
    int size1 = array.length / 2;
    int[] left = new int[size1];
    for (int i = 0; i < size1; i++) {
        left[i] = array[i];
    }
    return left;
}

public static int[] rightHalf(int[] array) {
    int size1 = (array.length + 1) / 2;
    int size2 = array.length - size1;
    int[] right = new int[size2];
    for (int i = 0; i < size2; i++) {
        right[i] = array[i + size1];
    }
    return right;
}

public static void merge(int[] result,
                         int[] left, int[] right) {
    int i1 = 0;
```

```
      int i2 = 0;

      for (int i = 0; i < result.length; i++) {
         if (i2 >= right.length || (i1 < left.length &&
               left[i1] <= right[i2])) {
            result[i] = left[i1];
            i1++;
         } else {
            result[i] = right[i2];
            i2++;
         }
      }
   }
}
```

Matrix Multiplication: Errors and Fixes

1. **How many errors are there in the program?**
   There is **1 error** in the program.
2. **How many breakpoints do you need to fix this error?**
   We need **1 breakpoint** to fix this error.

- **Steps Taken to Fix the Error:**

  Error: Incorrect array indexing in the matrix multiplication logic.
  Fix: Change first[c-1][c-k] and second[k-1][k-d] to first[c][k] and second[k][d]. These changes ensure that matrix elements are correctly referenced during multiplication.

```
public class MainClass {
  public static void main(String[] args) {
    int nDisks = 3;
    doTowers(nDisks, 'A', 'B', 'C');
  }
```

```
  public static void doTowers(int topN, char from,
  char inter, char to) {
    if (topN == 1){
      System.out.println("Disk 1 from "
      + from + " to " + to);
    }else {
      doTowers(topN - 1, from, to, inter);
      System.out.println("Disk "
      + topN + " from " + from + " to " + to);
      doTowers(topN - 1, inter, from, to);
    }
  }
}
```

Quadratic Probing Hash Table

- **Errors and Fixes:**

  How many errors are there in the program?
  There is 1 error in the program.

  How many breakpoints do you need to fix this error?
  We need 1 breakpoint to fix this error.

- **Steps Taken to Fix the Error:**

Error: In the insert method, the line i += (i + h / h--) % maxSize; is incorrect.

Fix: The correct logic should be i = (i + h * h++) % maxSize; to correctly implement quadratic probing.

```
import java.util.Scanner;

class QuadraticProbingHashTable {
   private int currentSize, maxSize;
   private String[] keys;
   private String[] vals;

   public QuadraticProbingHashTable(int capacity) {
      currentSize = 0;
      maxSize = capacity;
      keys = new String[maxSize];
      vals = new String[maxSize];
   }

   public void makeEmpty() {
      currentSize = 0;
      keys = new String[maxSize];
      vals = new String[maxSize];
   }

   public int getSize() {
      return currentSize;
   }

   public boolean isFull() {
      return currentSize == maxSize;
   }

   public boolean isEmpty() {
      return getSize() == 0;
   }

   public boolean contains(String key) {
      return get(key) != null;
   }

   private int hash(String key) {
```

```
    return key.hashCode() % maxSize;
}

public void insert(String key, String val) {
    int tmp = hash(key);
    int i = tmp, h = 1;
    do {
        if (keys[i] == null) {
            keys[i] = key;
            vals[i] = val;
            currentSize++;
            return;
        }
        if (keys[i].equals(key)) {
            vals[i] = val;
            return;
        }
        i = (i + h * h++) % maxSize; // Fixed quadratic probing
    } while (i != tmp);
}

public String get(String key) {
    int i = hash(key), h = 1;
    while (keys[i] != null) {
        if (keys[i].equals(key))
            return vals[i];
        i = (i + h * h++) % maxSize;
    }
    return null;
}

public void remove(String key) {
    if (!contains(key))
        return;

    int i = hash(key), h = 1;
    while (!key.equals(keys[i]))
```

```
            i = (i + h * h++) % maxSize;

        keys[i] = vals[i] = null;
        currentSize--;

        for (i = (i + h * h++) % maxSize; keys[i] != null; i = (i + h * h++) % maxSize) {
            String tmp1 = keys[i], tmp2 = vals[i];
            keys[i] = vals[i] = null;
            currentSize--;
            insert(tmp1, tmp2);
        }
    }

    public void printHashTable() {
        System.out.println("\nHash Table:");
        for (int i = 0; i < maxSize; i++)
            if (keys[i] != null)
                System.out.println(keys[i] + " " + vals[i]);
        System.out.println();
    }
}

public class QuadraticProbingHashTableTest {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Hash Table Test\n\n");
        System.out.println("Enter size");
        QuadraticProbingHashTable qpht = new
QuadraticProbingHashTable(scan.nextInt());

        char ch;
        do {
            System.out.println("\nHash Table Operations\n");
            System.out.println("1. insert ");
            System.out.println("2. remove");
            System.out.println("3. get");
            System.out.println("4. clear");
```

```
        System.out.println("5. size");

        int choice = scan.nextInt();
        switch (choice) {
          case 1:
            System.out.println("Enter key and value");
            qpht.insert(scan.next(), scan.next());
            break;
          case 2:
            System.out.println("Enter key");
            qpht.remove(scan.next());
            break;
          case 3:
            System.out.println("Enter key");
            System.out.println("Value = " + qpht.get(scan.next()));
            break;
          case 4:
            qpht.makeEmpty();
            System.out.println("Hash Table Cleared\n");
            break;
          case 5:
            System.out.println("Size = " + qpht.getSize());
            break;
          default:
            System.out.println("Wrong Entry \n");
            break;
        }
        qpht.printHashTable();

        System.out.println("\nDo you want to continue (Type y or n) \n");
        ch = scan.next().charAt(0);
      } while (ch == 'Y' || ch == 'y');
    }
}
```

Sorting Array

- **Errors and Fixes:**

  How many errors are there in the program?
  There are 2 errors in the program.

  How many breakpoints do you need to fix this error?
  We need 2 breakpoints to fix these errors.

- **Steps Taken to Fix the Errors:**

  Error 1: The loop condition for (int i = 0; i >= n; i++); is incorrect.

  Fix 1: Change it to for (int i = 0; i < n; i++) to correctly iterate over the array.

  Error 2: The condition in the inner loop if (a[i] <= a[j]) should be reversed.

  Fix 2: Change it to if (a[i] > a[j]) to correctly sort the array in ascending order.

```
import java.util.Scanner;

public class Ascending_Order {
   public static void main(String[] args) {
      int n, temp;
      Scanner s = new Scanner(System.in);
      System.out.print("Enter no. of elements you want in array:");
      n = s.nextInt();
      int[] a = new int[n];
      System.out.println("Enter all the elements:");
      for (int i = 0; i < n; i++) {
         a[i] = s.nextInt();
      }

      // Corrected sorting logic
      for (int i = 0; i < n; i++) {
```

```
        for (int j = i + 1; j < n; j++) {
            if (a[i] > a[j]) { // Fixed comparison
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }

    System.out.print("Ascending Order: ");
    for (int i = 0; i < n - 1; i++) {
        System.out.print(a[i] + ", ");
    }
    System.out.print(a[n - 1]);
  }
}
```

Stack Implementation (from Stack Implementation.txt)(Stack Implementation)

- **Errors and Fixes:**

  How many errors are there in the program?
  There are 2 errors in the program.

  How many breakpoints do you need to fix this error?
  We need 2 breakpoints to fix these errors.

- **Steps Taken to Fix the Errors:**

  Error 1: In the push method, the line top-- is incorrect.

  Fix 1: Change it to top++ to correctly increment the stack pointer.

  Error 2: In the display method, the loop condition for (int i=0; i>top; i++) is incorrect.

Fix 2: Change it to for (int i=0; i<=top; i++) to correctly display all elements.

```
public class StackMethods {
   private int top;
   int size;
   int[] stack;

   public StackMethods(int arraySize) {
      size = arraySize;
      stack = new int[size];
      top = -1;
   }

   public void push(int value) {
      if (top == size - 1) {
         System.out.println("Stack is full, can't push a value");
      } else {
         top++; // Fixed increment
         stack[top] = value;
      }
   }

   public void pop() {
      if (!isEmpty()) {
         top--;
      } else {
         System.out.println("Can't pop...stack is empty");
      }
   }

   public boolean isEmpty() {
      return top == -1;
   }

   public void display() {
      for (int i = 0; i <= top; i++) { // Corrected loop condition
         System.out.print(stack[i] + " ");
```

```
        }
        System.out.println();
    }
}

public class StackReviseDemo {
    public static void main(String[] args) {
        StackMethods newStack = new StackMethods(5);
        newStack.push(10);
        newStack.push(1);
        newStack.push(50);
        newStack.push(20);
        newStack.push(90);

        newStack.display();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display();
    }
}
```

Tower of Hanoi (from Tower of Hanoi.txt)(Tower of Hanoi)

- **Errors and Fixes:**

  How many errors are there in the program?
  There is 1 error in the program.

  How many breakpoints do you need to fix this error?
  We need 1 breakpoint to fix this error.

- **Steps Taken to Fix the Error:**

  Error: In the recursive call doTowers(topN ++, inter--, from+1, to+1);, incorrect increments and decrements are applied to the variables.

  Fix: Change the call to doTowers(topN - 1, inter, from, to); for proper recursion and to follow the Tower of Hanoi logic.

```
public class MainClass {
   public static void main(String[] args) {
      int nDisks = 3;
      doTowers(nDisks, 'A', 'B', 'C');
   }

   public static void doTowers(int topN, char from, char inter, char to) {
      if (topN == 1) {
         System.out.println("Disk 1 from " + from + " to " + to);
      } else {
         doTowers(topN - 1, from, to, inter);
         System.out.println("Disk " + topN + " from " + from + " to " + to);
         doTowers(topN - 1, inter, from, to); // Corrected recursive call
      }
   }
}
```