Benchmark

Using Benchmark SDK

> **WARNING**
>
> Before using the SDK you need to instantiate it using `sdk_init::benchmark()` - example provided on this page

Benchmarking

> **WARNING**
>
> Do not keep the benchmark in release builds
> Benchmarking is only available to verified developers account

> **TIP**
>
> Preferably you want to add an #ifdef to remove benchmarking completely from release builds

There is a macro defined to add a new benchmark (BENCHMARK_ADD)

```cpp
#define BENCHMARK

void __fastcall draw_world()
{
#ifdef BENCHMARK
    BENCHMARK_ADD( "awareness_draw_world" );

    benchmark->start();
#endif

    world::heroes::draw();
    world::towers::draw();
    world::traps::draw();

    for( auto& hero_info : awareness_heroes )
        spell_hud::draw( hero_info );

#ifdef BENCHMARK
    benchmark->stop();
#endif
}
```

Previous page
Notification

Clock Facade

Getting Current Game Time

INFO

This function returns the game time in **seconds**

```
float g_sdk->clock_facade->get_game_time()
```

Previous page
Object Manager

Next page
Hud Manager

Damage

Using Damage SDK

WARNING

Before using the SDK you need to instantiate it using `sdk_init::damage()` - example provided on this page

WARNING

This part of the SDK is prone to change and not completely supported yet

Damage Enums

cpp

```cpp
enum class damage_type
{
    physical = 0,
    magical,
};
```

Getting Spell Damage

Spells that do not scale on target

```
float sdk::damage->get_spell_damage( game_object* hero, int spell_slot )
```

Spells that scale on target

```
float sdk::damage->get_spell_damage( game_object* source, game_object* target, int spell_slot )
```

Getting Auto Attack Damage

```
float sdk::damage->get_aa_damage( game_object* source, game_object* target, bool next_attack = false )
```

Calculating Damage

```
float sdk::damage->calc_damage( damage_type type, game_object* source, game_object* target, float damage )
```

Previous page
Health Prediction

Next page
Infotab

Evade

Using Evade SDK

WARNING

Before using the SDK you need to instantiate it using `sdk_init::evade()` - example provided on this page

Getting Evade State

INFO

This function returns true if Evade is currently evading

```
bool sdk::evade->is_evading()
```

▶ **Example**

Cast Safely

INFO

This function returns true if the spell cast will not interfere with evading

```
bool sdk::evade->can_spell( int spell_slot, float cast_time )
```

Dash Safely

INFO

This function returns true if the dash will not interfere with evading

```
bool sdk::evade->can_dash( const math::vector3& pos, float
dash_speed, float cast_time = 0.f )
```

Knowing if Position is Safe

INFO

This function returns true if the world position is outside a spell

```
bool sdk::evade->is_position_safe( const math::vector3& pos )
```

Knowing if a Spell is dangerous

INFO

This function returns true if the Spell is dangerous
This requires a sm_sdk::spell* from Spell Manager

```
bool sdk::evade->is_dangerous_spell( sm_sdk::spell* spell )
```

▶ **Example**

Knowing if the Player is inside a Dangerous Spell

INFO

This function returns true if the Player is in a Dangerous Spell

```
bool sdk::evade->is_player_inside_dangerous_spell()
```

▶ **Example**

Getting Spell Intersection Time

INFO

This function returns the spell intersection time
This requires a sm_sdk::spell* from Spell Manager

WARNING

The function returns -1.f if the spell will never collide

`float sdk::evade->get_spell_intersection_time( const math::vector3& start_pos, const math::vector3& end_pos, float speed, sm_sdk::spell* spell )`

▶ **Example**

Evade Event Registering

In order to register to an evade event, `sdk::evade->register_callback` must be called inside `PluginLoad`

Example:

cpp

```
bool before_move()
{
    g_sdk->log_console( "Evade is going to issue a move order" );

    return true;
}

extern "C" __declspec( dllexport ) bool PluginLoad( core_sdk* sdk, void
{
    g_sdk = sdk;

    if ( !sdk_init::evade() )
        return false;

    sdk::evade->register_callback( evade_sdk::before_move, reinterpret_

    g_sdk->log_console( "[+] ExampleModule loaded!" );

    return true;
}
```

Evade Event Unregistering

In order to unregister from an evade event, `sdk::evade->unregister_callback` must be called inside `PluginUnload`

cpp

```cpp
extern "C" __declspec( dllexport ) void PluginUnload()
{
    sdk::evade->unregister_callback( evade_sdk::before_move, reinterpre

    g_sdk->log_console( "[-] ExampleModule unloaded!" );
}
```

Evade Events

cpp

```cpp
enum event_type: uint8_t
{
 before_move = 0,
};
```

INFO

The return value should be `false` if we want to block move

Previous page
Spell Manager

Next page
Health Prediction

Events

Event Registering

In order to register to a game event, `g_sdk->event_manager->register_callback` must be called inside `PluginLoad`

Example:

cpp

```cpp
void __fastcall present()
{
    g_sdk->log_console( "[+] Presenting the frame!" );
}

extern "C" __declspec( dllexport ) bool PluginLoad( core_sdk* sdk, void
{
    g_sdk = sdk;

    g_sdk->event_manager->register_callback( event_manager::event::pres

    g_sdk->log_console( "[+] ExampleModule loaded!" );

    return true;
}
```

Event Unregistering

In order to unregister from a game event, `g_sdk->event_manager->unregister_callback` must be called inside `PluginUnload`

cpp

```cpp
extern "C" __declspec( dllexport ) void PluginUnload()
{
    g_sdk->event_manager->unregister_callback( event_manager::event::pr

    g_sdk->log_console( "[-] ExampleModule unloaded!" );
}
```

event::present

INFO

Triggers every rendering frame - should only be used for **UI drawings**

TIP

The following example draws a white circle in the middle of the screen

cpp

```cpp
void __fastcall present()
{
    const auto width = g_sdk->renderer->get_window_width();
 const auto height = g_sdk->renderer->get_window_height();
 math::vector2 screen_center { static_cast< float >( width ) * 0.5f , s

 g_sdk->renderer->add_circle_2d( screen_center, 20.f, 1.f, 0xFFFFFFFF )
}
```

event::wndproc

INFO

Triggers when the game receives a window message
More info

TIP

The following example prints a console message whenever LMB is pressed

cpp

```cpp
void __fastcall wndproc( uint32_t msg, uint32_t wparam, uint32_t lparam
{
 if ( msg == WM_LBUTTONDOWN )
 {
  g_sdk->log_console( "[+] LMB was pressed" );
 }
}
```

event::game_update

INFO

Triggers every game frame - should only be used for **module logic**

TIP

The following example prints a console message while the player is moving

cpp

```cpp
void __fastcall game_update()
{
    const auto player = g_sdk->object_manager->get_local_player();
    if ( player && player->is_moving() )
    {
        g_sdk->log_console( "[+] Moving!" );
    }
}
```

event::create_object

INFO

Triggers for each object creation

TIP

The following example logs the name of the created object

cpp

```cpp
void __fastcall create_object( game_object* object )
{
    g_sdk->log_console( "[+] Object %s created", object->get_name().c_s
}
```

event::delete_object

INFO

Triggers for each object deletion

TIP

The following example logs the name of the deleted object

cpp

```
void __fastcall delete_object( game_object* object )
{
    g_sdk->log_console( "[-] Object %s deleted", object->get_name().c_s
}
```

event::create_missile

INFO

Triggers for each missile creation

TIP

The following example logs the name and the attack id of the cast linked to the missile

cpp

```
void __fastcall create_missile( game_object* missile )
{
    const auto spell_cast = missile->get_missile_spell_cast();
    if ( spell_cast )
    {
        const auto attack_id = spell_cast->get_attack_id();
        const auto name = spell_cast->get_spell_data()->get_static_data

        g_sdk->log_console( "[+] Missile linked to attack id %d created
    }
}
```

event::basic_attack

INFO

Triggers when an AI object starts the cast of a basic attack

TIP

The following example logs the names of both the source and the target of the attack

cpp

```
void __fastcall basic_attack( game_object* object, game_object* target,
{
    g_sdk->log_console( "[+] %s started attacking %s", object->get_name
}
```

event::stop_cast

INFO

Triggers when an AI stops the cast

TIP

The following example logs the name of the cast that got stopped

cpp

```
void __fastcall stop_cast( game_object* object, spell_cast* cast, bool
{
    const auto name = cast->get_spell_data()->get_static_data()->get_na

    g_sdk->log_console( "[+] %s stopped the cast %s (processed: %d)", o
}
```

event::process_cast

INFO

Triggers when an AI starts the cast of a spell

TIP

The following example logs the name of the cast that was started

cpp

```
void __fastcall process_cast( game_object* object, spell_cast* cast )
{
    const auto name = cast->get_spell_data()->get_static_data()->get_na

    g_sdk->log_console( "[+] %s started casting %s", object->get_name()
}
```

event::buff_gain

INFO

Triggers when an AI gains a buff

TIP

The following example logs the name of the gained buff

cpp

```
void __fastcall buff_gain( game_object* object, buff_instance* buff )
{
 g_sdk->log_console( "[+] %s gained buff %s", object->get_name().c_str(
}
```

event::buff_loss

INFO

Triggers when an AI loses a buff

TIP

The following example logs the name of the lost buff

cpp

```
void __fastcall buff_loss( game_object* object, buff_instance* buff )
{
 g_sdk->log_console( "[+] %s lost buff %s", object->get_name().c_str(),
}
```

event::draw_world

INFO

Triggers every world rendering frame - should only be used for **world layer drawings**

TIP

The following example draws a 3D white circle around the player

cpp

```
void __fastcall draw_world()
{
 const auto player = g_sdk->object_manager->get_local_player();
 if ( player )
 {
  auto position = player->get_position();
  g_sdk->renderer->add_circle_3d( position, player->get_bounding_radius
 }
}
```

event::neutral_minion_kill

INFO

Triggers every time a jungle camp monster gets killed

TIP

The following example logs information about the killed monster

cpp

```
void __fastcall neutral_minion_kill( game_object* object, game_object* 
{
 const auto ally_camp = camp_side_team_id == object->get_team_id();

 g_sdk->log_console( "[+] %s killed a monster (%s) from an %s camp", ob
         minion->get_name().c_str(), ally_camp ? "allied" : "enemy" );
}
```

event::new_path

INFO

Triggers every time an AI changes its path

TIP

The following example logs the speed of the dash when an AI dashes

cpp

```
void __fastcall new_path( game_object* object, bool is_dash, float dash
{
 if ( is_dash )
 {
  g_sdk->log_console( "[+] %s has dashed with %.02f speed", object->get
 }
}
```

event::execute_cast

INFO

Triggers when an AI finishes the active cast

TIP

The following example logs the name of the cast that was finished

cpp

```
void __fastcall execute_cast( game_object* object, spell_cast* cast )
{
    const auto name = cast->get_spell_data()->get_static_data()->get_na

    g_sdk->log_console( "[+] %s finished casting %s", object->get_name(
}
```

event::issue_order

DANGER

This event does not trigger for `issue_order` called from modules (including `Orbwalker`)

WARNING

This is a preventable event, the return value determines whether the order will be allowed or not

INFO

Triggers when the local player tries to issue an order

TIP

The following example prevents `game_object_order::stop` orders from being executed

cpp

```cpp
bool __fastcall issue_order( game_object* object, game_object_order ord
{
 if ( order_type == game_object_order::stop )
 {
  g_sdk->log_console( "[+] Stop order prevented!" );
  return false;
 }

 return true;
}
```

event::cast_spell

DANGER

This event does not trigger for `cast_spell` called from modules

WARNING

This is a preventable event, the return value determines whether the cast will be allowed or not

INFO

Triggers when the local player tries to cast a spell

TIP

The following example prevents the spell on slot `0` from being casted

cpp

```cpp
bool __fastcall cast_spell( game_object* object, int spell_slot, math::
{
 if ( spell_slot == 0 )
 {
  g_sdk->log_console( "[+] Prevented the cast of spell 0!" );
  return false;
 }

 return true;
}
```

event::packet

WARNING

This event is only enabled on `DEVELOPER` builds

INFO

Triggers when the client receives a network packet

TIP

The following example logs the opcode of the received packet

cpp

```
void __fastcall packet( uint16_t packet_opcode, uint64_t packet_data, u
{
    g_sdk->log_console( "[+] Received packet with opcode 0x%x", packet_
}
```

event::animation

INFO

Triggers when an AI plays an animation

TIP

The following example logs the hash of the played animation

cpp

```
void __fastcall animation( game_object* object, uint32_t animation_hash
{
 g_sdk->log_console( "[+] %s plays animation 0x%x", object->get_name().
}
```

event::cast_heal

INFO

Triggers when an AI object casts heal

TIP

The following example logs the source, the target and the amount of heal that was casted

cpp

```
void __fastcall cast_heal( game_object* object, game_object* target, fl
{
 g_sdk->log_console( "[+] %s casted heal on %s (amount: %.02f)", object
         target->get_name().c_str(), amount );
}
```

Previous page
Logging

Next page
Menu

Game Info

Getting the Game Mode

INFO

This function returns the name of the current game mode

```
char* g_sdk->game_info->get_game_mode()
```

Previous page
Nav Mesh

Next page
Setup

Getting Started

Information to help you setup the module project

Requirements

Visual Studio 2022
VEN SDK

Creating New Project

1. Start **Visual Studio 2022** and press "**Create a new project**"

2. Select "**C++ Empty Project**" and press "**Next**"

3. Change the name and location of the project and press "**Create**"

4. Right click on the created project and press "**Add -> New Item**"

5. Enter the desired name of the file that will contain the entry point of the module (f.e. "**main.cpp**")

Configuring the Project

1. Right click on the project name and press "**Properties**"

2. Select "**General**" and change the following settings to:
   - "**Output Directory**": `$(LOCALAPPDATA)\VEN\League\Modules\`
   - "**Configuration Type**": `Dynamic Library (.dll)`
   - "**C++ Language Standard**": `ISO C++20 Standard (/std:c++20)`

3. Select "**C/C++ -> All Options**" and change the following settings to:
   - "**Runtime Library**": `Multi-threaded (/MT)`

- "**SDL checks**": `No (/sdl-)`

4. Copy the "**sdk**" directory from the downloaded **VEN SDK** archive to the project root directory

5. Select "**VC++ Directories**" and change "**Include Directories**" to
   `$(ProjectDir)sdk;$(IncludePath)`

Module Properties

Export properties that core can read and load the module properly

cpp

```cpp
extern "C" __declspec( dllexport ) int SDKVersion = SDK_VERSION;
```

Load Callback

Register "**PluginLoad**" callback that triggers whenever core loads the module

cpp

```cpp
extern "C" __declspec( dllexport ) bool PluginLoad( core_sdk* sdk, void
{
    g_sdk = sdk;

    g_sdk->log_console( "[+] ExampleModule loaded!" );

    return true;
}
```

Unload Callback

Register "**PluginUnload**" callback that triggers whenever core unloads the module

cpp

```cpp
extern "C" __declspec( dllexport ) void PluginUnload()
{
    g_sdk->log_console( "[-] ExampleModule unloaded!" );
}
```

Example main.cpp

Example file declaring module load & unload callbacks

cpp

```
#include <Windows.h>
#include "sdk.hpp"

extern "C" __declspec( dllexport ) int SDKVersion = SDK_VERSION;

extern "C" __declspec( dllexport ) bool PluginLoad( core_sdk* sdk, void
{
    g_sdk = sdk;

    g_sdk->log_console( "[+] ExampleModule loaded!" );

    return true;
}

extern "C" __declspec( dllexport ) void PluginUnload()
{
    g_sdk->log_console( "[-] ExampleModule unloaded!" );
}
```

Summary

Now that we setup the module project, we can go in game and run it.

On next page, we will dive deeper into the SDK with functionalities such as registering and unregistering from events.

Previous page
Welcome

Next page
Logging

Health Prediction

Using Health Prediction SDK

WARNING

Before using the SDK you need to instantiate it using `sdk_init::health_prediction()` - example provided on this page

Predict Object Health

`float sdk::health_prediction->get_predicted_health( game_object* target, const float time )`

INFO

The `time` argument is in seconds from the start of the game - like g_sdk->clock_facade->get_game_time()

Example predicting hovered target health in 0.25 seconds

cpp

```cpp
const auto hovered_target = g_sdk->hud_manager->get_hovered_tar
if ( hovered_target )
{
    const auto game_time = g_sdk->clock_facade->get_game_time()
    const auto predicted_health = sdk::health_prediction->get_p
    g_sdk->log_console( "%s health will be %.2f (currently %.2f
}
```

Getting an Ally Minion Focus

To get the ally minion focus you may use the function

```
game_object* sdk::health_prediction->get_minion_focus( game_object*
source )
```

Previous page
Evade

Next page
Damage

Hud Manager

Getting Current Cursor Position

INFO

This function returns the position in 3D world coordinates

```
math::vector3 g_sdk->hud_manager->get_cursor_position()
```

▶ **Example**

Getting Currently Hovered Target

INFO

This function returns the currently hovered target

```
game_object* g_sdk->hud_manager->get_hovered_target()
```

▶ **Example**

Welcome

> **WARNING**
>
> You are using an early version of the SDK which is prone to major ABI changes.

Welcome to the documentation page of VEN SDK - here you will find all the required information to start writing modules for VEN.

Next page will cover all the information related to the setup of the module project.

Next page
Getting Started

Infotab

Using Infotab SDK

WARNING

Before using the SDK you need to instantiate it using `sdk_init::infotab()` - example provided on this page

Adding an Infotab Display Entry

`uint32_t sdk::infotab->add_text( const infotab_sdk::text_entry& title, const std::function< infotab_sdk::text_entry() >& fn )`

WARNING

This function should only be used in `PluginLoad`
You must call `sdk::infotab->remove_text` on `PluginUnload` using the ID returned by this function

Example

cpp

```cpp
infotab_id = sdk::infotab->add_text( { "Test" }, [ ]() -> infot
{
    infotab_sdk::text_entry entry{};

    entry.text = "HEY";
    entry.color = 0xFFFF0000;

    return entry;
} );
```

Adding an Infotab Hotkey Entry

```
uint32_t sdk::infotab->add_hotkey_text( std::string* hotkey, const
infotab_sdk::text_entry& title, const std::function<
infotab_sdk::text_entry() >& fn );
```

WARNING

This function should only be used in `PluginLoad`
You must call `sdk::infotab->remove_text` on `PluginUnload` using the ID returned by this function

Example

cpp

```
infotab_id = sdk::infotab->add_text( orb::spell_farm_key, { "Sp
{
    infotab_sdk::text_entry entry{};

    if( orb::allow_spell_farm )
    {
        entry.text = "ON";
        entry.color = 0xFF00FF00;
    }
    else
    {
        entry.text = "OFF";
        entry.color = 0xFFFF0000;
    }

    return entry;
} );
```

Removing an Infotab Entry

```
void sdk::infotab->remove_text( uint32_t id )
```

WARNING

This function should only be used in `PluginUnload`
The `id` argument must be the one that returned from the `sdk::infotab->add_text` function

Example

cpp

```
extern "C" __declspec(dllexport) void __fastcall PluginUnload()
{
    sdk::infotab->remove_text( infotab_id );
}
```

Previous page
Damage

Next page
Notification

Logging

INFO

As you can see, there are 3 types of prefixes that change the color of the log:

`ANY` - Info
`[!]` - Error
`[?]` - Warning

Info

cpp
```
g_sdk->log_console( "Hello from logger!" );
```

Error

cpp
```
g_sdk->log_console( "[!] Hello from logger!" );
```

Warning

cpp
```
g_sdk->log_console( "[?] Hello from logger!" );
```

Menu

INFO

Callbacks passed to the menu elements are called when their value changes
As an extra - they are also called when they are being created for initialization purpose

Creating a New Menu

First things first, a category needs to be created, so we're gonna use `menu_category* g_sdk->menu_manager->add_category( std::string const& name, std::string const& display_name )`

`name` must be unique so config won't be accidentally shared with other modules `display_name` will be the name of the category that will be displayed in the menu

▶ **Example**

Adding a Label

INFO

This function adds a label in a category/sub-category

```
void menu_category::add_label( std::string const& text )
```

▶ **Example**

Adding a Checkbox

INFO

This function adds a checkbox in a category/sub-category

```
void menu_category::add_checkbox( std::string const& element_name, std::string const& element_display_name, bool default_value, std::function< void( bool ) > const& callback )
```

▶ **Example**

Adding a Hotkey

INFO

This function adds a hotkey in a category/sub-category

```
void menu_category::add_hotkey( std::string const& element_name, std::string const& element_display_name, unsigned char default_key, bool default_value = false, bool toggle = false, std::function< void( std::string*, bool ) > const& callback = nullptr )
```

▶ **Example**

Adding a Separator

INFO

This function adds a separator in a category/sub-category

```
void menu_category::add_separator()
```

▶ **Example**

Adding a Slider Int

INFO

This function adds a slider int in a category/sub-category

```
void menu_category::add_slider_int( std::string const& element_name, std::string const& element_display_name, int min, int max, int step, int default_value, std::function< void( int ) > const& callback = nullptr )
```

▶ **Example**

Adding a Slider Float

INFO

This function adds a slider float in a category/sub-category

```
void menu_category::add_slider_float( std::string const& element_name, std::string const& element_display_name, float min, float max, float step, float default_value, std::function< void( float ) > const& callback = nullptr )
```

▶ **Example**

Adding a Combo

INFO

This function adds a combo in a category/sub-category

```
void menu_category::add_combo( std::string const& element_name, std::string const& element_display_name, std::vector< std::string > const& items, int default_value, const std::function< void( int ) >& callback = nullptr )
```

▶ **Example**

Adding a Colorpicker

INFO

This function adds a colorpicker in a category/sub-category

```
void menu_category::add_colorpicker( std::string const& element_name, std::string const& element_display_name, uint32_t default_color, std::function< void( uint32_t ) > const& callback = nullptr )
```

▶ **Example**

Adding a Sub-category

INFO

This function adds a sub-category in a category/sub-category

```
menu_category* menu_category::add_sub_category( std::string const& element_name, std::string const& element_display_name )
```

▶ **Example**

Previous page
Events

Next page
Object Manager

Nav Mesh

Checking if Terrain is Pathable

INFO

This function returns whether or not a terrain position is a valid path target

```
bool g_sdk->nav_mesh->is_pathable( math::vector3& position )
```

▶ **Example**

Checking if Position is in FOW

INFO

This function returns whether or not a world position is in Fog of War

```
bool g_sdk->nav_mesh->is_in_fow( math::vector3& position )
```

▶ **Example**

Net Client

Getting the Ping

INFO

This function returns the current ping in **milliseconds**

```
int g_sdk->net_client->get_ping()
```

▶ **Example**

Notification

Using Notification SDK

> **WARNING**
>
> Before using the SDK you need to instantiate it using `sdk_init::notification()` - example provided on this page

Displaying Notification

```
void sdk::notification->add( const std::string& title, const std::string& content, const color& clr = 0xfff0e6d2 )
```

▶ **Example**

Object Manager

Getting Local Player

INFO

While in replay mode - this function returns the currently selected player; otherwise returns the controlled player

```
game_object* g_sdk->object_manager->get_local_player()
```

▶ **Example**

Getting Objects by Type

```
std::span< game_object* > g_sdk->object_manager->get_turrets()
std::span< game_object* > g_sdk->object_manager->get_heroes()
std::span< game_object* > g_sdk->object_manager->get_minions()
std::span< game_object* > g_sdk->object_manager->get_nexuses()
std::span< game_object* > g_sdk->object_manager->get_inhibitors()
std::span< game_object* > g_sdk->object_manager->get_monsters()
std::span< game_object* > g_sdk->object_manager->get_traps()
std::span< game_object* > g_sdk->object_manager->get_wards()
std::span< game_object* > g_sdk->object_manager->get_plants()
```

▶ **Example**

Getting Object by Network Id

```
game_object* get_object_by_network_id( uint32_t network_id )
```

Orbwalker

Using Orbwalker SDK

> **WARNING**
>
> Before using the SDK you need to instantiate it using `sdk_init::orbwalker()` - example provided on this page

Modes

Below are all the modes that orbwalker uses:

cpp

```cpp
bool sdk::orbwalker->combo()
bool sdk::orbwalker->harass()
bool sdk::orbwalker->clear()
bool sdk::orbwalker->lasthit()
bool sdk::orbwalker->flee()
bool sdk::orbwalker->freeze()
bool sdk::orbwalker->fast_clear()
```

▶ **Example**

Spell Weaving

To do proper spell weaving you may use the function

`bool sdk::orbwalker->can_spell( game_object* target, const float time = 0.f )`

> **INFO**
>
> The target argument is required
> The time is not but it's **recommended** to put the **spell cast delay** for perfect weaving

WARNING

Note that if your attack will come back faster than the spell cast delay, the spell will not be casted which may lead to a behaviour that you did not want If you are just trying to avoid cancelling auto attacks, take a look at how to avoid cancelling attacks

▶ **Example**

Attack is Ready

To know if the attack will be ready (now or later) you may use the function

```
bool sdk::orbwalker->is_attack_ready_in( const float time )
```

INFO

The time argument is required, it takes the time in seconds - you can also pass 0.f if you want to know if the attack is ready **now**

▶ **Example**

Avoid Cancelling Attacks

To avoid cancelling attacks you may use the function

```
bool sdk::orbwalker->would_cancel_attack()
```

▶ **Example**

Getting the Attack Cast End Time

To get the attack cast end time you may use the function

```
float sdk::orbwalker->get_attack_cast_end_time()
```

INFO

The returned time is in seconds from the start of the game - like g_sdk->clock_facade->get_game_time()

Getting a Target in Attack Range

To get a target in attack range using the orbwalker you may use the function

```
game_object* sdk::orbwalker->get_target_in_attack_range()
```

WARNING

This function only return **heroes** target

▶ **Example**

Is Target in Attack Range

To know if a target is in attack range you may use the function

```
bool sdk::orbwalker->is_in_auto_attack_range( game_object* source, game_object* target, float offset = 0.f )
```

INFO

The **source** and **target** arguments are required, **source** is often the player
The **offset** argument is optional, it will add the radius to the attack range

▶ **Example**

Is Target in Attack Range (Future)

To know if a target is in attack range at a specific position you may use the function

```
bool sdk::orbwalker->is_in_auto_attack_range( game_object* source, const math::vector3& source_position, game_object* target, float offset = 0.f );
```

INFO

The **offset** argument is optional, it will add the radius to the attack range

Spell Farming

To spell farm you may use the function

```
pred_sdk::pred_data sdk::orbwalker->spell_farm( pred_sdk::spell_data spell_data, const int aoe_hits_needed, const spell_farm_flag flags = spell_farm_flag::none, const dmg_sdk::damage_type damage_type = dmg_sdk::damage_type::physical )
```

INFO

The **spell_data** argument is a prediction spell data

The **aoe_hits_needed** argument is how many hits are needed to cast the spell (note: -1 will try to find the maximum amount of hits possible)
The **flags** argument takes a spell_farm_flag
The **damage_type** argument takes a damage_type is optional, defaults to physical damage

Return is pred_data

INFO

The following example showcases how to last hit objects with the desired spell while the player's attack is not ready

WARNING

The example is not compilable due to the lack of the **q_pred_data** variable, it is supposed to be a showcase on how to use it within your module

▶ **Example using Ezreal Q**

Spell Farming Related Enums

cpp

```cpp
enum spell_farm_flag: uint8_t
{
 none = 0,
 outside_of_attack_range = 1,
 lasthit = 2,
 when_attack_not_ready = 4,
};

enum class damage_type
{
    physical = 0,
    magical,
};
```

Is Spell Farm Enabled

To know if spell farming is enabled by the player you may use the function

`bool sdk::orbwalker->is_spell_farm_enabled()`

NOTE

The function `sdk::orbwalker->spell_farm` already checks this variable internally

▶ **Example**

Is Spell Farm Lasthit Enabled

To know if spell farming lasthit is enabled by the player you may use the function

```
bool sdk::orbwalker->is_spell_farm_lasthit_enabled()
```

▶ **Example**

Forcing an Attack

To force an attack on an object you may use the function

```
bool sdk::orbwalker->attack( game_object* target )
```

DANGER

This may break kiting for one attack if used incorrectly

WARNING

The example is not compilable due to the lack of the **util::get_hero_target** and **util::cast_spell** functions (core utility), it is supposed to be a showcase on how to use it within your module

▶ **Example using Kog'Maw W**

Orbwalker Event Registering

In order to register to an orbwalker event, `sdk::orbwalker->register_callback` must be called inside `PluginLoad`

Example:

cpp

```cpp
bool before_attack( orb_sdk::event_data* data )
{
    if ( data.target )
        g_sdk->log_console( "Orbwalker will attack target: %s", data.ta

    return true;
}

extern "C" __declspec( dllexport ) bool PluginLoad( core_sdk* sdk, void
{
    g_sdk = sdk;

    if ( !sdk_init::orbwalker() )
        return false;

    sdk::orbwalker->register_callback( orb_sdk::before_attack, reinterp

    g_sdk->log_console( "[+] ExampleModule loaded!" );

    return true;
}
```

Orbwalker Event Unregistering

In order to unregister from an orbwalker event, `sdk::orbwalker->unregister_callback` must be called inside `PluginUnload`

cpp

```cpp
extern "C" __declspec( dllexport ) void PluginUnload()
{
    sdk::orbwalker->unregister_callback( orb_sdk::before_attack, reinte

    g_sdk->log_console( "[-] ExampleModule unloaded!" );
}
```

Orbwalker Events

cpp

```cpp
enum event_type: uint8_t
{
 before_attack = 0,
 before_move,
};
```

INFO

The `before_move` event does not contain any target
The return value should be `false` if we want to block the attack/move

Both events are defined the same way:

cpp

```
class event_data
{
public:
 game_object* target{};
};
```

cpp

```
bool before_attack( orb_sdk::event_data* data )
{
    if ( data.target )
        g_sdk->log_console( "Orbwalker will attack target: %s", data.ta

    return true;
}

bool before_move( orb_sdk::event_data* data )
{
    g_sdk->log_console( "Orbwalker before move" );

    return true;
}
```

Previous page
Setup

Next page
Target Selector

Prediction

Using Prediction SDK

> **WARNING**
>
> Before using the SDK you need to instantiate it using `sdk_init::prediction()` - example provided on this page

Prediction Enums

```cpp
enum hitchance: int {
    automatic = -1,
    any = 0,
    low = 30,
    medium = 50,
    high = 70,
    very_high = 85,
    guaranteed_hit = 100,
};

enum class hit_type: uint8_t {
    normal = 0,
    undodgeable,
    cast,
    zhonyas,
    cc,
    cc_hard,
    dash,
};

enum class spell_type: uint8_t {
    linear = 0,
    targetted,
    circular,
    vector,
};

/*
    basic attack range from edge to edge
    targeted skill range from center to center (mainly not always)
    skillshot range from center to edge (mainly not always)
    and the range of self-centered area of effects are from the center
*/
enum class targetting_type: uint8_t {
    center = 0,
    center_to_edge,
    edge_to_edge,
};

enum class collision_type: uint8_t {
    unit = 0,
    hero,
    turret,
    terrain,
    yasuo_wall,
    braum_wall,
};
```

Spell Data

INFO

This is the data that needs to be fed to most prediction functions

cpp

```
class spell_data {
public:
    spell_type spell_type{};
    targetting_type targetting_type{};
    int expected_hitchance = hitchance::automatic; // the expected hitcl

    game_object* source{}; // source object, if none player will be tak
    math::vector3 source_position{}; // position where the skillshot or

    bool bypass_anti_buffering{}; // allows casting spells while other
    int spell_slot = -1; // will be used to check for CD if expected_hi
    float range{}; // max range of the spell
    float radius{}; // circle is the same as linear, use * 0.5f of the
    float cast_range{}; // the cast range of the spell for vector types
    float delay{}; // cast delay
    float proc_delay{}; // delay until the spell hits, for example synd
    float projectile_speed = FLT_MAX; // projectile speed if any, FLT_M
    float extension_override{}; // if we want to override the predictio

    std::vector< collision_type > forbidden_collisions{}; // things we
    std::vector< hit_type > expected_hit_types{}; // if we want special
    std::function< bool( game_object* ) > additional_target_selection_cl
};
```

Prediction Data

INFO

This is the data returned by most prediction functions

```
struct collision_data
{
    game_object* object{};
    math::vector3 collided_position{};
};

struct collision_ret
{
    bool collided{};
    std::vector< collision_data > collided_units{};
};

class pred_data {
public:
    bool is_valid{}; // if this is true, prediction was successful we a
    game_object* target{}; // the target prediction is aiming at
    int hitchance{}; // use expected_hitchance field inside spell_data

    math::vector3 predicted_position{}; // predicted position of the ta
    math::vector3 predicted_dodge_position{}; // predicted dodge positi
    math::vector3 cast_position{}; // position to use when casting the
    math::vector3 first_cast_position{}; // for vector types: viktor E,
    float intersection_time{}; // time until the skillshot will hit the

    math::vector3 collision_pos{}; // the point where the spell collide
    std::vector< collision_data > collided_units{}; // the forbidden_co

    pred_data() {};
    pred_data( game_object* target )
    {
        this->target = target;
    }
};
```

Predicting

There are multiple functions you can use depending on your need:

Find a target and predict it

```
pred_sdk::pred_data sdk::prediction->predict(
pred_sdk::spell_data spell_data )
```

Predicts a specific target

```
pred_sdk::pred_data sdk::prediction->predict( game_object* obj,
pred_sdk::spell_data spell_data )
```

Finds a target and predicts it (for targetted spells only, used to account for wall collisions)

```
pred_sdk::pred_data sdk::prediction->targetted(
pred_sdk::spell_data spell_data )
```

▶ **Getting the collision position**

Example using Ezreal Q

cpp

```cpp
pred_sdk::spell_data q_data{};

q_data.spell_type = pred_sdk::spell_type::linear;
q_data.targetting_type = pred_sdk::targetting_type::center;
q_data.expected_hitchance = 45; // pred_sdk::hitchance::automat
q_data.spell_slot = 0;
q_data.range = 1150.f;
q_data.radius = 60.f;
q_data.delay = 0.25f;
q_data.projectile_speed = 2000.f;
q_data.forbidden_collisions =
{
 pred_sdk::collision_type::unit,
 pred_sdk::collision_type::hero,
 pred_sdk::collision_type::yasuo_wall,
 pred_sdk::collision_type::braum_wall,
};

const auto pred = sdk::prediction->predict( q_data );
if ( pred.is_valid )
{
    g_sdk->log_console( "Casting spell at %s | hitchance %d", p

    util::cast_spell( q_data.spell_slot, pred.cast_position );
}
```

Predicting on Path

WARNING

Only use this function if you cannot achieve what you want with the functions above

```
math::vector3 sdk::prediction->predict_on_path( game_object* obj,
float time, bool use_server_pos = true )
```

INFO

The `time` argument takes seconds

The `use_server_pos` argument lets you choose between client predicted position or server position

Collision

To check whether a spell collides with any forbidden_collisions you may use the function

cpp

```
struct collision_data
{
    game_object* object{};
    math::vector3 collided_position{};
};

struct collision_ret
{
    bool collided{};
    std::vector< collision_data > collided_units{};
};
```

```
collision_ret sdk::prediction->collides( const math::vector3&
end_point, pred_sdk::spell_data spell_data, const game_object*
target )
```

INFO

This is used internally inside the predicting functions

Prediction Utilities

cpp

```
float sdk::prediction->util()->get_spell_range( pred_sdk::spell_data& da
bool sdk::prediction->util()->is_in_range( pred_sdk::spell_data& data, r
float sdk::prediction->util()->get_spell_hit_time( pred_sdk::spell_data
float sdk::prediction->util()->get_spell_escape_time( pred_sdk::spell_da
```

▶ **Example**

Previous page
Target Selector

Next page
Spell Manager

Renderer

Adding a 2D Circle

INFO

This function adds a 2D circle of a specified radius, thickness and color at a screen position

```
void g_sdk->renderer->add_circle_2d( math::vector2& position, float radius, float thickness, uint32_t color )
```

▶ **Example**

Adding a Filled 2D Circle

INFO

This function adds a filled 2D circle of a specified radius and color at a screen position

```
void g_sdk->renderer->add_circle_filled_2d( math::vector2& position, float radius, uint32_t color )
```

▶ **Example**

Adding a 3D Circle

INFO

This function adds a 3D circle of a specified radius, thickness and color at a world position

```
void g_sdk->renderer->add_circle_3d( math::vector3& position, float radius, float thickness, uint32_t color )
```

▶ **Example**

Adding a Rectangle

INFO

This function adds a rectangle of a specified color and thickness

```
void g_sdk->renderer->add_rectangle( math::rect const& rectangle, uint32_t color, float thickness )
```

Adding a Filled Rectangle

INFO

This function adds a filled rectangle of a specified color

```
void g_sdk->renderer->add_rectangle_filled( math::rect const& rectangle, uint32_t color )
```

Adding Text

INFO

This function adds text of specified size and color at a 2D screen position

Available flags:

`1` - Centered Text

```
void g_sdk->renderer->add_text( std::string const& text, float size, math::vector2 position, uint32_t flags, uint32_t color )
```

▶ **Example**

Adding a 2D Line

INFO

This function adds a 2D line of a specified thickness and color from one screen point to another

```
void g_sdk->renderer->add_line_2d( math::vector2& point1,
```

```
math::vector2& point2, float thickness, uint32_t color )
```

Adding a Sprite

INFO

This function adds a sprite at a screen position

Color should be `0xFFFFFFFF` for the sprite to maintain the original color

```
void g_sdk->renderer->add_sprite( void* sprite, math::vector2 const& position, uint32_t color, math::vector2 const& scale = { 1.f , 1.f } )
```

▶ **Example**

Adding a Minimap Circle

INFO

This function adds a circle on the minimap at a world position

```
void g_sdk->renderer->add_circle_minimap( math::vector3 const& position, float radius, float thickness, uint32_t color )
```

▶ **Example**

Translating World Position to Screen

INFO

This function returns the input world position translated into screen coordinates

```
math::vector2 g_sdk->renderer->world_to_screen( math::vector3 world_position )
```

Translating World Position to Minimap

INFO

This function returns the input world position translated into minimap screen coordinates

```
math::vector2 g_sdk->renderer->world_to_minimap( math::vector3
world_position )
```

Adding Damage/Heal/Shield Indicator

INFO

The argument must be the actual damage/heal/shield value

cpp

```
void g_sdk->renderer->add_damage_indicator( game_object* object, float
void g_sdk->renderer->add_heal_indicator( game_object* object, float hea
void g_sdk->renderer->add_shield_indicator( game_object* object, float
```

Previous page
Hud Manager

Next page
Net Client

Setup

Loading the Dependencies

For every interface you want to use you need to instantiate it inside `PluginLoad` using `sdk_init::` - for example:

cpp

```cpp
extern "C" __declspec( dllexport ) bool PluginLoad( core_sdk* sdk, void
{
    g_sdk = sdk;

    if ( !sdk_init::orbwalker() )
        return false;

    g_sdk->log_console( "[+] ExampleModule loaded!" );

    return true;
}
```

This example instantiates the pointer sdk::orbwalker which can be found on this page

INFO

Every `sdk_init::` function returns a `bool` - the function will return `false` if the dependency failed to load

Spell Manager

Using Spell Manager SDK

WARNING

Before using the SDK you need to instantiate it using `sdk_init::spell_manager()` - example provided on this page

Spell Manager Enums

cpp

```cpp
enum class spell_iteration: uint8_t {
    all = 0,
    ally = 1,
    enemy = 2,
};

enum class spell_type: uint8_t {
    unsupported = 0,
    linear = 1,
    circular = 2,
};
```

Spell Class

WARNING

This class is likely to change in the future

DANGER

Those variables should NEVER be written to, else it will create exception/issues !!

```
class spell {
public:
    bool operator==( const spell& a )
    {
        return a.owner == this->owner && a.slot == this->slot;
    }

    int id{}; // Unique Spell ID created by Spell manager
    game_object* owner{}; // Spell caster
    game_object* target{}; // Target (for skills like Fizz R, Hwei R an
    game_object* missile{}; // Missile if exists (can be nullptr)
    game_object* particle{}; // Particle if exists (can be nullptr)
    math::vector3 start_pos{}; // Spell start position
    math::vector3 end_pos{}; // Spell end position
    int slot{}; // Spell slot
    spell_type type{}; // Spell type
    int team_id{}; // Caster Team ID
    bool is_drawing_only{}; // Is drawing only
    bool is_cc{}; // Is CC
    bool is_particle_on_ground{}; // Is particle on ground
    bool has_projectile{}; // If the spell will create a projectile (Do
    bool missile_created{}; // If the missile has been created
    uint32_t missile_effect_key{}; // Spell hash
    float radius{};
    float projectile_speed{}; // If the spell has a projectile speed, o
    float travel_time{}; // If the spell has a static travel time, othe
    float cast_delay{}; // Spell cast delay (not always set !)
    float cast_end_time{}; // Spell cast end time in game time
    float deletion_time{}; // Spell deletion time in game time
    float creation_time{}; // Spell creation time in game time
    std::string missile_name{}; // Spell missile name
    spell* parent_spell{}; // Parent spell (can be nullptr)
    std::vector< spell* > additional_spells{}; // Additional spells exa
    bool from_fow{}; // If the spell was casted in Fog of War

    bool allow_missile_positions = true; // For internal use
    bool delete_on_missile_deletion = true; // For internal use
    bool pending_deletion{}; // For internal use
    color color{}; // For internal use
    uint8_t previous_alpha = 0; // For internal use
};
```

Iterating Spells

INFO

This function iterates all the hero spells currently casting/travelling in the game

```
void sdk::spell_manager->iterate_spells( sm_sdk::spell_iteration
```

```
iter, const std::function< bool( sm_sdk::spell* spell ) >& fn )
```

▶ **Example**

Getting Spells Static Data

INFO

This function returns an array with sm_sdk::static_data of all spells from a specific hero

DANGER

This function should ONLY be called in `PluginLoad`
You may cache the results yourself if needed for later use in events

```
std::array< sm_sdk::static_data, 64 > sdk::spell_manager->get_spells_static_data( game_object* hero )
```

Getting a Spell Missile Position

INFO

This function returns the world position of the spell missile
This can only be called on spells that have projectiles (`spell->has_projectile`)

```
math::vector3 sdk::spell_manager->get_missile_position( sm_sdk::spell* spell )
```

▶ **Example**

Knowing if the Spell got casted

INFO

This function returns true if the spell has been casted

```
bool sdk::spell_manager->is_casted( sm_sdk::spell* spell )
```

▶ **Example**

Target Selector

Using Target Selector SDK

WARNING

Before using the SDK you need to instantiate it using `sdk_init::target_selector()` - example provided on this page

Getting an Enemy Hero Target

To get an enemy hero target you may use the function
`game_object* sdk::target_selector->get_hero_target( std::function< bool( game_object* ) > fn = {} )`

INFO

You can filter targets according to the lambda function you pass as argument
If no argument is passed to the function, the function checks by default `hero->is_visible() && hero->is_targetable()`

▶ **Example finding a target 600 units around the player**

Getting an Ally Hero Target

To get an ally hero target you may use the function
`game_object* sdk::target_selector->get_ally_hero_target( std::function< bool( game_object* ) > fn = {} )`

INFO

You can filter targets according to the lambda function you pass as argument
If no argument is passed to the function, the function checks by default `hero->is_visible() && hero->is_targetable()`

▶ **Example finding a target 600 units around the player**

Getting a Monster Target

To get a monster target you may use the function

```
game_object* sdk::target_selector->get_monster_target(
std::function< bool( game_object* ) > fn = {} )
```

INFO

You can filter targets according to the lambda function you pass as argument
If no argument is passed to the function, the function checks by default `monster->is_visible() && monster->is_targetable()`

▶ **Example finding a target 600 units around the player**

Getting the Forced Target

To get the forced target you may use this function

```
game_object* sdk::target_selector->get_forced_target()
```

▶ **Example**

Getting Sorted Heroes

To get the sorted heroes you may use this function

```
const std::span< game_object* >& sdk::target_selector-
>get_sorted_heroes()
```

Getting Sorted Monsters

To get the sorted monsers you may use this function

```
const std::span< game_object* >& sdk::target_selector-
>get_sorted_monsters()
```

Orbwalker

Next page
Prediction