User Tag List

Thread: Creating a Pixel Bot in C# from zero - "kinda" tutorial

1 › Last ›

Thread Tools Search Thread

12-03-2019 #1

Humbleguy Active Member

Some info at these foruns helped me when I need to start and did not know from where to start, and I now am giving something back to community as a "thank you" word for the help you guys silently gave me. My bot is not ready, but I have done some work and it is somehow functional, and I believe I got enough material to share some useful stuff.
I aman intermediate-level C# programmer, but I do have some knowlege on algorithims, so I decided that a "pixel-bot" would be the easier option to start. So... here is my "not-a-tutorial", but a collection of code in Lua and C# functions, on how to write a pixel bot, or... at least start one.
Update: Creating a grind bot from zero will take about 120 hours of coding if you have no knowledge about how to do it. Feel free to ask me anything you need to know to address any issue here, I am willing to help (my knowledge is restricted to C# and algorithims in general).

PART 1 - CREATING THE ADDON / WEAK AURA PIXEL DISPLAY SQUARES
----

1. What is a Pixel bot?
- A pixel bot, for those that do not know, is a bot that instead of getting game info from memory-readying (injecting wow proccess), will read info from pixels at the screen. Those pixels can be native wow pixels (e.g. health bars) or an addon-made bar. Using native pixels is somehow easier, but using an addon allows us to access more complex functions.

2. Where should I start ?
- The first step should be doing an addon, but i strongly suggest using Weak Aura. I present here two weakaura strings to be imported with "import" function from the addon,, that will draw colored pixels at the screen, for you to read with your program. Here I give you an example:

Code:

```
!TJvBVjUrq4)lr6Kivqo)(lv6(qifYLtein25stvKHf8AW354LU2obNpWV9o7U2GbSVY
```

and the second string:

Code:

```
!TR16snUXs43L8RDtbSA0fFPQKTklWMLeV2CSeHDpfLThBn2wbHKJUWc8d)SF6EMr3Tr
```

Those Weak Aura strings can be imported directly using the Weak Auras addon for wow classic.

When you make your c# coding, you need to read the colours of the squares, decode the R, G, B colors, and each color will give you a number.

IMPORTANT THING: Keep in mind that gamma and contrast might affect the effective display of the colours, so always keep contrast in 50, brightness in 50 and gamma in 1.0 in game settings or you may have wrong results. The picture at the link below shows briefly what each color component of each square means.
teste jpeg.jpg
I am no portinari, but the picture above shows what each color component of the squares from the Weak Aura code show.
[TO BE CONTINUED]

fedelis , Angler23 , Plobbi , dlablo (4 members gave Thanks to Humbleguy for this useful post)

12-03-2019 #2

Humbleguy Active Member

PART 2 - VISUAL STUDIO C# PREPARATIONS NEEDED TO CREATE THE BOT

Once you got the addon / weak aura that will diplay the squares with the player info, you are ready to start creating C# coding that will enable you to manipulate the game itself.

What "preparations" do I need?

1. A method to focus game window
2. A method to send key strokes to the game
3. A method do send mouse clicks to the game
4. A method to read and decode the RGB elements from game pixels
5. Method for "wait" without freezing the interface

As I had no idea how to do this, and my coding skills are not so great, I spend several ours at Stackoverflow until I figured that all. Not easy task, but fun for sure.

FIRST - SELECT THE "USING", DLL IMPORTS, AND KEY/MOUSE CODES

I dont remember exactly which ones were need for each of the bot methods, I will just paste all of them to make sure thing will work. you will also need to know mouse and key codes for future bot output. THose can be declared as constants.

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.VisualBasic;
using System.Runtime.InteropServices; //required for APIs
using System.Diagnostics;
```

```
    public partial class Form1 : Form
    {
        [DllImport("user32.dll")]
        private static extern bool SetForegroundWindow(IntPtr hWnd);
        [DllImport("user32.dll")]
        static extern void keybd_event(byte bVk, byte bScan, int dwF
        [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConv
        static extern void mouse_event(int dwFlags, int dx, int dy,
        private const int MOUSEEVENTF_LEFTDOWN = 0x02;
        private const int MOUSEEVENTF_LEFTUP = 0x04;
        private const int MOUSEEVENTF_RIGHTDOWN = 0x08;
        private const int MOUSEEVENTF_RIGHTUP = 0x10;
        public const int KEYEVENTF_EXTENDEDKEY = 0x0001; //Key down
        public const int KEYEVENTF_KEYUP = 0x0002; //Key up flag
        public const int QKEY = 0x51; // tecla Q
        public const int WKEY = 0x57; // tecla W
        public const int EKEY = 0x45; // tecla E
        public const int SKEY = 0x53; // tecla S
        public const int SPACEBAR = 0x20; // espaço
        public const int PLAYERSTATSX = 1574; // pixel do status do
        public const int PLAYERSTATSY = 470; // pixel do status do p
        public const int RANGEX = 1662; // pixel do status do player
        public const int RANGEY = 6003; // pixel do status do player
        public const int FACINGX = 1577; // pixel do status do playe
```

METHOD FOR WAITING WITHOUT FREEZING INTERFACE

Code:

```
        public void wait(int milliseconds) // It will wait number o
        {
            System.Diagnostics.Stopwatch sw = System.Diagnostics.Sto
            while (sw.ElapsedMilliseconds <= milliseconds)
            {
                Application.DoEvents();
            }
        }
```

METHOD FOR FOCUSING GAME WINDOW

This is important, because when C# send fake keypress or mouse move/click , it will be to the focused (active) window; then it is necessary to focus game window properly to do so. This method will focus World of Warcraft Window.

Code:

```
        void focawow() // Focus world of warcraft windows
        {
            var prc = Process.GetProcessesByName("wow");
            if (prc.Length > 0)
            {
                SetForegroundWindow(prc[0].MainWindowHandle);
            }
```

METHOD FOR MOVING THE MOUSE TO DESIRED POSITION

Code:

```
        // função que move o cursor.... (em construção)
        void mousemove(int x, int y)
        {
            this.Cursor = new Cursor(Cursor.Current.Handle);
            Cursor.Position = new Point(x, y);
        }
```

METHODS FOR SENDING KEYSTROKES TO THE GAME

This method will send keyspress to the window key; first it will call function to focus wow, above.
It will also use the "wait" method above.

Code:

```
        // envia tecla para o wow
        void aperta(byte key, int time = 50) // it will send coded k
        {
            focawow(); // this is the focust window method above
            if (time != 2) keybd_event(key, 0, KEYEVENTF_EXTENDEDKEY
            if (time != 2) wait(time); // this is wait method above
            if (time > 0) keybd_event(key, 0, KEYEVENTF_KEYUP, 0); /
        }
```

METHOD FOR SENDING MOUSE CLICKS TO THE GAME
----

code will not work without headers stated above (using, constants...)

Code:

```
        public void DoMouseClick(int botao=1) // argument is button,
        {
            //Call the imported function with the cursor's current p
            int X = Cursor.Position.X;
            int Y = Cursor.Position.Y;
            if (botao==1) mouse_event(MOUSEEVENTF_LEFTDOWN | MOUSEEV
            else if (botao==2)
                mouse_event(MOUSEEVENTF_RIGHTDOWN | MOUSEEVENTF_RIGH
        }
```

To simplify things, you can make another method to do a mouseclick at desired position:

Code:

```
        void clica(int x, int y,int botao=1)
        {
```

METHOD TO READ PIXEL COLOR AT COORDINATES X / Y

Code:

```
        Color GetColorAt(int x, int y)
        {
            Rectangle bounds = new Rectangle(x, y, 1, 1);
            using (Graphics g = Graphics.FromImage(bmp))
                g.CopyFromScreen(bounds.Location, Point.Empty, boun
            return bmp.GetPixel(0, 0);
        }
```

The code above will return a Color variable (e.g. declare as: "Color mycolour"), and you will access its components as mycolour.R, mycolour.G, and mycolour.B

Well, the methods above will enable you to do the basic things: Focus WOW windows, read pixel colors, move and click mouse, and send key press to the game.

Reply With Quote

12-03-2019 #3

Humbleguy Active Member

PART 3 - SPECIALLY CHALLENGING STUFF

Once you can properly read your position coordinates in the game, and send keystrokes and mouse clicks to the game, next thing you must do is a navigation system. This is somehow challenging, but doable. It consists in:

1. Read your coordinates from the game (from the addon/weak aura)
2. Read your actual facing direction (are you looking north, south, east?)
3. Turn your char to the direction you must move.
4. Check the distance between your current position and desired position.
5. Press W until this distance reaches "zero"

Unfortunetly, all you got is your currently and desired X and Y coordinates, and there is no other way to convert that in real clicks and keypress, but using a lot of trigonometry and math. If you were not a good math student, and hated trigonometry. this may be not for you.

UNDERSTANDING RADIANS

First, you must understand how game deals with angles. You probably know that a full turn is 360 degrees. But wow does not use degrees. Instead, angles are presented in RADIANS. One radian = aprox. 57 degree. To be more precise, 360 degrees = 2 pi radians. Got it? "The circumference

When you press game keys A or D, you will take exactly 2 seconds to make a full turn. That translates into one pi radian per second, or 180 degrees per second. So, if you want to make a 90 degree turn to the right, you must press the D key for 500 miliseconds, or the A key for (2-500) miliseconds.

DISCOVERING WHERE TO PLAYER IS FACING

This can be done easily with a Lua function (included in the Weak Auras at the first post), but the function is:

Code:

```
fa = GetPlayerFacing()/6.2832
```

The code above will return your currently facing postition. I have divided it by 2 pi, so that you get a percentage result. Remember that when you transform it in a colour element, result will be from 0-255 range.

DISCOVER THE ANGLE NECESSARY TO TURN FROM CURRENT(X,Y) to TARGET(X,Y)

This is tricky. The only way to do this is using the trigonometric function atan2. This is not much friendly. The most important thing you need to know is that this function uses Y before X at the arguments. I am very sorry for you, but there is no easy way to do that without using trigonometry functions. It will be something like this:

Code:

```
        double getangle(double y1, double y2, double x1, double x2)
            {
            double ang =    Math.Atan2(x1 - x2, y1 - y2)/Math.PI;
            if (ang < 0) ang += 2; // this is used to avoind negati
            return Math.Round(ang*1000);
            }
```

METHOD TO CHECK DISTANCE BETWEEN TWO COORDINATES (TARGET -> CURRENT)

For the below code, I used a simple struct to store locations, to make things easier to manage:

Code:

```
       // Estrutura de coordenada
        struct loc
        {
            public int x;
            public int y;
        }
        struct loc
        {
            public int x;
```

Code:

```
int dist(loc orig, loc tar)
        {
            double distance = (Math.Sqrt(Math.Pow(Math.Abs(orig.x -
                Math.Pow(Math.Abs(orig.y - tar.y), 2))); // formula
            int temp = (int)distance; // converte double para int
            // tbdist.Text = temp.ToString(); // escreve distancia
            return temp; // retorna distancia entre os pontos
        }
```

GET YAW (TURNING ANGLE) NEEDED

Once you got actual player facing position, and you know the needed new face, you need to discover the yaw (angle diference), that means, the necessary turn. You must turn to the lowest side (e.g turn 10 degrees left, not 350 degrees right).

When measuring angles, and angle needed to turn, you can use degrees (0-360), radians (0-2 pi), or a 0-2000 scale.
Keep in mind that when you press A or D, your char will take 2 seconds to make a full turn, so we know that:

- It turns 90 degrees per second, or 1 radian per second, or 1000 miliseconds per second.
C# uses natively radians, but I recommend for you to use a 0-2000 scale, so that you just use the number of miliseconds needed to turn. For example, if you have a method called "turn" that have as arguments the angle.... you would have, to turn a full 360 degree turn:

turn(2*math.pi) = rads
turn(360) = degs
turn(2000) = using a milisecond scale. <--- EASIER METHOD.

I used the functions below as example, but they have my own variables so will not be usable instantly:

First you read the angle from the addon:

Code:

```
Color cor = GetColorAt(x,y);   // X and Y are screen location of ad
Double facing = Math.Round(cor.B * 7.81);   // Convert from 0-255 t
```

Then you calculate angle needed to your new coordinates

Code:

```
        //Calcula o ângulo até a coordenada alvo; norte = 0; sul =
        double getangle(double y1, double y2, double x1, double x2)
            {
            double ang =    Math.Atan2(x1 - x2, y1 - y2)/Math.PI;
            if (ang < 0) ang += 2;
```

Code:

```
        void getyaw(loc orig, loc tar)
        {
            // Retirar as linhas abaixo

            getpos();
            orig = player.pos; // this is the xy struct, shown above
            tar = t;

             double yaw_result;
            pitch = getangle(orig.y, tar.y, orig.x, tar.x); // remem
            // Calcula angulo necessacio (yaw)
            if (pitch > facing) yaw_result = pitch - facing;
            else if (pitch < facing) yaw_result = 2000 -facing + pit
            else yaw_result = 0;
            yaw =  yaw_result;
        }
```

I know that is not easy, but theres no easier way to make a navigation method.

Once you know this all, you just make a while loop to move, keep checking angles and turning while you walk, until distance < 10 or so. You must do this way to avoid robot-like movement (stop-turn-walk....) . Code below is just exemple, as it has my own variables and wont be usable "as is", but can be used to learn algorithim structure for walking from point a to point b, so I will comment it heavliy

Code:

```
do // start moving loop
            {
                getstats(ref player); // Check player stats (I need
                if (player.combat==true) // oh no... entered combat
                {
                    combatloop();   // start combat routine until mo
                    aperta(WKEY, 0); // start walking again after co
                }

                temp++;
                focawow(); // focus wow process
                getpos(); // check my position
                getyaw(player.pos, t); // check yaw (turning) neede
                facepoint(); // turn to my target.
                wait(500); // walk a bit....
                if (temp % 7 == 0) aperta(SPACEBAR, 100); // jump so
                if (temp % 2 == 0) // every 2 seconds will check if
                {
                    if (player.pos.x == oldloc.x && player.pos.y ==
                    // no...
                    {
                        unstuck(); // start unstuck routine - code i
                    }
```

It is necessary. From my experience, jumping every 3 seconds and the unstuck routine below will make you move without problems thru the game. (walks around trees and things like that)
You must include this routine at your walk loop.

Code:

```
        void unstuck()
        {
            focawow(); // focus wow window, method stated above
            if (!enroscou) aperta(SPACEBAR,100); // first time it i
            if (enroscou) // (player.spd < 100) // second stuck, wi
            {
                aperta(WKEY); // Start walking
                aperta(EKEY, 700); // turn southeast  (should be D k
                aperta(WKEY, 1000); // walk for one second

            }
            aperta(WKEY, 0); // start walking again , 0 time means i
            enroscou = !enroscou;  // just to alternate jump-circle

        }
```

Well guys, that was what I had. If anyone has an specific question or need help at implementing stuff about, feel free to ask.
Currently I am working on multi-threading to make interface responsive. Unfortunatly unsuscessful so far. I am also having issues to make bot face my target too.

Thank you!!

/// EXTRA INFO- NEW USEFUL / ADVANCED METHODS - ADDED FEBRUARY 2020

After having the grind bot fully working (best class so far to me was a paladin), I have some feedback, and extra methods / algorithims to share.
Best class overall for make a bot for me was a paladin. They have high survivability, can heal, and look less botish while playing (as a paladin itself does not move much). Combat routine is easier too, as all you got to do is keep Seal of Righteousness up and judge on cooldown.

I came with some issues, that i would like to share.
First: sometimes when waiting for things like eating, drinking, looting or skinning, mobs attacked us, and if we use a simple "wait" command do do these things, if you get attacked while eating you would wait a lot without reacting. So you better do an "active" wait method, that will check for combat status every 1 second, and you use this to wait while eating, looting, or skinning, so that if you enter combat, bot will stop waiting and react properly. The "wait" function, when used to drink or eat, should also check for "full life" status, so that you stop drinking if your mana gets to 100%. that will save you a lot of time. Check the pseudo-code below:
----

lose target from mob after you have looted, just check for losing target while you wait for loot complete.

Code:

```
        //----------------------------------
        // WAIT WHILE LOOTING
        //---------------------------------
        bool waitloot(int seconds)
        {
            check-my-status();
            bool loot_sucessful = true;
            for (int i = 0; i < seconds; i++)
            {
                wait(1000); // wait 1 second
                check-my-status();
                if (me.combat) { wait_sucessful = false; break; }
                if (me.tarhp==0) { return loot_sucessful ; } // wow

            }
            return loot_sucessful ;
```

Waiting to eat / drink may vary in time , as you may recover life faster, or even someone may heal you. So you should check your hp every 1 second while eating / drinking, or even casting long heals like 2,5s heals. Also should check for combat state while waiting. Do it eveyr 1 second.

Code:

```
        //----------------------------------------------------------
        // WAIT WHILE EATING / DRINKING / LONG HEALS
        //----------------------------------------------------------
        bool espera(int segundos,int hp=0) // Arguments: seconds, ar
        {
            checkme(); // check status
            bool deucerto = true; // waited fine return
                for (int i=0;i<segundos;i++) // wait 1 second, loop
                {
                    checkme();
                if (me.combat) { deucerto = false; break; } // enter
                if (hp == 101 && me.mana > 90) break; // stop drinki
                if (hp!= 101 && hp>0 && me.hp >= hp) { break; } // s
                    wait(1000); // wait 1 second
                }
            return deucerto; // returns successful.
        }
```

Reply With Quote

12-04-2019 #4

InnerSilence Active Member

i.e. unless you intent to wait for less than 40 ms, your wait function is such a waste of cpu cycles. You could use 'await Task.Delay(millisecondsToWait) ' in an async method instead.

Reply With Quote

12-04-2019 #5

Humbleguy Active Member

Thank you for your feedback; I didnt know what exactly my post would be... it ended up being a bunch of routines and code collection; but I cannot edit the title.... I´m not much into Github stuff; Im not elite coder. Still trying to find out how to make async methods, so that i can keep ui responsive. I wish I could click a button in the windows forms UI, that would set a global bool variable to "true" to stop the bot, like this:

-START- button at winforms interface

Code:

```
while (bool stopbot  == false)
{
   ...walk thru waypoints routine...
   while (bool incombat==true && stopbot == false)
    {
    combat loop routine;
    }

}
```

-STOP- button at winforms interface

Code:

```
stopbot = true;
```

But this simply does not work... because when at combat loops interface freezes, and I cannot press stop. Then the only way to make bot stop is close bot. ☹

Reply With Quote

12-04-2019 #6

InnerSilence Active Member

stopbot should be a class variable you can not define it in your while like that. Anyway try to learn some basic github and upload your code I will comment on your code to show you how to do it.

Humbleguy (1 members gave Thanks to InnerSilence for this useful post)

12-04-2019 #7

Humbleguy Active Member

This is what I got using the above code, so far (bot demonstration) below. I still have not solved "how to face target" issue. I have circunvented the issue using warlock class, and using voidwalker+health funnel / drain soul to "focus" my target, but that limits available classes to warlock / priest (mind flay).

Youtube link to the pixel bot demonstration:

Combat loop is like: Pull with curse of agony -> corruption -> health funnel -> immolate -> drain soul -> shadow bolt until 35%, then drain sould to get soulstone.

Code:

```
try pull stuff, clicking "try pull" macro: /cleartarget /targetenemy
if (me.combat) combatloop(); // pulled something or body-aggroed

void combatloop()
{
while (mob.health > 0)
{
If (mob.health >= 66) cast corruption // does not require facing tar
if (me.health < 40) use healthstone // if low life, use healthstone
if (me.health < 40 && no healthstone available) cast drain life // r
if (me.mana < 40 cast life tap)
}
}
```

before pull, check for low mana (lifetap), buff (skin armor), healthstone available, pet alive., or low health (eat food)

If we have no soul shard control, they will stack in your bag too much.... so I made a macro that will delete any exceeding soul shard whenever I cast immolate:

Code:

```
/run i="Soul Shard"d=GetItemCount(i)-6 for x=0,4 do for y=1,GetConta
/assist pet
/cast immolate
```

Best way to pull stuff while grinding is just "click-frenzy" a pull macro while walking, and check for combat afterwards (pulled stuff):

Code:

```
/cleartarget
/targetenemy
/cast curse of agony
```

The macro above will pull any mob in front of the bot while it walks, if you keep clicking the macro every 1-2 seconds. Then if your combat status gets true, you call the combat loop method, that will run while you still in combat. This way you dont need to check range or anything if you just want to grind/skin/level up.

To loot/skin, you just click twice the area in front of your char and hopefully you will loot / skin stuff. It works 70% of the time. Turning on "camera follow" as "always", will make camera rotate when you target mobs so that they stay in the right position.

Reply With Quote

12-05-2019 #8

InnerSilence Active Member

I dont know much about what kind of info you can get from Lua ingame. But, it depends if you can get enemy coordinates. If that's possible, facing enemy can be easily done. I assume you can not do that. If so, one way I can think of to achieve this before getting in combat is:
1- Mark your target with some distinct color marker.
2- Searching for mark color in your screen.

2- You need a fast algorithm to find enemy marker (lets say less than 10 ms)

Reply With Quote

Humbleguy (1 members gave Thanks to InnerSilence for this useful post)

12-05-2019 #9

Kwapuzzi Member

Mentioned this idea in my lua thread. You can check if your target has raidtarget 1,2,3,4,5 etc. if it has the raidtarget, but its not visible - turn 180deg.
there is no other option ingame to check target coords. i heard its hard in c# to write a fast biteshift method for fast rgb read.

in java it was way easier. just extending the robot class and you are ready 🙂

Reply With Quote

12-05-2019 #10

Kwapuzzi Member

> To loot/skin, you just click twice the area in front of your char and hopefully you will loot / skin stuff. It works 70% of the time. Turning on "camera follow" as "always", will make camera rotate when you target mobs so that they stay in the right position.

Better- check for tooltiplinecount = 3. if its skinnable, its 3.
you can determine if you have looted the mob with the event LOOT_OPENED and LOOT_CLOSED. so you are not wasting time and could imedietly resume your loops when looted.

If you have any questions to solve with lua, feel free to ask.

Reply With Quote

12-05-2019 #11

Humbleguy Active Member

> Originally Posted by Kwapuzzi

events), like "Target needs to be in front of you." or "you are facing the wrong way"?

If I could check that messages being displayed, I could easily just turn around. But I tried to use some lua code to check for that error messages without success. I think that checking for the emotes would work as well.

I tried the weak aura below, but it simply did not work - and I dont know why - it should check for UI_ERROR_MESSAGE event:

Code:

```
function(event, message)
    if(message == ("You are facing the wrong way!"))then
        return true
    end
end
```

Here is the import string:

Code:

```
!ns1xVnUnm8pl7EQfyOOnTbB7H7b3g3M0LMey5SddOqok202cNJKNKCsZEiF2hPu(
```

Reply With Quote

12-06-2019 #12

InnerSilence Active Member

> Originally Posted by Kwapuzzi
>
> Mentioned this idea in my lua thread. You can check if your target has raidtarget 1,2,3,4,5 etc. if it has the raidtarget, but its not visible - turn 180deg.
> there is no other option ingame to check target coords. i heard its hard in c# to write a fast biteshift method for fast rgb read.
>
> in java it was way easier. just extending the robot class and you are ready 😉

LoL, I see, looking at that thread it seems finding where target is and moving to it is a challenge in pixel based bots 😉.
In c#, one can use unsafe code with pointers to do such a things pretty fast.
I remember, back then when D3 released, I had used the 3D to 2D projection to calculate where on the screen I should click or vice versa. In D3 camera was fixed so calculations were easier.
It gonna be hard for wow given the freedom of the camera. But I do believe that if one could get the current camera projection, then it is possible to convert object locations to and from

Reply With Quote

12-06-2019 #13

Kwapuzzi Member

> Originally Posted by Humbleguy
>
> I do have a doubt, if you can help me. How do I chack for UI interface errors (combat error events), like "Target needs to be in front of you." or "you are facing the wrong way"?
>
> If I could check that messages being displayed, I could easily just turn around. But I tried to use some lua code to check for that error messages without success. I think that checking for the emotes would work as well.
>
> I tried the weak aura below, but it simply did not work - and I dont know why - it should check for UI_ERROR_MESSAGE event:
>
> Code:
>
> ```
> function(event, message)
>     if(message == ("You are facing the wrong way!"))then
>         return true
>     end
> end
> ```
>
> Here is the import string:
>
> Code:
>
> ```
> !ns1xVnUnm8pl7EQfyOOnTbB7H7b3g3M0LMey5SddOqok202cNJKNKCsZEiF2hPu(hwUcCib20uKuK)4ps
> ```

So, for what i know you can not check for fight event logs while fighting. I couldnt find a way either.
Currently im managing the in front of me with a +-20 deg left or right. you can just check if a certain spell is in range. for a caster class its easy. if you are in range, cast, if not - hit interact with target twice in combiantion with click 2 move and your char is moving straight towars your target until its in range to cast. for a melee you need a bit more fine tuning.

check for camera movement is set to always follow and fast.

Update: I could barely find infos for classic, but UI_ERROR_MESSAGE is not handled as in BFA currently.
You can not filter for specific messages. errorType and message are nil or always the same. you can not match like https://www.townlong-yak.com/framexm...Strings.lua/DE one of these.

Currently its just possibel to catch ANY ui message, and this could be anything. maybe its useful anyways.

stonebent Member

> Originally Posted by Humbleguy
>
> Thank you for your feedback; I didnt know what exactly my post would be... it ended up being a bunch of routines and code collection; but I cannot edit the title.... I´m not much into Github stuff; Im not elite coder. Still trying to find out how to make async methods, so that i can keep ui responsive. I wish I could click a button in the windows forms UI, that would set a global bool variable to "true" to stop the bot, like this:
>
> -START- button at winforms interface
>
> Code:
>
> ```
> while (bool stopbot  == false)
> {
>    ...walk thru waypoints routine...
>    while (bool incombat==true && stopbot == false)
>     {
>     combat loop routine;
>     }
>
> }
> ```
>
> -STOP- button at winforms interface
>
> Code:
>
> ```
> stopbot = true;
> ```
>
> But this simply does not work... because when at combat loops interface freezes, and I cannot press stop. Then the only way to make bot stop is close bot. ☹

Here's something quick i just wrote to help you out. You need to multithread or use tasks.

Code:

```
using System;
using System.Threading;

public class Program
{
        public static bool botIsRunning;

        private void button1_Click(object sender, EventArgs e)
        {
                startNewThread();
        }

        public static void mainBotMethod()
        {
                while(botIsRunning)
                {
                        // All your stuff in here
                        // Check it from time to time just to make
                        if(!botIsRunning)
                        {
                                break;
```

Reply With Quote

Humbleguy (1 members gave Thanks to stonebent for this useful post)

12-06-2019 #15

Humbleguy Active Member

> Originally Posted by Kwapuzzi
>
> So, for what i know you can not check for fight event logs while fighting. I couldnt find a way either.
>
> Update: I could barely find infos for classic, but UI_ERROR_MESSAGE is not handled as in BFA currently.
> You can not filter for specific messages. errorType and message are nil or always the same. you can not match like https://www.townlong-yak.com/framexm...Strings.lua/DE one of these.
>
> Currently its just possibel to catch ANY ui message, and this could be anything. maybe its useful anyways.

I was able to catch UI errors messages, selectively, using an addon called TELL ME WHEN. Its a bit clunky, but it does the job. Being able to write the code ourselves (e.g. Weak Aura) would make stuff easier do deal. Check the video below of the addon:

And here (click below) you can check a picture of the addon interface which created the video above. It generates a red square on the screen when I get the UI error message

Reply With Quote

1 › Last ›

« Previous Thread | Next Thread »

OWNEDCORE FORUMS

News
World of Warcraft
Pokemon GO
Diablo 3
Buy Sell Trade
MMO
Overwatch
RTS
General

OWNEDCORE CORECOINS

CoreCoins
CoreCoins FAQ
Shout-Out
Banner Ads

MY OWNEDCORE

My Profile
Notifications
Settings
Buy CoreCoins

OwnedCore- World of Warcraft Exploits, Hacks, Bots and Guides

Privacy Policy | Cookie Policy | Terms | Contact Us

Available Payment Methods:

PayPal Skrill PayOp

All times are GMT -5. The time now is 10:42 AM. Powered by vBulletin® Version 4.2.3

Digital Point modules: Sphinx-based search