CoSpace Rescue Tutorial 1: Getting Started

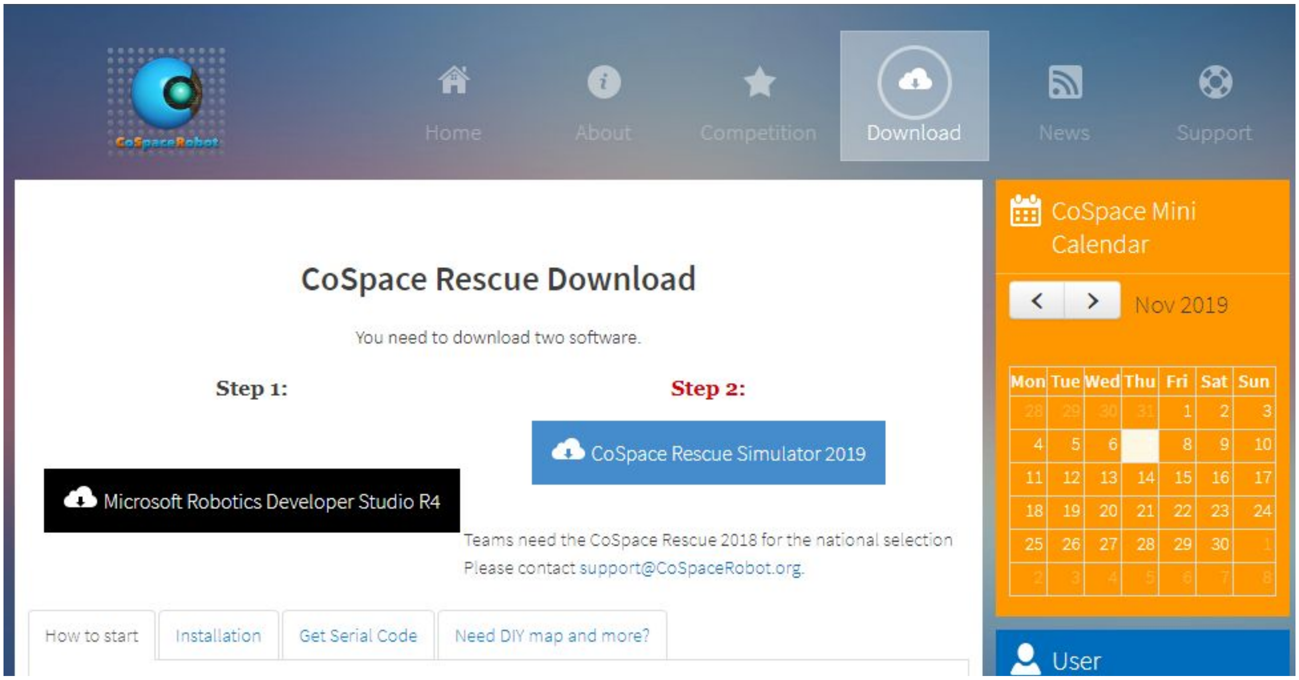
CoSpace Rescue is a virtual robot competition, where virtual robots you code in C/C++ compete to get the most points in a virtual simulation.
https://www.cospacerobot.org/

There are competitions about this software. The main competition using this software is Robocup Junior Rescue Simulation. For now you can check more about the rules here:
https://www.cospacerobot.org/competition/cospace-rescue

In this tutorial, you are going to learn how to install & compile code to run on the virtual robot in Cospace Rescue Simulator.
This tutorial series assumes knowledge of programming, and more specifically C++ programming. If you know how to program in other languages, chances are just learning the C++ syntax. However, if you don't know how to program, you should probably program in other languages first, like Python.

Download & Install

You can download the files from here:
https://www.cospacerobot.org/download/cospace-rescue-download



CoSpace Rescue Download

You need to download two software.

Step 1:

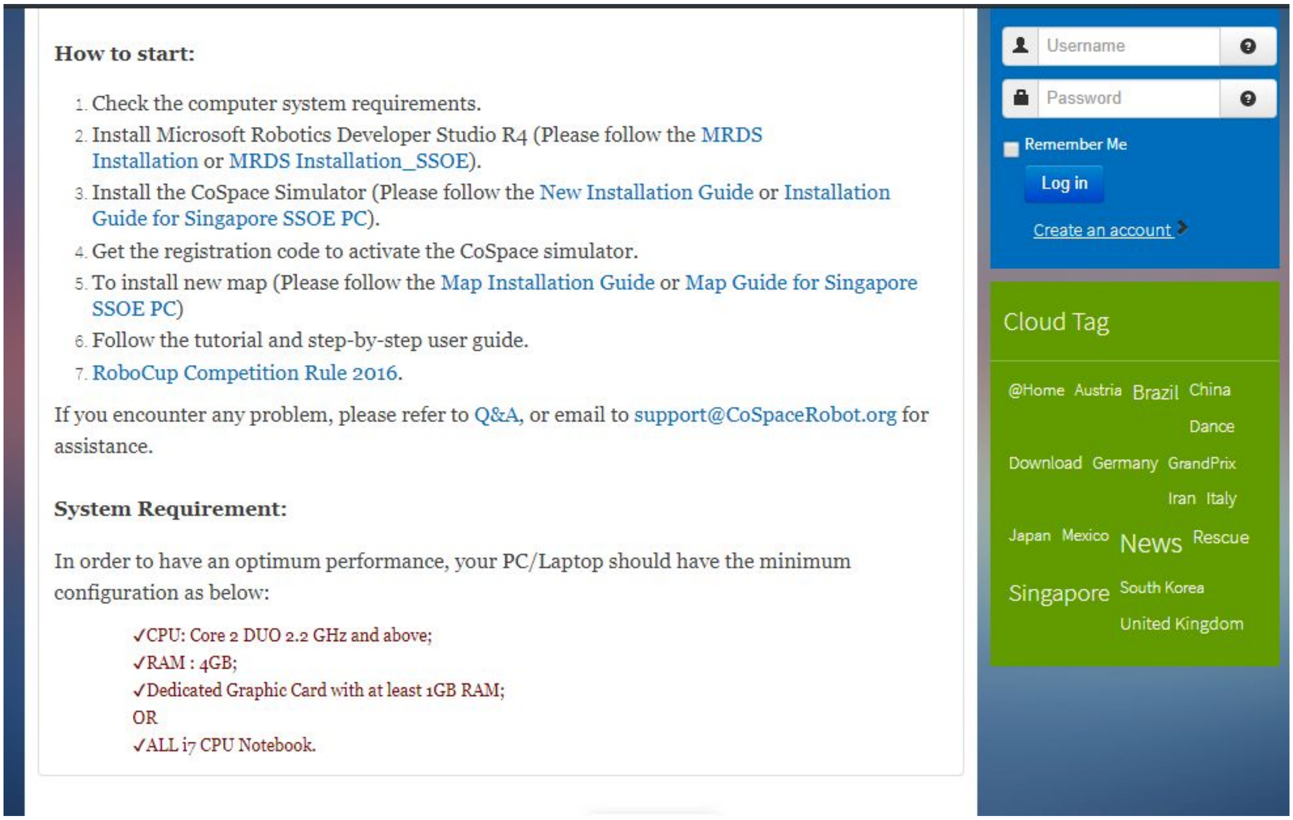
Step 2:

Teams need the CoSpace Rescue 2018 for the national selection
Please contact support@CoSpaceRobot.org.

CoSpace Mini Calendar

Nov 2019

Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	31	1	2	3
4	5	6		8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8


Username
Password
Remember Me
Log in
Create an account
Cloud Tag
@Home Austria Brazil China Dance Download Germany GrandPrix Iran Italy Japan Mexico News Rescue Singapore South Korea United Kingdom

How to start:

1. Check the computer system requirements.
2. Install Microsoft Robotics Developer Studio R4 (Please follow the MRDS Installation or MRDS Installation_SSOE).
3. Install the CoSpace Simulator (Please follow the New Installation Guide or Installation Guide for Singapore SSOE PC).
4. Get the registration code to activate the CoSpace simulator.
5. To install new map (Please follow the Map Installation Guide or Map Guide for Singapore SSOE PC)
6. Follow the tutorial and step-by-step user guide.
7. RoboCup Competition Rule 2016.

If you encounter any problem, please refer to Q&A, or email to support@CoSpaceRobot.org for assistance.

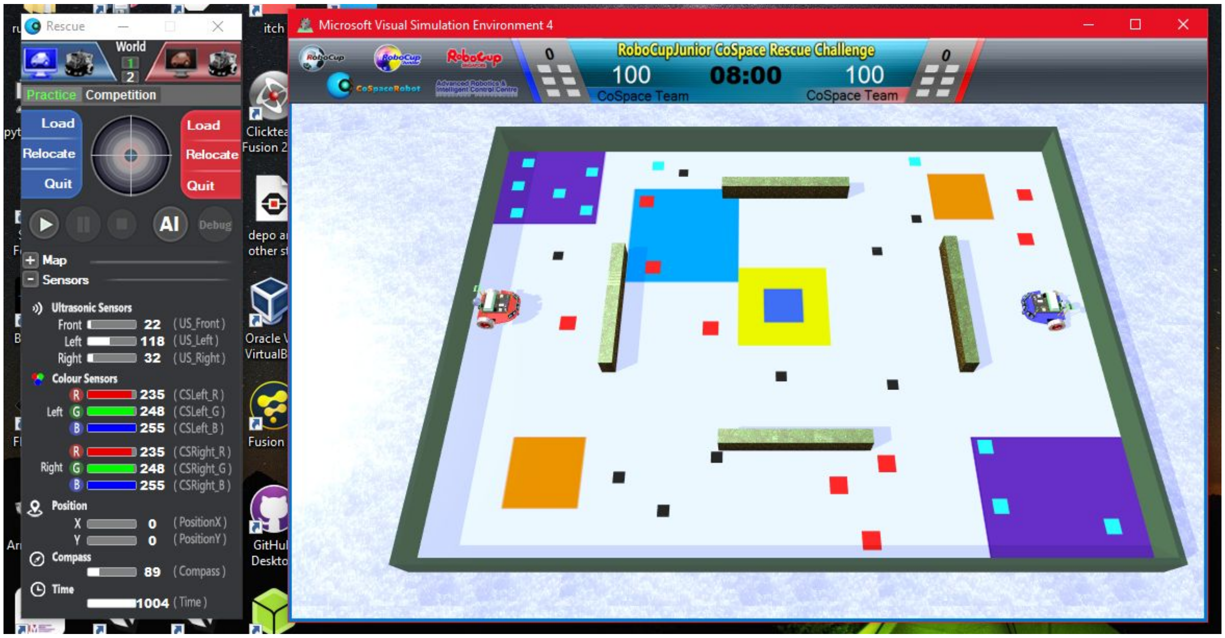
System Requirement:

In order to have an optimum performance, your PC/Laptop should have the minimum configuration as below:

✓CPU: Core 2 DUO 2.2 GHz and above;
✓RAM : 4GB;
✓Dedicated Graphic Card with at least 1GB RAM;
OR
✓ALL i7 CPU Notebook.

You may need to get a serial code to use all of Cospace Rescue's features, however, even without a serial code, you should still be able to use the software to compile & run code. It's just that you would not have manual relocation of the robot along with some other features.

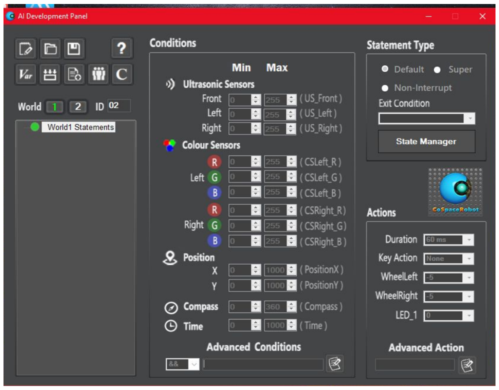
When you are done you should be able to get a screen like this. The difference in the map does not matter.



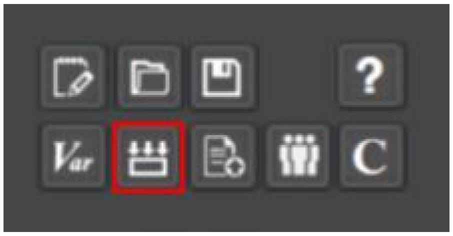
Ultrasonic Sensors
Front 22 (US_Front)
Left 118 (US_Left)
Right 32 (US_Right)
Colour Sensors
Left R 235 (CSLeft_R)
G 248 (CSLeft_G)
B 255 (CSLeft_B)
Right R 235 (CSRight_R)
G 248 (CSRight_G)
B 255 (CSRight_B)
Position
X 0 (PositionX)
Y 0 (PositionY)
Compass 89 (Compass)
Time 1004 (Time)

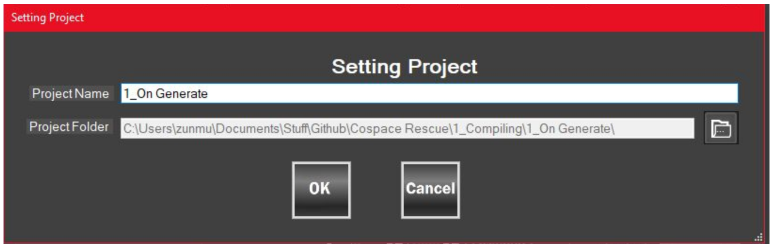
Using C

Let’s start by compiling for C, shall we? Click the AI button. You should get this window



To get code for C click the build project icon in the menu. Create your project, and in that new folder, ai.c should be generated. Only ai.c is needed

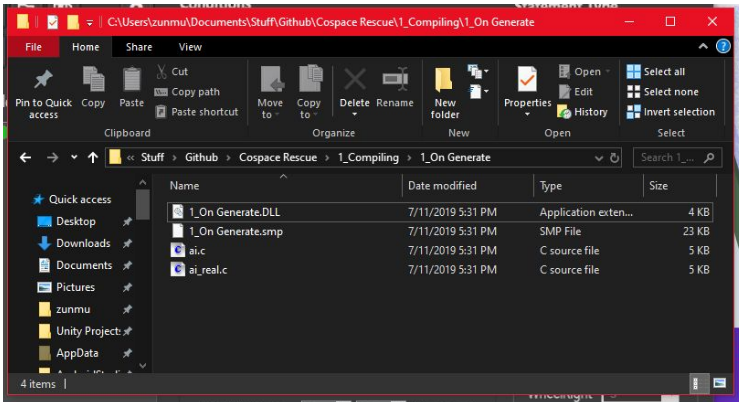



Setting Project

Setting Project

Project Name: 1_On Generate

Project Folder: C:\Users\zunmu\Documents\Stuff\Github\Cospace Rescue\1_Compiling\1_On Generate\

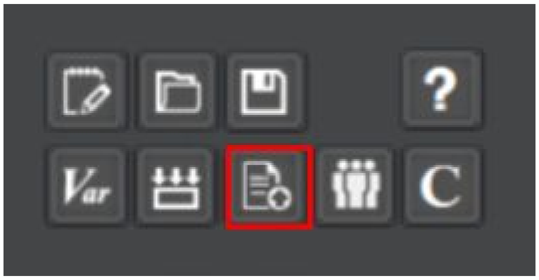


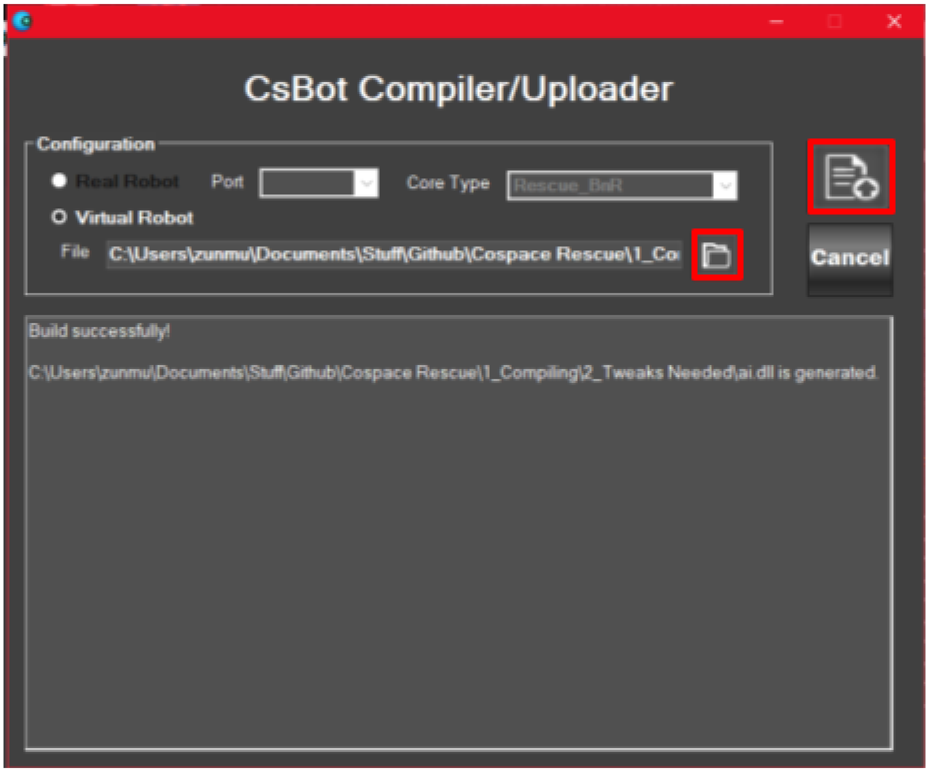
C:\Users\zunmu\Documents\Stuff\Github\Cospace Rescue\1_Compiling\1_On Generate

Name	Date modified	Type	Size
1_On Generate.DLL	7/11/2019 5:31 PM	Application exten...	4 KB
1_On Generate.smp	7/11/2019 5:31 PM	SMP File	23 KB
ai.c	7/11/2019 5:31 PM	C source file	5 KB
ai_real.c	7/11/2019 5:31 PM	C source file	5 KB

4 items

To compile the code click the Upload to Robot icon in the main menu. Select your file and then compile





CsBot Compiler/Uploader

Configuration

● Real Robot   Port   Core Type Rescue_BnR

○ Virtual Robot

File C:\Users\zunmu\Documents\Stuff\Github\Cospace Rescue\1_Co

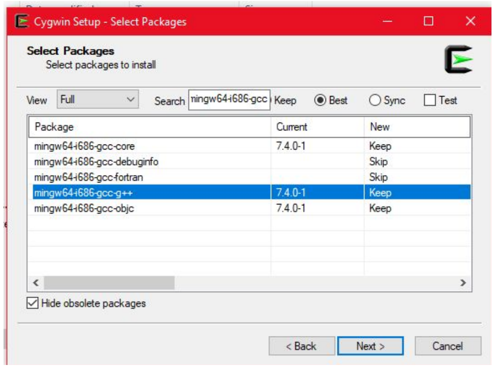
```
Build successfully!
C:\Users\zunmu\Documents\Stuff\Github\Cospace Rescue\1_Compiling\2_Tweaks Needed\ai.dll is generated.
```

And you successfully compiled for C. The compiled file is ai.dll.

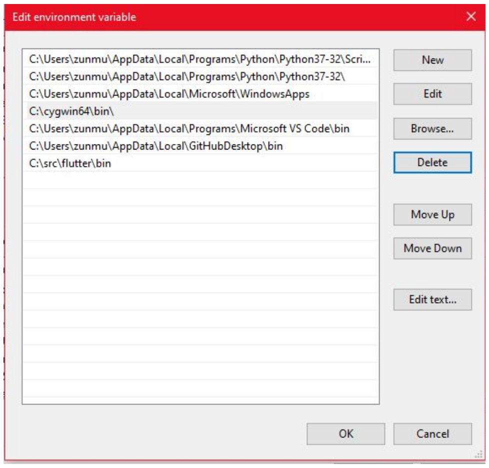
Using C++

However, there are many stuff (like data structures, Object Oriented Programming) that you can get with only C++. Hence, most of the national players use C++.

To compile for C++, you have to install Cygwin and the C++ compiler on your computer, which you can find here https://cygwin.com/setup-x86_64.exe. During the Cygwin install, take note to install the mingw-g++ C++ compiler. Also take note to add to Windows' PATH Variable, like here.

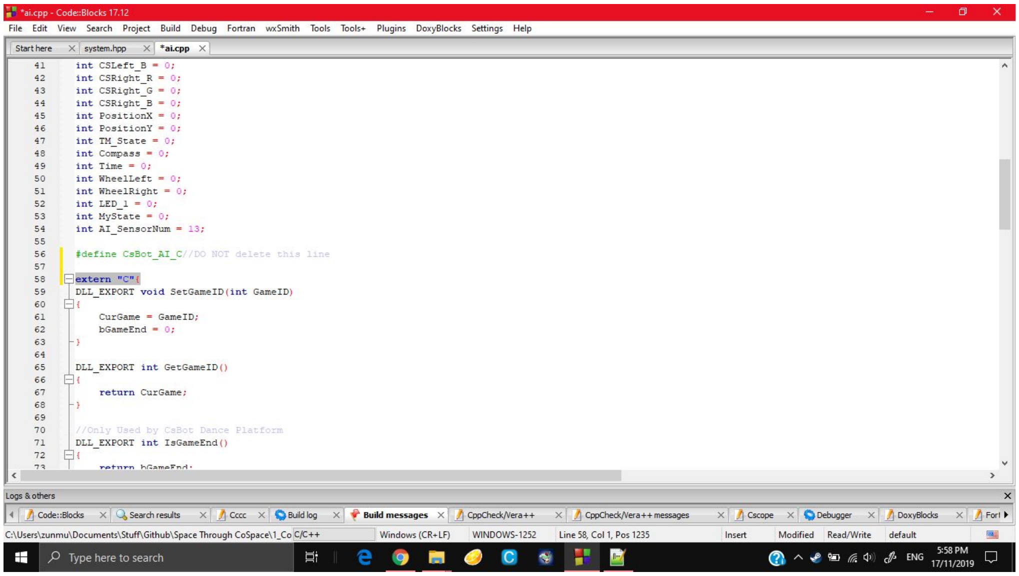


Package	Current	New
mingw64-i686-gcc-core	7.4.0-1	Keep
mingw64-i686-gcc-debuginfo		Skip
mingw64-i686-gcc-fortran		Skip
mingw64-i686-gcc-g++	7.4.0-1	Keep
mingw64-i686-gcc-objc	7.4.0-1	Keep

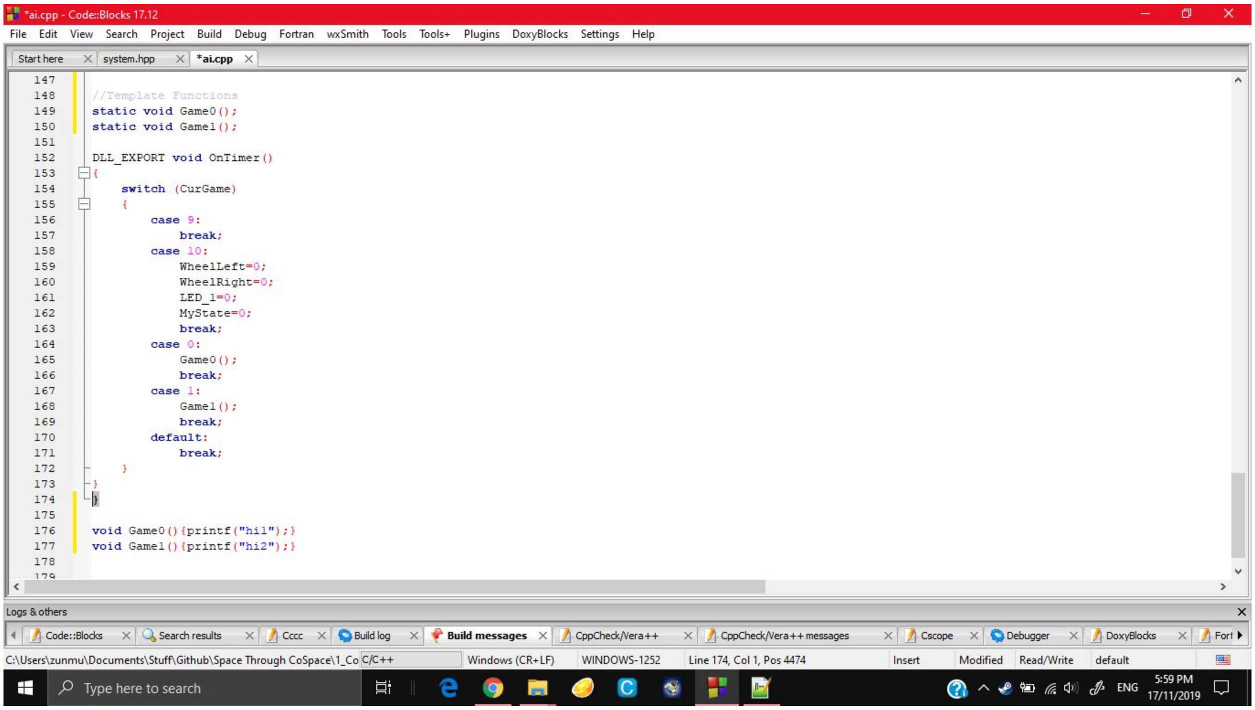


```
C:\Users\zunmu\AppData\Local\Programs\Python\Python37-32\Scri...
C:\Users\zunmu\AppData\Local\Programs\Python\Python37-32\
C:\Users\zunmu\AppData\Local\Microsoft\WindowsApps
C:\cygwin64\bin\
C:\Users\zunmu\AppData\Local\Programs\Microsoft VS Code\bin
C:\Users\zunmu\AppData\Local\GitHubDesktop\bin
C:\src\flutter\bin
```

Once you have installed the programs, you can just copy the code into a new file, ai.cpp. However, you have to make some modifications to the code first in ai.cpp. You have to put extern "C"{...} over the DLL_EXPORT Functions, so that they will compile as C code, to be used in CoSpace Simulator



```
int CSLeft_B = 0;
int CSRight_R = 0;
int CSRight_G = 0;
int CSRight_B = 0;
int PositionX = 0;
int PositionY = 0;
int TM_State = 0;
int Compass = 0;
int Time = 0;
int WheelLeft = 0;
int WheelRight = 0;
int LED_1 = 0;
int MyState = 0;
int AI_SensorNum = 13;

#define CsBot_AI_C//DO NOT delete this line

extern "C"{
DLL_EXPORT void SetGameID(int GameID)
{
    CurGame = GameID;
    bGameEnd = 0;
}

DLL_EXPORT int GetGameID()
{
    return CurGame;
}

//Only Used by CsBot Dance Platform
DLL_EXPORT int IsGameEnd()
{
    return bGameEnd;
```

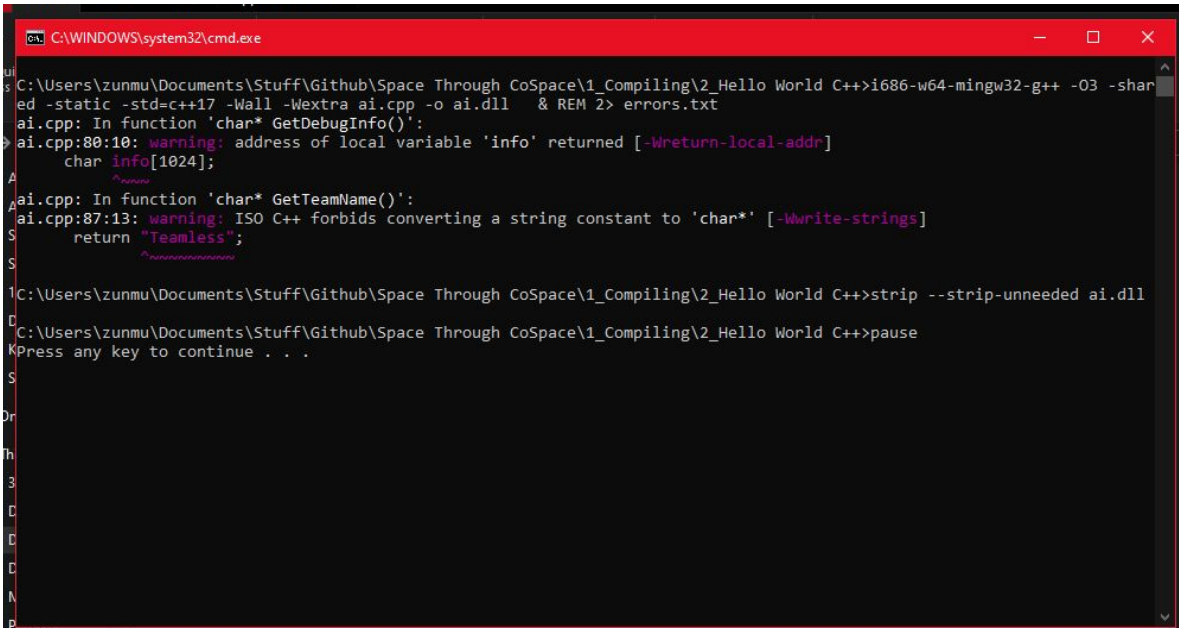


```

//Template Functions
static void Game0();
static void Game1();

DLL_EXPORT void OnTimer()
{
    switch (CurGame)
    {
        case 9:
            break;
        case 10:
            WheelLeft=0;
            WheelRight=0;
            LED_1=0;
            MyState=0;
            break;
        case 0:
            Game0();
            break;
        case 1:
            Game1();
            break;
        default:
            break;
    }
}
}

void Game0(){printf("hi1");}
void Game1(){printf("hi2");}

```

The main commands to compile the code in the directory are

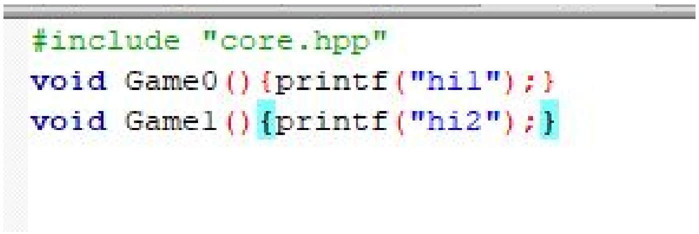
```
i686-w64-mingw32-g++ -O3 -shared -static -std=c++17 -Wall -Wextra
ai.cpp -o ai.dll & REM 2> errors.txt
strip --strip-unneeded ai.dll
pause
```

You can change the ai.cpp and ai.dll files to the appropriate filenames if you want. You can put this command into a .bat file and run it.



```
C:\Users\zunmu\Documents\Stuff\Github\Space Through CoSpace\1_Compiling\2_Hello World C++>i686-w64-mingw32-g++ -O3 -shar
ed -static -std=c++17 -Wall -Wextra ai.cpp -o ai.dll   & REM 2> errors.txt
ai.cpp: In function 'char* GetDebugInfo()':
ai.cpp:80:10: warning: address of local variable 'info' returned [-Wreturn-local-addr]
     char info[1024];
          ^~~~
ai.cpp: In function 'char* GetTeamName()':
ai.cpp:87:13: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
      return "Teamless";
             ^~~~~~~~~~

C:\Users\zunmu\Documents\Stuff\Github\Space Through CoSpace\1_Compiling\2_Hello World C++>strip --strip-unneeded ai.dll

C:\Users\zunmu\Documents\Stuff\Github\Space Through CoSpace\1_Compiling\2_Hello World C++>pause
Press any key to continue . . .
```

Nice it works! You can actually move most of the above code into a seperate header file (as you would not need them anymore), leaving only the main Game0() and Game1() functions in ai.cpp, like this.



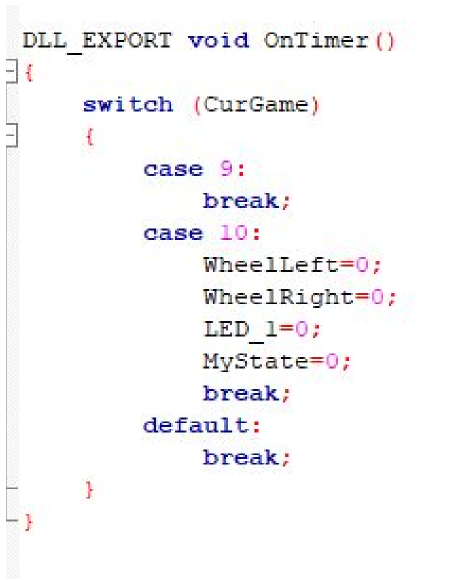
```
#include "core.hpp"
void Game0(){printf("hi1");}
void Game1(){printf("hi2");}
```

So much easier to read now! We will be using this code as a template for the next tutorials.

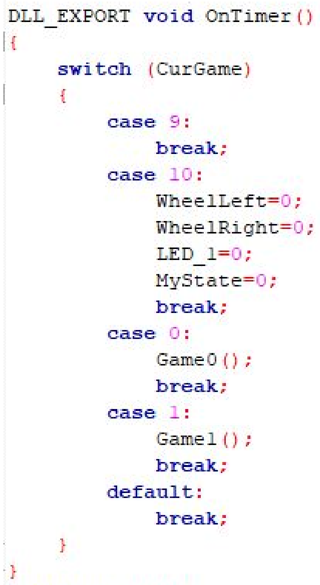
Running some code!

Let’s start with a simple hello world program.
Firstly, we need to make some tweaks to the ai.c for it to run our code.

Right now we should change the bottom of the code like so. This is so that your code will run in the functions Game0() and Game1().

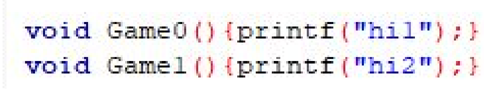


```
DLL_EXPORT void OnTimer()
{
    switch (CurGame)
    {
        case 9:
            break;
        case 10:
            WheelLeft=0;
            WheelRight=0;
            LED_1=0;
            MyState=0;
            break;
        default:
            break;
    }
}
```



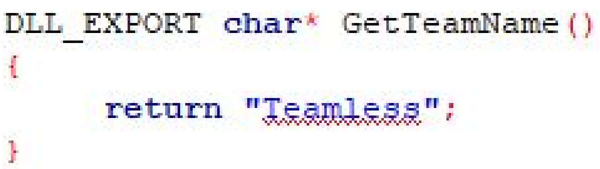
```
DLL_EXPORT void OnTimer()
{
    switch (CurGame)
    {
        case 9:
            break;
        case 10:
            WheelLeft=0;
            WheelRight=0;
            LED_1=0;
            MyState=0;
            break;
        case 0:
            Game0();
            break;
        case 1:
            Game1();
            break;
        default:
            break;
    }
}
```

This also means that you have to add this code before the DLL_EXPORT.



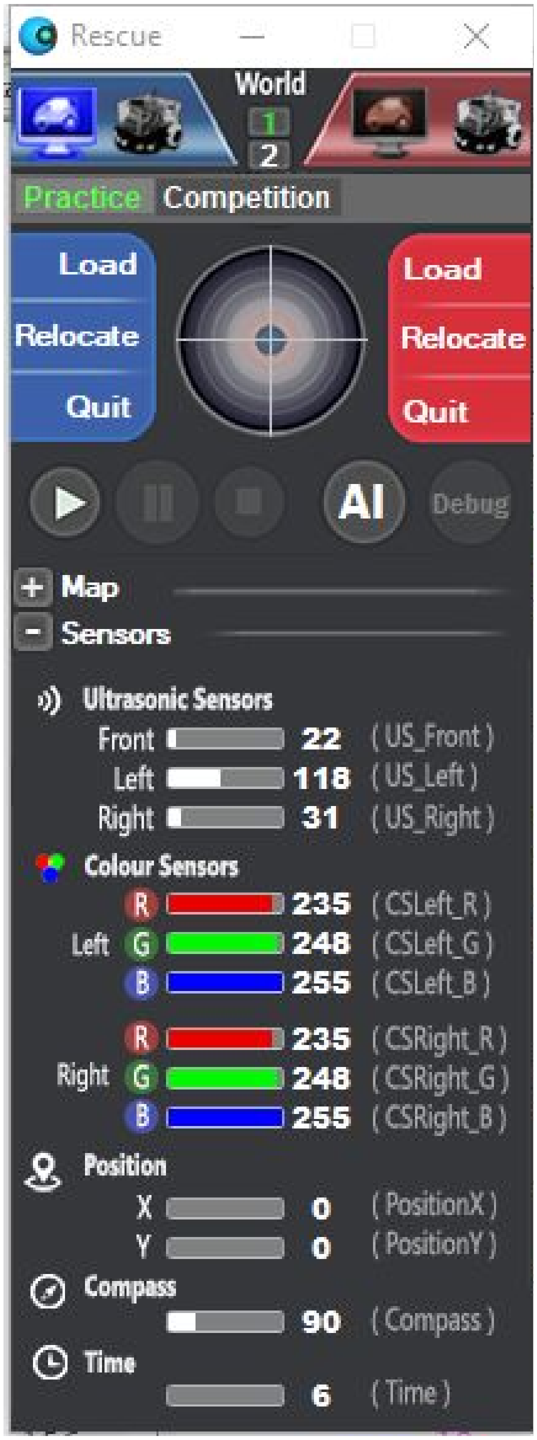
```
void Game0(){printf("hi1");}
void Game1(){printf("hi2");}
```

You can insert your team name into this function in the code



```
DLL_EXPORT char* GetTeamName()
{
    return "Teamless";
}
```

Run your code. Click Load (Blue for blue robot, red for red robot), select your file, and then play.



Ultrasonic Sensors
Front 22 (US_Front)
Left 118 (US_Left)
Right 31 (US_Right)

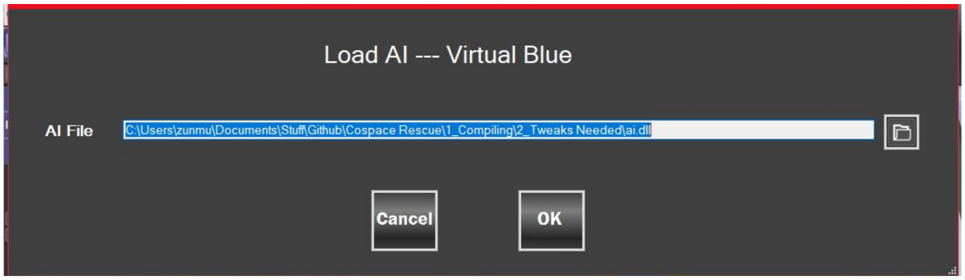
Colour Sensors
Left
R 235 (CSLeft_R)
G 248 (CSLeft_G)
B 255 (CSLeft_B)

Right
R 235 (CSRight_R)
G 248 (CSRight_G)
B 255 (CSRight_B)

Position
X 0 (PositionX)
Y 0 (PositionY)

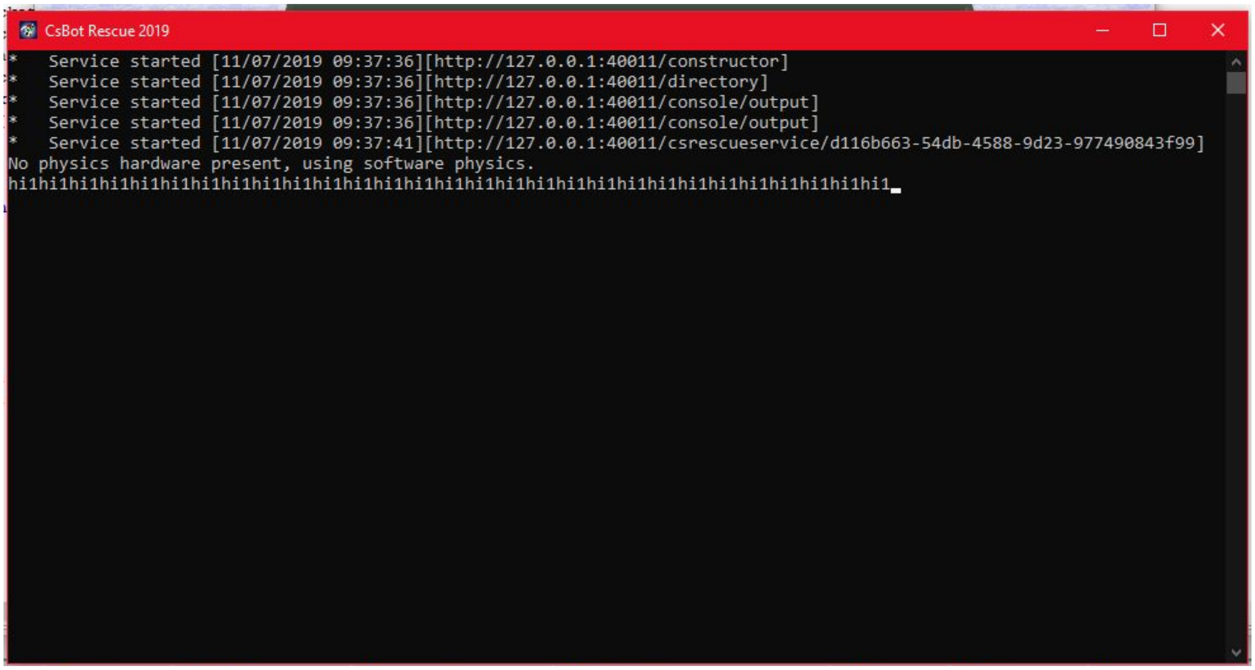
Compass
90 (Compass)

Time
6 (Time)



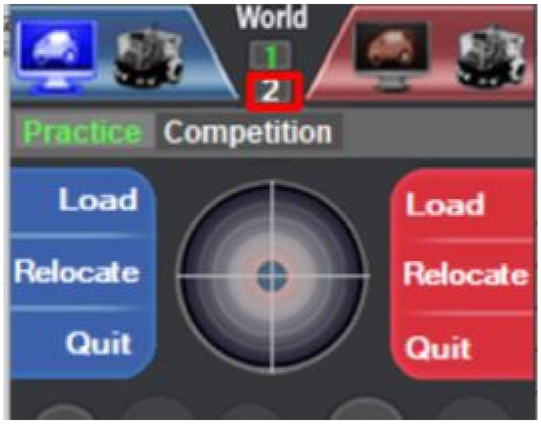
Load AI --- Virtual Blue

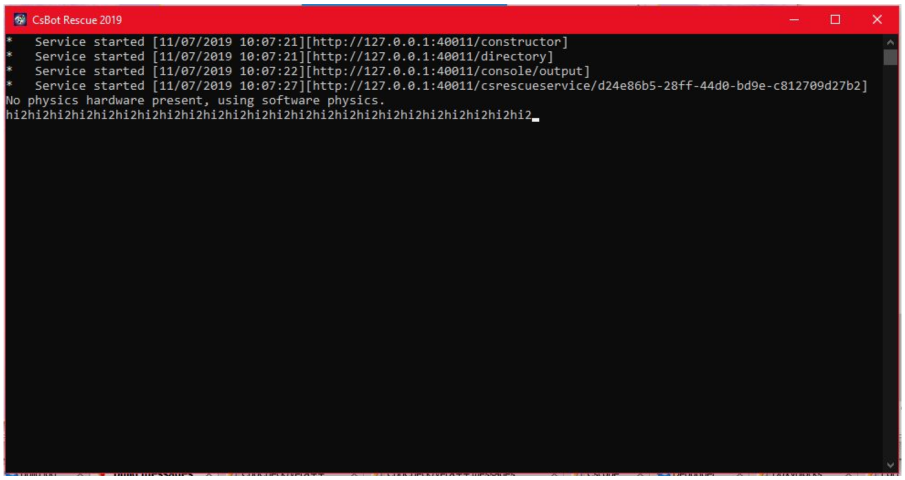
AI File C:\Users\zunmu\Documents\Stuff\Github\Cospace Rescue\1_Compiling\2_Tweaks Needed\ai.dll



```
*   Service started [11/07/2019 09:37:36][http://127.0.0.1:40011/constructor]
*   Service started [11/07/2019 09:37:36][http://127.0.0.1:40011/directory]
*   Service started [11/07/2019 09:37:36][http://127.0.0.1:40011/console/output]
*   Service started [11/07/2019 09:37:36][http://127.0.0.1:40011/console/output]
*   Service started [11/07/2019 09:37:41][http://127.0.0.1:40011/csrescueservice/d116b663-54db-4588-9d23-977490843f99]
No physics hardware present, using software physics.
hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1hi1_
```

When you click play your code in Game0() should run like above. However, to run your code in Game1(), click world2, load ai.dll again and then click play.





CsBot Rescue 2019

```
*   Service started [11/07/2019 10:07:21][http://127.0.0.1:40011/constructor]
*   Service started [11/07/2019 10:07:21][http://127.0.0.1:40011/directory]
*   Service started [11/07/2019 10:07:22][http://127.0.0.1:40011/console/output]
*   Service started [11/07/2019 10:07:27][http://127.0.0.1:40011/csrescueservice/d24e86b5-28ff-44d0-bd9e-c812709d27b2]
No physics hardware present, using software physics.
hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2hi2
```

Yay, the code works! So this is how to use Cospace Rescue Simulator in a nutshell. In the next tutorial, we would learn how to control the robot itself.