

無人載具技術與應用

ROS Computer Vision

徐瑋隆

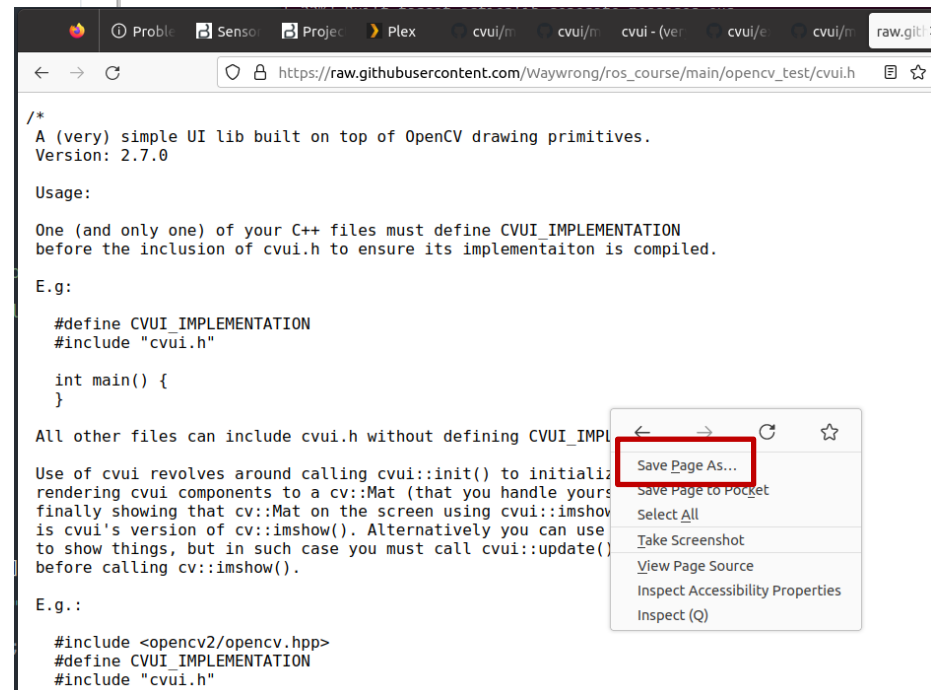
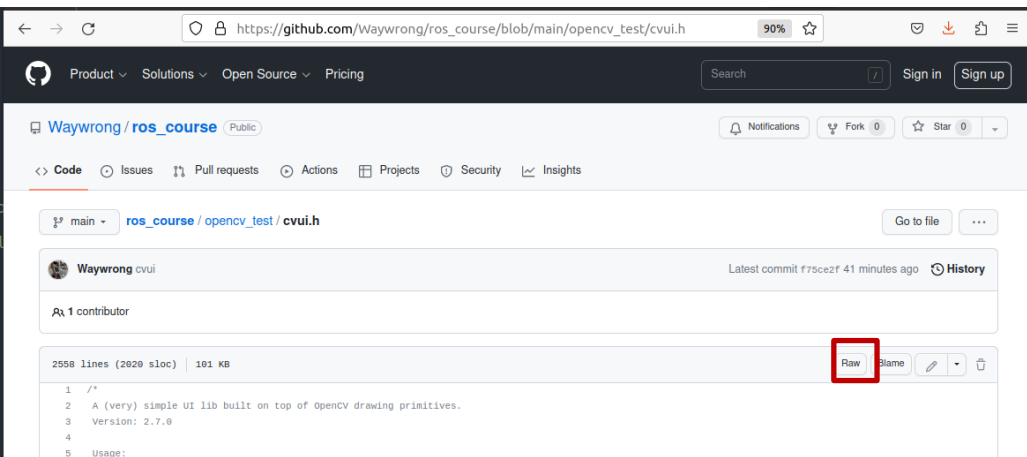
wlhsu304@gmail.com

OpenCV UI 01

ROS COMPUTER VISION

cv 01

https://github.com/Waywrong/ros_course/blob/main/opencv_test



cp ~/Downloads/cvui.h ~/projects/opencv/

cv 02

```
user@user-virtual-machine:~$ cd projects/opencv/  
user@user-virtual-machine:~/projects/opencv$ gedit cv_ui01.cpp &  
[1] 12622
```

Open



cv_ui01.cpp
~/projects/opencv

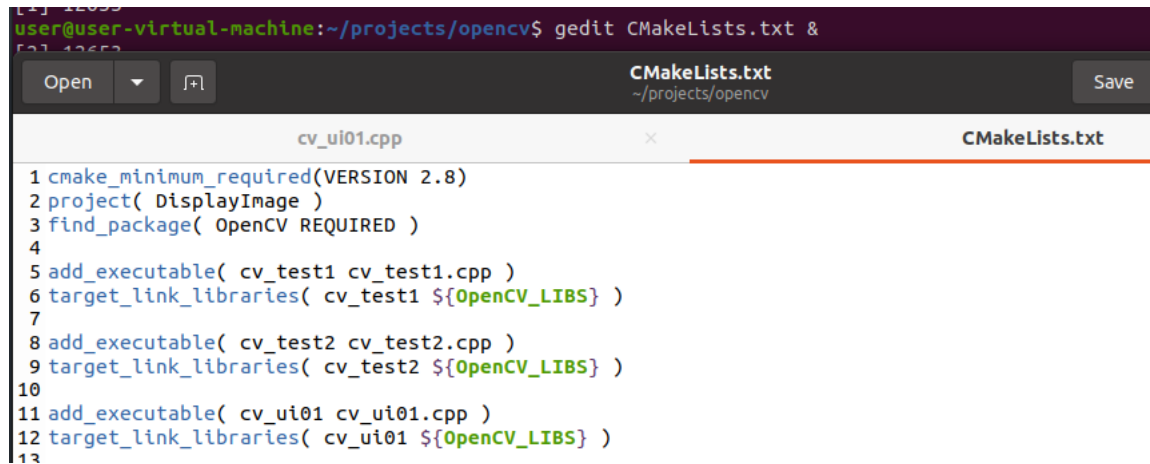
Save

```
1 #include <iostream>  
2 #include <opencv2/core.hpp>  
3 #include <opencv2/imgproc.hpp>  
4 #include <opencv2/highgui.hpp>  
5 #define CVUI_IMPLEMENTATION  
6 #include "cvui.h"  
7 #define WINDOW_NAME "Button"  
8  
9 int main(int argc, const char *argv[])  
10 {  
11     cv::Mat frame = cv::Mat(300, 600, CV_8UC3);  
12     cvui::init(WINDOW_NAME);  
13     std::string btn_name = "OFF";  
14     bool bBtn = false;  
15     while (true) {  
16         frame = cv::Scalar(49, 52, 49);  
17         cvui::text(frame, 150, 50, "Button example");  
18         if (cvui::button(frame, 360, 80, btn_name)) {  
19             std::cout << "Regular button clicked!" << std::endl;  
20             bBtn=!bBtn;  
21             btn_name = bBtn? "ON":"OFF";  
22         }  
23         cvui::update();  
24         cv::imshow(WINDOW_NAME, frame);  
25         // Check if ESC key was pressed  
26         if (cv::waitKey(20) == 27) {  
27             break;  
28         }  
29     }  
30     return 0;  
31 }
```

```
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#define WINDOW_NAME "Button"

int main(int argc, const char *argv[])
{
    cv::Mat frame = cv::Mat(300, 600, CV_8UC3);
    cvui::init(WINDOW_NAME);
    std::string btn_name = "OFF";
    bool bBtn = false;
    while (true) {
        frame = cv::Scalar(49, 52, 49);
        cvui::text(frame, 150, 50, "Button example");
        if (cvui::button(frame, 360, 80, btn_name)) {
            std::cout << "Regular button clicked!" << std::endl;
            bBtn=!bBtn;
            btn_name = bBtn? "ON":"OFF";
        }
        cvui::update();
        cv::imshow(WINDOW_NAME, frame);
        // Check if ESC key was pressed
        if (cv::waitKey(20) == 27) {
            break;
        }
    }
    return 0;
}
```

cv 03



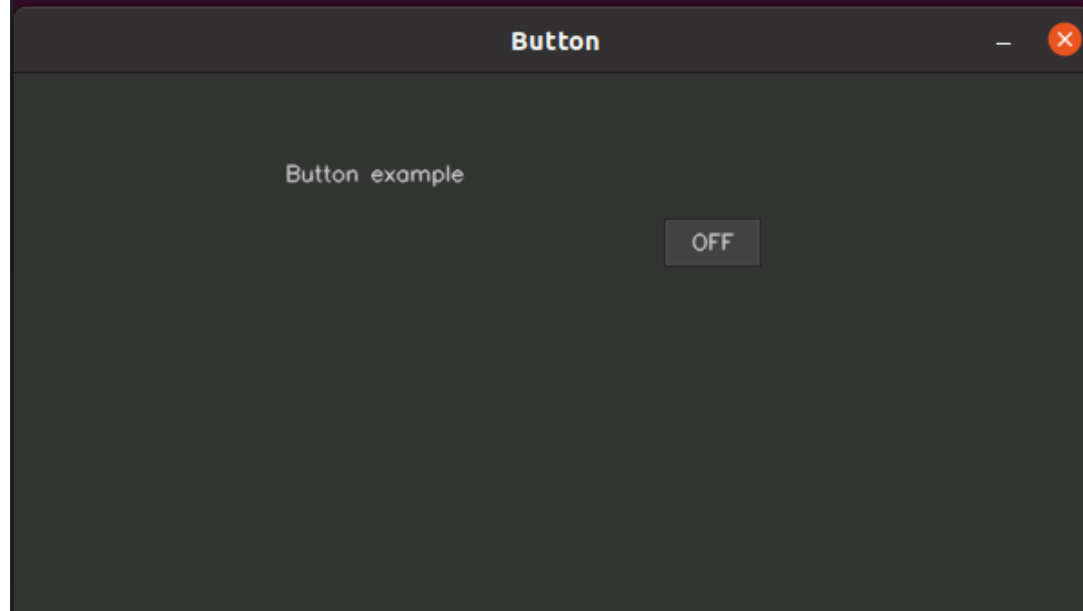
The image shows a terminal window at the top with the command `user@user-virtual-machine:~/projects/opencv$ gedit CMakeLists.txt &`. Below the terminal is a code editor window titled `CMakeLists.txt` with a path of `~/projects/opencv`. The editor has tabs for `cv_ui01.cpp` and `CMakeLists.txt`, with the latter being the active tab. The code in the editor is as follows:

```
1 cmake_minimum_required(VERSION 2.8)
2 project( DisplayImage )
3 find_package( OpenCV REQUIRED )
4
5 add_executable( cv_test1 cv_test1.cpp )
6 target_link_libraries( cv_test1 ${OpenCV_LIBS} )
7
8 add_executable( cv_test2 cv_test2.cpp )
9 target_link_libraries( cv_test2 ${OpenCV_LIBS} )
10
11 add_executable( cv_ui01 cv_ui01.cpp )
12 target_link_libraries( cv_ui01 ${OpenCV_LIBS} )
13
```

```
add_executable( cv_ui01 cv_ui01.cpp )
target_link_libraries( cv_ui01 ${OpenCV_LIBS} )
```

cv 04

```
user@user-virtual-machine:~/projects/opencv$ cd build/  
user@user-virtual-machine:~/projects/opencv/build$ make  
[ 25%] Built target cv_ui02  
[ 50%] Built target cv_ui01  
[ 75%] Built target cv_test2  
[100%] Built target cv_test1  
user@user-virtual-machine:~/projects/opencv/build$ ./cv_ui01  
□
```



OpenCV UI 02

ROS COMPUTER VISION

cv 01

```
ser@user-virtual-machine:~/projects/opencv/build$ cd ..
ser@user-virtual-machine:~/projects/opencv$ gedit cv_ui02.cpp
ser@user-virtual-machine:~/projects/opencv$
```

```
1 #include <iostream>
2 #include <opencv2/core/core.hpp>
3 #include <opencv2/highgui/highgui.hpp>
4
5 #define CVUI_IMPLEMENTATION
6 #include "cvui.h"
7
8 #define WINDOW_NAME "Trackbar and columns"
9
10 int main(int argc, const char *argv[])
11 {
12     int intValue1 = 30;
13     float floatValue1 = 12.;
14     cv::Mat frame = cv::Mat(cv::Size(450, 250), CV_8UC3);
15
16     // Size of trackbars
17     int width = 400;
18
19     cvui::init(WINDOW_NAME, 20);
20
21     while (true) {
22         frame = cv::Scalar(49, 52, 49);
23
24         cvui::beginColumn(frame, 20, 20, -1, -1, 6);
25         cvui::text("int trackbar, no customization");
26         cvui::trackbar(width, &intValue1, 0, 100);
27         cvui::space(5);
28
29         cvui::text("float trackbar, no customization");
30         cvui::trackbar(width, &floatValue1, 10.f, 15.f);
31         cvui::space(5);
32
33         std::cout<<"intValue1:"<<intValue1<<std::endl;
34         std::cout<<"floatValue1:"<<floatValue1<<std::endl;
35         if (cvui::button("&Quit")) {
36             break;
37         }
38         cvui::endColumn();
39
40         cvui::update();
41
42         cv::imshow(WINDOW_NAME, frame);
43     }
44
45     return 0;
46 }
```

```

#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

#define CVUI_IMPLEMENTATION
#include "cvui.h"

#define WINDOW_NAME "Trackbar and columns"

int main(int argc, const char *argv[])
{
    int intValue1 = 30;
    float floatValue1 = 12.;
    cv::Mat frame = cv::Mat(cv::Size(450, 250), CV_8UC3);

    // Size of trackbars
    int width = 400;

    cvui::init(WINDOW_NAME, 20);

```

```

while (true) {
    frame = cv::Scalar(49, 52, 49);

    cvui::beginColumn(frame, 20, 20, -1, -1, 6);
    cvui::text("int trackbar, no customization");
    cvui::trackbar(width, &intValue1, 0, 100);
    cvui::space(5);

    cvui::text("float trackbar, no customization");
    cvui::trackbar(width, &floatValue1, 10.f, 15.f);
    cvui::space(5);

    std::cout<<"intValue1:"<<intValue1<<std::endl;
    std::cout<<"floatValue1:"<<floatValue1<<std::endl;
    if (cvui::button("&Quit")) {
        break;
    }
    cvui::endColumn();

    cvui::update();


    cv::imshow(WINDOW_NAME, frame);
}

return 0;
}

```

cv 02

```
user@user-virtual-machine:~/projects/opencv$ gedit CMakeLists.txt &
```

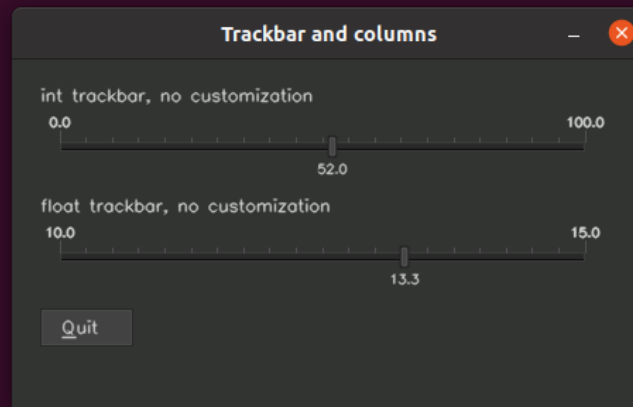


```
1 cmake_minimum_required(VERSION 2.8)
2 project( DisplayImage )
3 find_package( OpenCV REQUIRED )
4
5 add_executable( cv_test1 cv_test1.cpp )
6 target_link_libraries( cv_test1 ${OpenCV_LIBS} )
7
8 add_executable( cv_test2 cv_test2.cpp )
9 target_link_libraries( cv_test2 ${OpenCV_LIBS} )
10
11 add_executable( cv_ui01 cv_ui01.cpp )
12 target_link_libraries( cv_ui01 ${OpenCV_LIBS} )
13
14 add_executable( cv_ui02 cv_ui02.cpp )
15 target_link_libraries( cv_ui02 ${OpenCV_LIBS} )
```

```
add_executable( cv_ui02 cv_ui02.cpp )
target_link_libraries( cv_ui02 ${OpenCV_LIBS} )
```

cv 03

```
user@user-virtual-machine:~/projects/opencv$ cd build/  
user@user-virtual-machine:~/projects/opencv/build$ make  
[ 25%] Built target cv_ui02  
[ 50%] Built target cv_ui01  
[ 75%] Built target cv_test2  
[100%] Built target cv_test1  
user@user-virtual-machine:~/projects/opencv/build$ ./cv_ui02
```

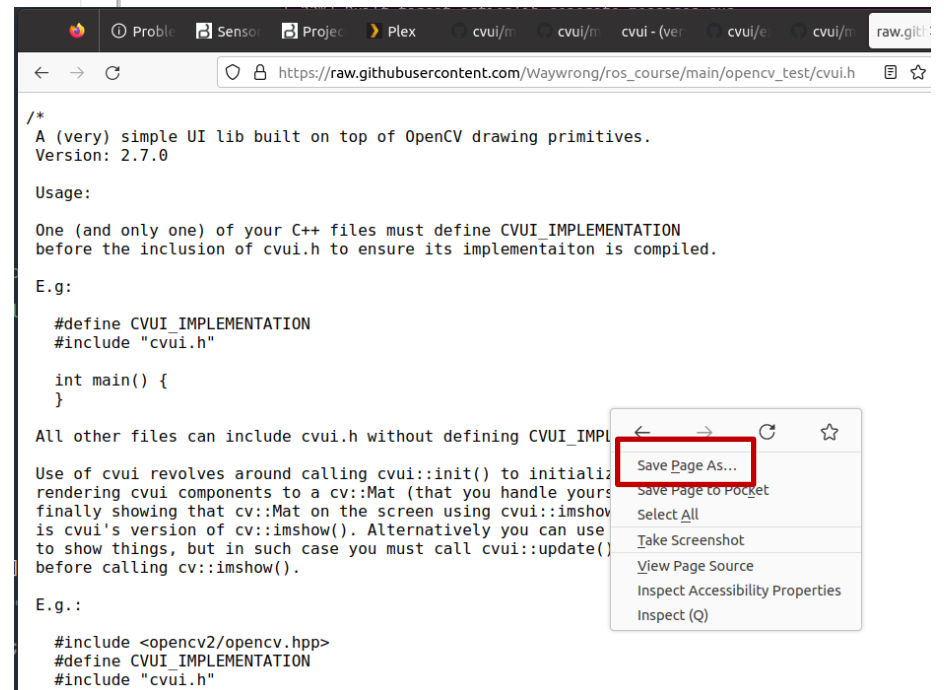
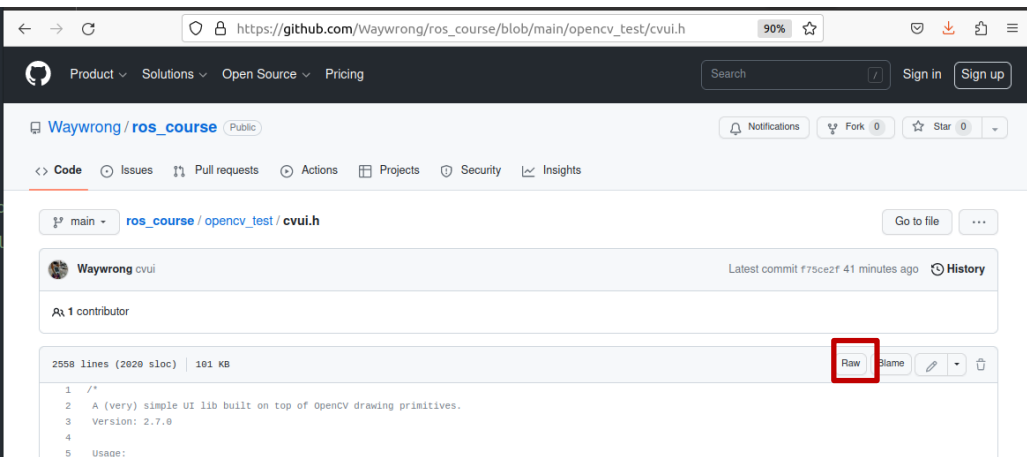
[illegible]

ROS + OpenCV UI : Topic

ROS COMPUTER VISION

cv – topic 01

https://github.com/Waywrong/ros_course/blob/main/opencv_test



cp ~/Downloads/cvui.h ~/catkin_ws/src/my_cv/src/

```
user@user-virtual-machine:~$ cd catkin_ws/src/my_cv/src/
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ ls
myNodeCV01.cpp  myNodeCV02.cpp
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ cp ~/Downloads/cvui.h .
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ ls
cvui.h  myNodeCV01.cpp  myNodeCV02.cpp
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ gedit myNode_ui01.cpp
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$
```

*Untitled Document 1 x myNodeCV01.cpp x myNodeCV02.cpp x myNode_ui01.cpp x

```
1 #include "ros/ros.h"
2 #include <iostream>
3 #include <opencv2/core.hpp>
4 #include <opencv2/imgproc.hpp>
5 #include <opencv2/highgui.hpp>
6 #define CVUI_IMPLEMENTATION
7 #include "cvui.h"
8 #define WINDOW_NAME "Button"
9
10 cv::Mat frame;
11 std::string btn_name = "OFF";
12 bool bBtn = false;
13 void img_Draw(void)
14 {
15     frame = cv::Scalar(49, 52, 49);
16     cvui::text(frame, 150, 50, "Button example");
17     if (cvui::button(frame, 360, 80, btn_name)) {
18         std::cout << "Regular button clicked!" << std::endl;
19         bBtn=!bBtn;
20         btn_name = bBtn? "ON":"OFF";
21     }
22     cvui::update();
23     cv::imshow(WINDOW_NAME, frame);
24     cv::waitKey(30);
25 }
26
27 int main(int argc, char **argv)
28 {
29     ros::init(argc, argv, "myNode_ui01");
30     ros::NodeHandle n;
31     frame = cv::Mat(300, 600, CV_8UC3);
32     cvui::init(WINDOW_NAME);
33     ros::Rate r(30);
34     while(ros::ok())
35     {
36         img_Draw();
37         r.sleep();
38         ros::spinOnce();
39     }
40     return 0;
41 }
```

```

#include "ros/ros.h"
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#define WINDOW_NAME "Button"

```

```

cv::Mat frame;
std::string btn_name = "OFF";
bool bBtn = false;
void img_Draw(void)
{
    frame = cv::Scalar(49, 52, 49);
    cvui::text(frame, 150, 50, "Button example");
    if (cvui::button(frame, 360, 80, btn_name)) {
        std::cout << "Regular button clicked!" << std::endl;
        bBtn=!bBtn;
        btn_name = bBtn? "ON":"OFF";
    }
    cvui::update();
    cv::imshow(WINDOW_NAME, frame);
    cv::waitKey(30);
}

```

```

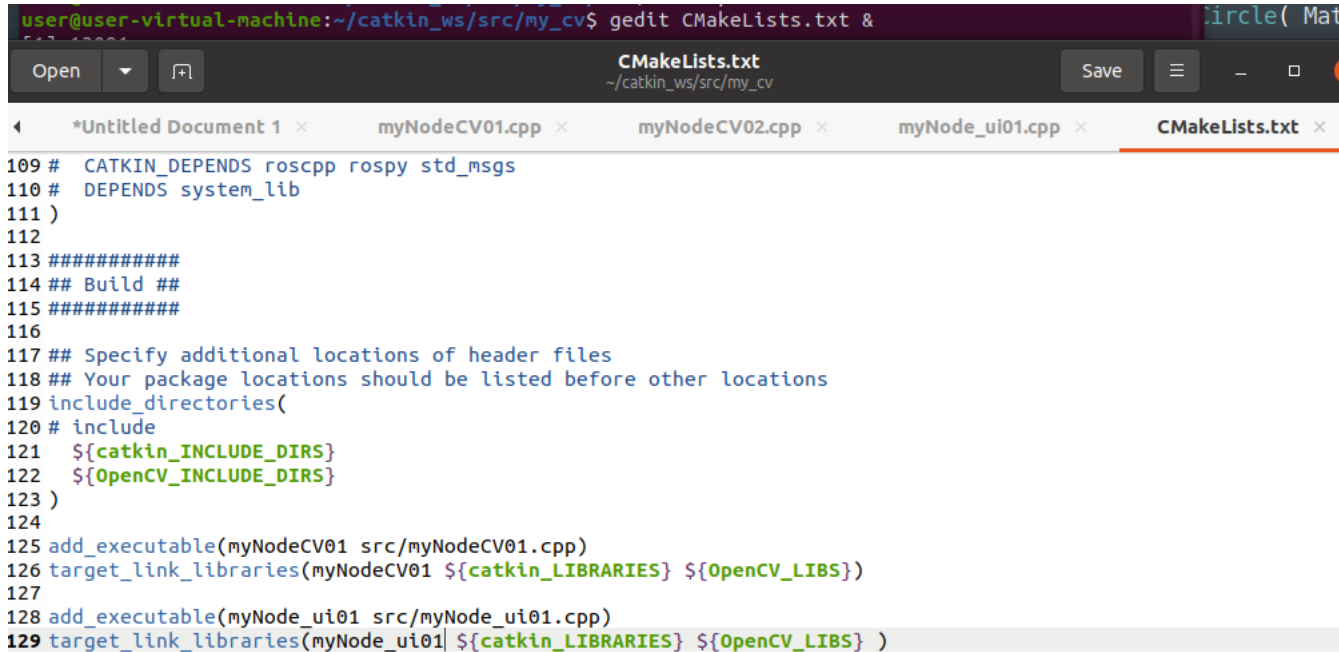
int main(int argc, char **argv)
{
    ros::init(argc, argv, "myNode_ui01");
    ros::NodeHandle n;
    frame = cv::Mat(300, 600, CV_8UC3);
    cvui::init(WINDOW_NAME);
    ros::Rate r(30);
    while(ros::ok())
    {
        img_Draw();
        r.sleep();
        ros::spinOnce();
    }
    return 0;
}

```



```
find_package(OpenCV REQUIRED)
```

```
include_directories(  
  ${catkin_INCLUDE_DIRS}  
  ${OpenCV_INCLUDE_DIRS}  
)
```



```
user@user-virtual-machine:~/catkin_ws/src/my_cv$ gedit CMakeLists.txt &  
CMakeLists.txt  
~/catkin_ws/src/my_cv  
*Untitled Document 1 x myNodeCV01.cpp x myNodeCV02.cpp x myNode_ui01.cpp x CMakeLists.txt x  
109 # CATKIN_DEPENDS roscpp rospy std_msgs  
110 # DEPENDS system_lib  
111 )  
112  
113 #####  
114 ## Build ##  
115 #####  
116  
117 ## Specify additional locations of header files  
118 ## Your package locations should be listed before other locations  
119 include_directories(  
120 # include  
121 ${catkin_INCLUDE_DIRS}  
122 ${OpenCV_INCLUDE_DIRS}  
123 )  
124  
125 add_executable(myNodeCV01 src/myNodeCV01.cpp)  
126 target_link_libraries(myNodeCV01 ${catkin_LIBRARIES} ${OpenCV_LIBS})  
127  
128 add_executable(myNode_ui01 src/myNode_ui01.cpp)  
129 target_link_libraries(myNode_ui01 ${catkin_LIBRARIES} ${OpenCV_LIBS} )
```

```
add_executable(myNode_ui01 src/myNode_ui01.cpp)  
target_link_libraries(myNode_ui01 ${catkin_LIBRARIES} ${OpenCV_LIBS} )
```

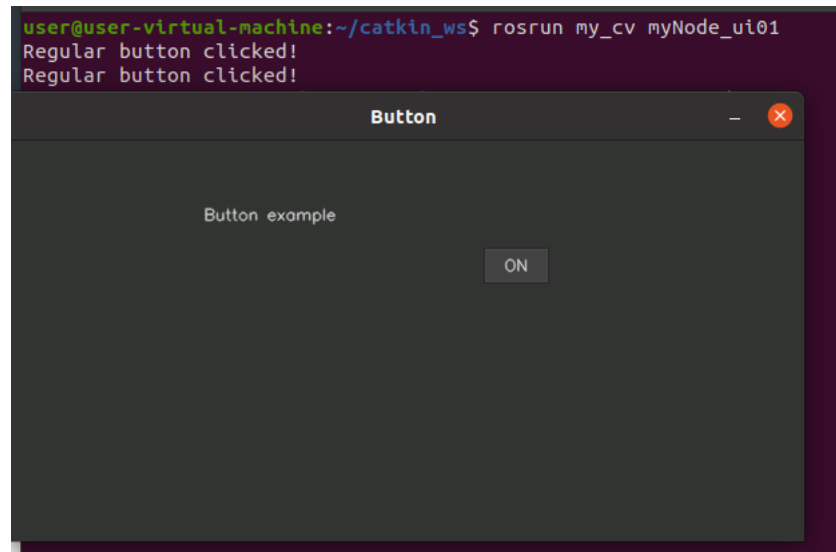
```
cd ~/catkin_ws  
catkin_make
```

編譯程式

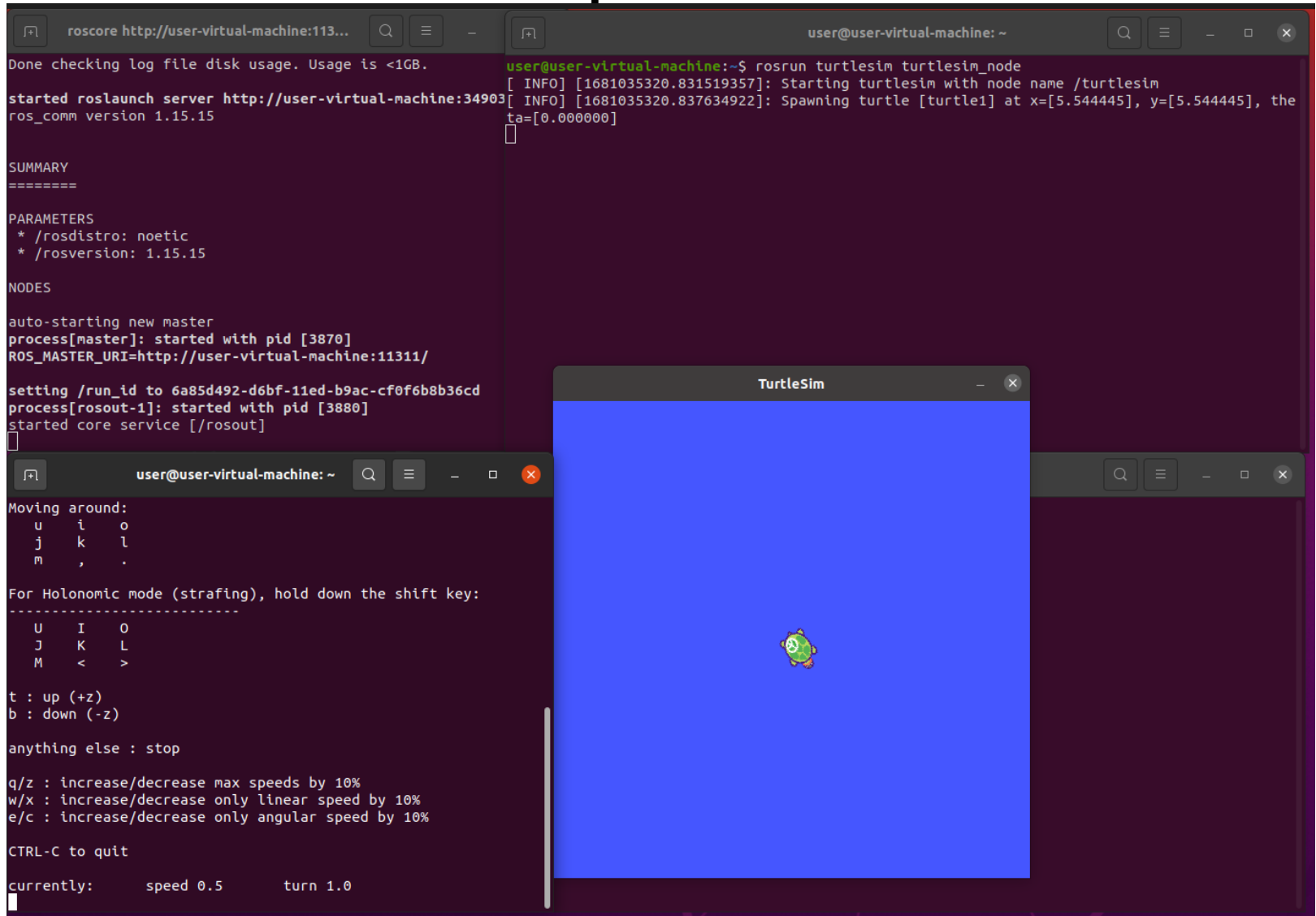
cv – topic 02

roscore

roslaunch my_cv myNode_ui01



cv – topic 03



The screenshot shows a terminal window with three panes. The top-left pane shows the output of `roscore`, including disk usage checks, master startup logs, and a summary of parameters and nodes. The top-right pane shows the output of `roslaunch` for `turtlesim`, displaying INFO messages about starting the node and spawning a turtle. The bottom pane shows the `teleop_twist_keyboard` interface, including movement instructions (u, j, m for forward, left, right; U, J, M for strafing), speed/turn controls (t, b, q/z, w/x, e/c), and a status line showing current speed and turn values.

```
roscore http://user-virtual-machine:113...
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://user-virtual-machine:34903/
ros_comm version 1.15.15

SUMMARY
=====

PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.15.15

NODES
auto-starting new master
process[roscpp]: started with pid [3870]
ROS_MASTER_URI=http://user-virtual-machine:11311/

setting /run_id to 6a85d492-d6bf-11ed-b9ac-cf0f6b8b36cd
process[roscpp-1]: started with pid [3880]
started core service [/roscpp]

user@user-virtual-machine:~$ roslaunch turtlesim turtlesim_node
[ INFO] [1681035320.831519357]: Starting turtlesim with node name /turtlesim
[ INFO] [1681035320.837634922]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], the
ta=[0.000000]

user@user-virtual-machine:~$
Moving around:
u    i    o
j    k    l
m    ,    .

For Holonomic mode (strafing), hold down the shift key:
-----
U    I    O
J    K    L
M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:    speed 0.5    turn 1.0
```

roscore

roslaunch turtlesim turtlesim_node

roslaunch teleop_twist_keyboard teleop_twist_keyboard.py /cmd_vel:=/turtle1/cmd_vel

cv – topic 04

複習rosservice

rosservice list

rosservice call /spawn 1 2 0 kk

rosparam list

rosparam get /turtlesim/background_r

rosparam set /turtlesim/background_r 150

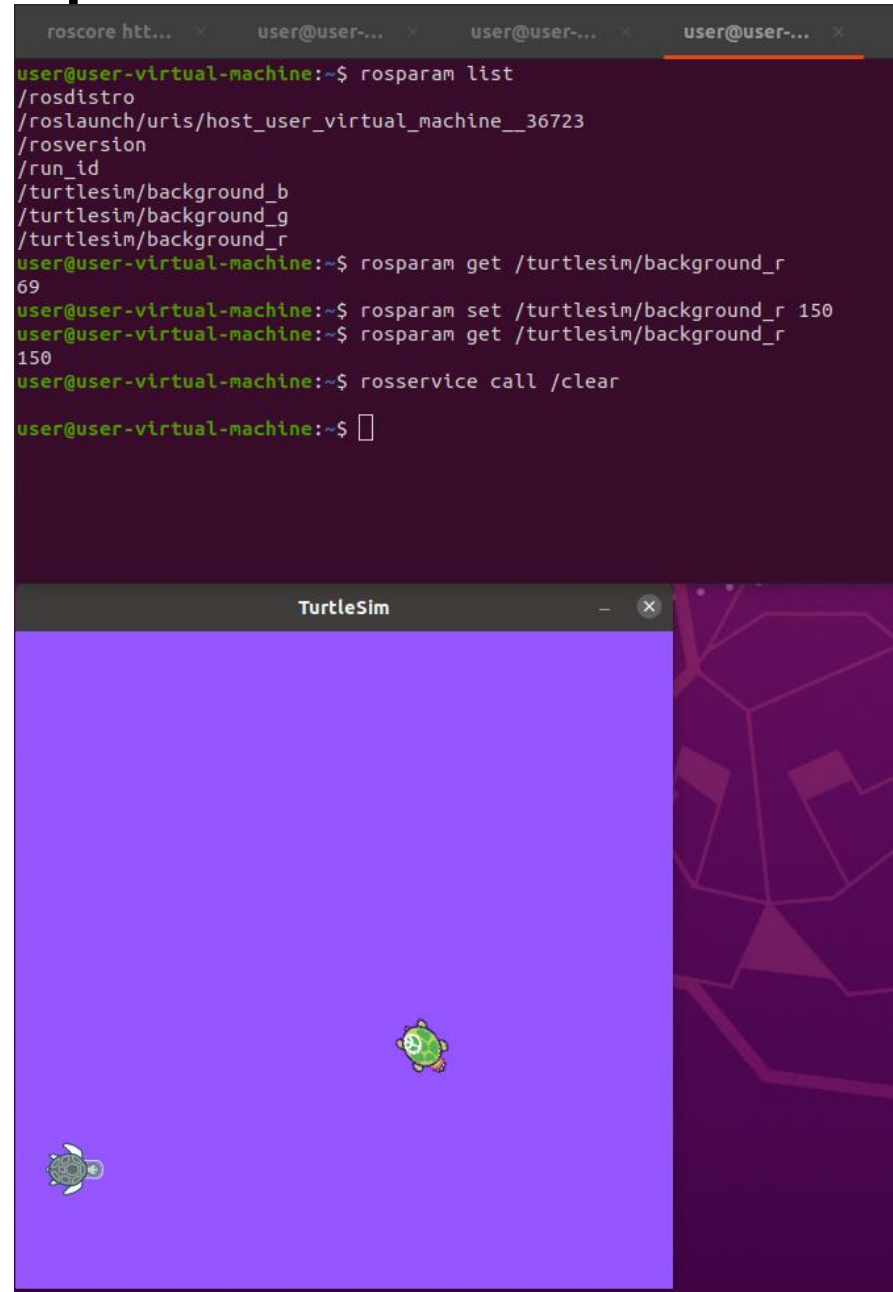
rosparam get /turtlesim/background_r

rosservice call /clear

清潔背景(強制程式讀取取ros param改變才會生效)

rosparam set /turtlesim/background_r 50

rosservice call /clear



cv – topic 05

```
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ gedit myNode_ui02.cpp &

myNode_ui02.cpp
~/catkin_ws/src/my_cv/src

myNodeCV02.cpp x myNode_ui01.cpp x CMakeLists.txt x package.xml x myNode_ui02.cpp x

1 #include "ros/ros.h"
2 #include <iostream>
3 #include <opencv2/core.hpp>
4 #include <opencv2/imgproc.hpp>
5 #include <opencv2/highgui.hpp>
6 #define CVUI_IMPLEMENTATION
7 #include "cvui.h"
8 #include <geometry_msgs/Twist.h>
9 #define WINDOW_NAME "Button"
10
11 cv::Mat frame;
12 ros::Publisher cmdpub;
13
14 void my_cmd_vel(float _linear, float _angular)
15 {
16     geometry_msgs::Twist twist;
17     geometry_msgs::Vector3 linear;
18     linear.x=_linear;
19     linear.y=0;
20     linear.z=0;
21     geometry_msgs::Vector3 angular;
22     angular.x=0;
23     angular.y=0;
24     angular.z=_angular;
25     twist.linear=linear;
26     twist.angular=angular;
27     cmdpub.publish(twist);
28 }
29 void img_Draw(void)
30 {
31     frame = cv::Scalar(49, 52, 49);
32     cvui::text(frame, 150, 50, "Button example");
33     if (cvui::button(frame, 360, 80, "up")) {
34         std::cout << "up button clicked!" << std::endl;
35         my_cmd_vel(1,0);
36     }
37     cvui::update();
38     cv::imshow(WINDOW_NAME, frame);
39     cv::waitKey(30);
40 }
41
42 int main(int argc, char **argv)
43 {
44     ros::init(argc, argv, "myNode_ui01");
45     ros::NodeHandle n;
46     cmdpub = n.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel", 10);
```

```

#include "ros/ros.h"
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#include <geometry_msgs/Twist.h>
#define WINDOW_NAME "Button"

cv::Mat frame;
ros::Publisher cmdpub;

void my_cmd_vel(float _linear, float _angular)
{
    geometry_msgs::Twist twist;
    geometry_msgs::Vector3 linear;
    linear.x=_linear;
    linear.y=0;
    linear.z=0;
    geometry_msgs::Vector3 angular;
    angular.x=0;
    angular.y=0;
    angular.z=_angular;
    twist.linear=linear;
    twist.angular=angular;
    cmdpub.publish(twist);
}

```

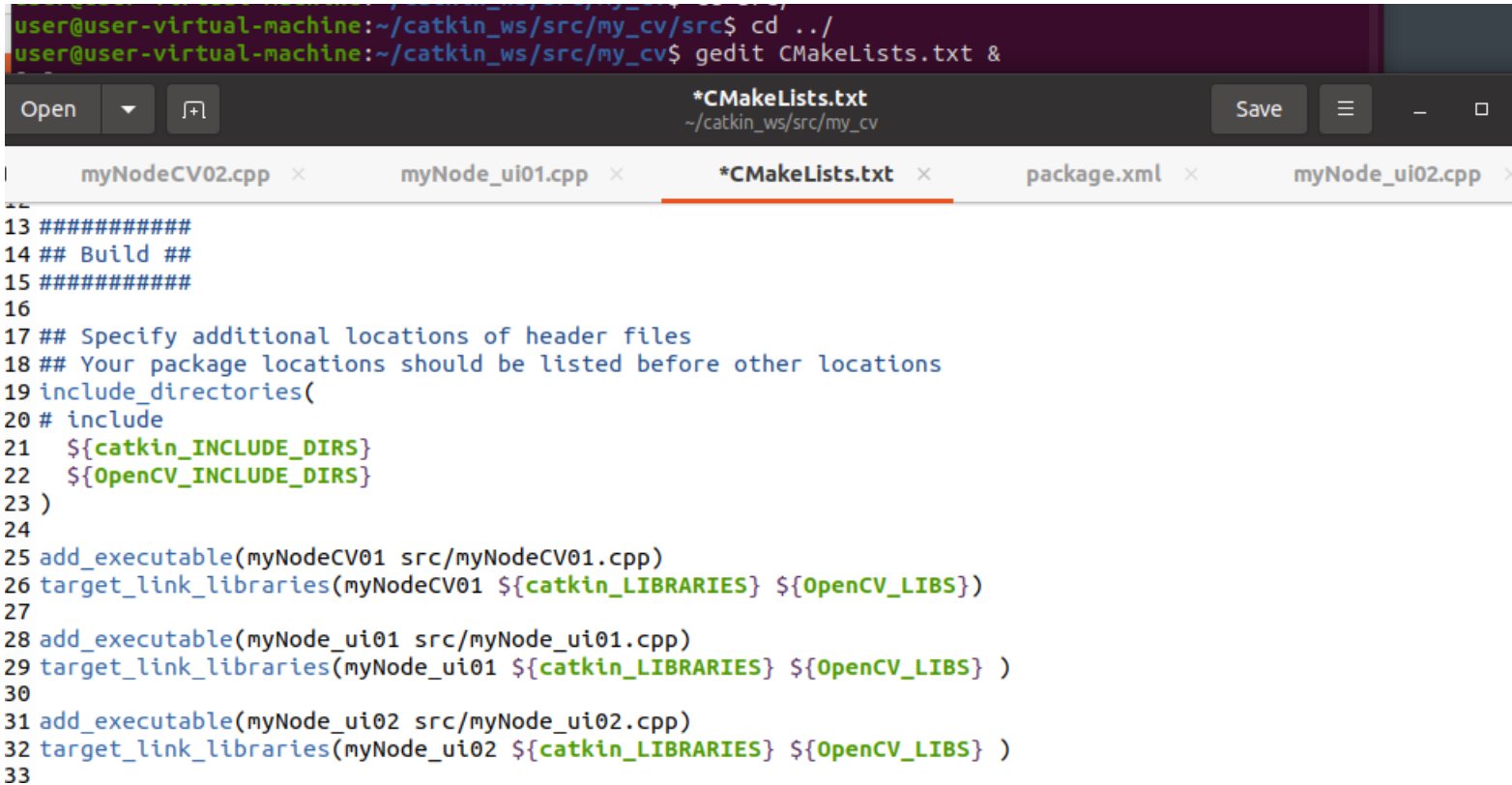
```

void img_Draw(void)
{
    frame = cv::Scalar(49, 52, 49);
    cvui::text(frame, 150, 50, "Button example");
    if (cvui::button(frame, 360, 80, "up")) {
        std::cout << "up button clicked!" << std::endl;
        my_cmd_vel(1,0);
    }
    cvui::update();
    cv::imshow(WINDOW_NAME, frame);
    cv::waitKey(30);
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "myNode_ui01");
    ros::NodeHandle n;
    cmdpub = n.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel",
10);
    frame = cv::Mat(300, 600, CV_8UC3);
    cvui::init(WINDOW_NAME);
    ros::Rate r(30);
    while(ros::ok())
    {
        img_Draw();
        r.sleep();
        ros::spinOnce();
    }
    return 0;
}

```

```
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ cd ../
user@user-virtual-machine:~/catkin_ws/src/my_cv$ gedit CMakeLists.txt &
```



```
13 #####
14 ## Build ##
15 #####
16
17 ## Specify additional locations of header files
18 ## Your package locations should be listed before other locations
19 include_directories(
20 # include
21 ${catkin_INCLUDE_DIRS}
22 ${OpenCV_INCLUDE_DIRS}
23 )
24
25 add_executable(myNodeCV01 src/myNodeCV01.cpp)
26 target_link_libraries(myNodeCV01 ${catkin_LIBRARIES} ${OpenCV_LIBS})
27
28 add_executable(myNode_ui01 src/myNode_ui01.cpp)
29 target_link_libraries(myNode_ui01 ${catkin_LIBRARIES} ${OpenCV_LIBS} )
30
31 add_executable(myNode_ui02 src/myNode_ui02.cpp)
32 target_link_libraries(myNode_ui02 ${catkin_LIBRARIES} ${OpenCV_LIBS} )
33
```

```
add_executable(myNode_ui02 src/myNode_ui02.cpp)
target_link_libraries(myNode_ui02 ${catkin_LIBRARIES} ${OpenCV_LIBS} )
```

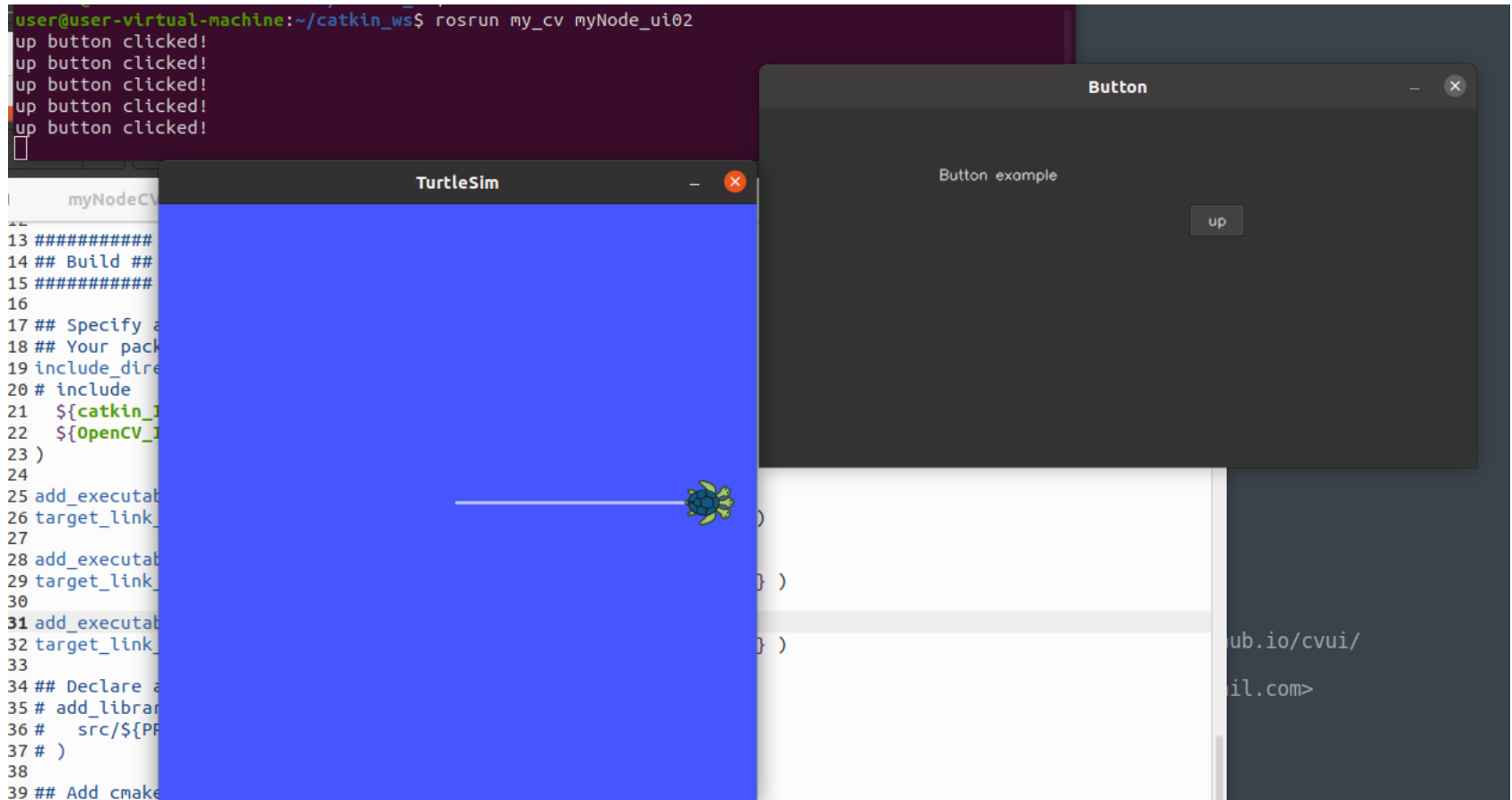
```
cd ~/catkin_ws
catkin_make
```

編譯程式

roscore

roslaunch turtlesim turtlesim_node

roslaunch my_cv myNode_ui02



練習

