Barrett BURT Robot Quick Guide

2017. 09. 24 Nan Wei

1 Starting Instructions

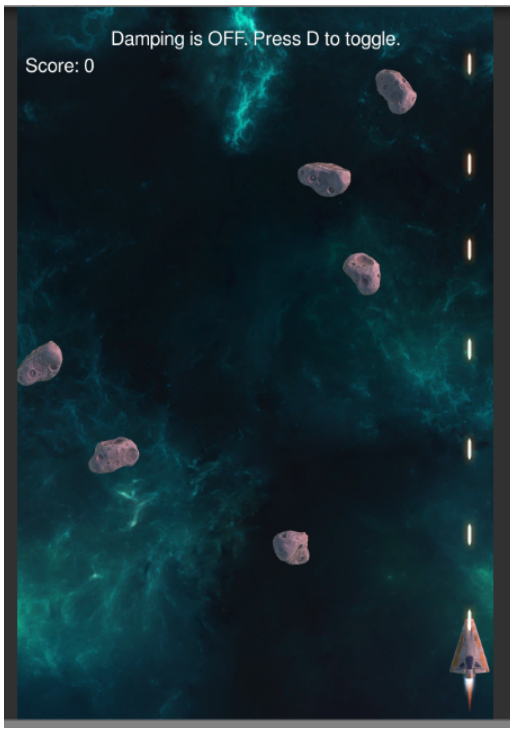
- Before you start, take a look at the installation guide and mounting layout. Ensure the wheels of the T-plate are locked and the all wires and cables are correctly connected. Check if any obstacle is in the work space of the robot.
- Turn on the toggle switch on the back side of the console (the side facing the screen) for both robots. Adjust the heights by pressing the up/down button if needed.
- Turn on one of the computers on the front side of the computer. I used the one on the right to develop programs.
- The password is 'proficio' (with no apostrophe).
- Change the monitor setting if the aspect ratio is not right: Menu/Settings -> Picture -> Aspect Ratio -> Just Scan
- When developing programs, always sit outside of the work space of the robots to avoid danger. I never used the wheelchair but always sat behind the robots to be safe. Then last thing you want is to be hit by them.

2 Program BURT with Unity

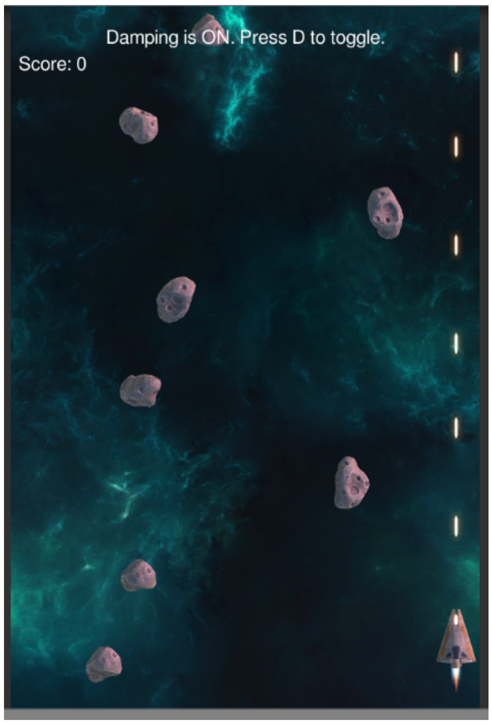
Before programming BURT, you may want to go through the Unity Roll-a-Ball tutorial which will give you some basic background of C sharp and Unity.

The official repository for the BURT robot haptic programs is here. Note that the link to the Unity deb file is wrong. It should be this for the correct version (5.6.0xb6). The program will not run if you just downloaded it. In Unity, go to the Hierarchy list and select RobotConnection, you will see in the Inspector that the script is not selected. Point it to Assets - Scripts - dll - ProficioClient - RobotConnection, and the program will work if you press play. You can select which robot to run by specifying the IP address in RobotConnection, 192.168.100.200 for left and 192.168.100.201 for right. Go through the Hierarchy and Assets-Scripts to understand the coding structure. Barrett provided very clear explanations in their code, and you can also explore the Unity API for extra help.

I also recommended the space shooter tutorial as it involves more advanced functions that will be helpful when you program on your own. It is also simply fun to code a classic arcade game like the space shooter.



(a) Damping is off.



(b) Damping is on.

Figure 1: Screen shots of the Space Shooter program.

There are 2 Unity projects on the desktop of the computer on the right. One of them is a modified space shooter that can be played using the left arm. The other is a haptic world program for dual-arm configuration.

2.1 Space Shooter

This is in essence the tutorial space shooter program fitted to run with the BURT platform. The differences are:

- Instead of using the keyboard to control the velocity of the spacecraft, we directly use the end-effector position with some translation and scaling to fit to the screen dimension. This transformation needs to be adjusted if you wish to run with the right arm.
- The spacecraft automatically fires at a fixed rate with no need to press the space button. This fire rate can be modified under the player object.
- Added damping to increase the resistance. The player can press "D" to turn the damping on or off. The damping coefficient can also be modified under the player object, with default value of 8. Try to be conservative when adjusting gain values such as damping and stiffness for stability.
- When the game is over, ideally the player can press "R" and restart. However, with robots involved I have not figured out a way to do this. The robot does not activate again after pressing "R." To restart, you will need to manually place the robot back to home position, stop the game, and press play again.

2.2 Haptic World

I consider the haptic world an advanced version of the haptic program in the repository.

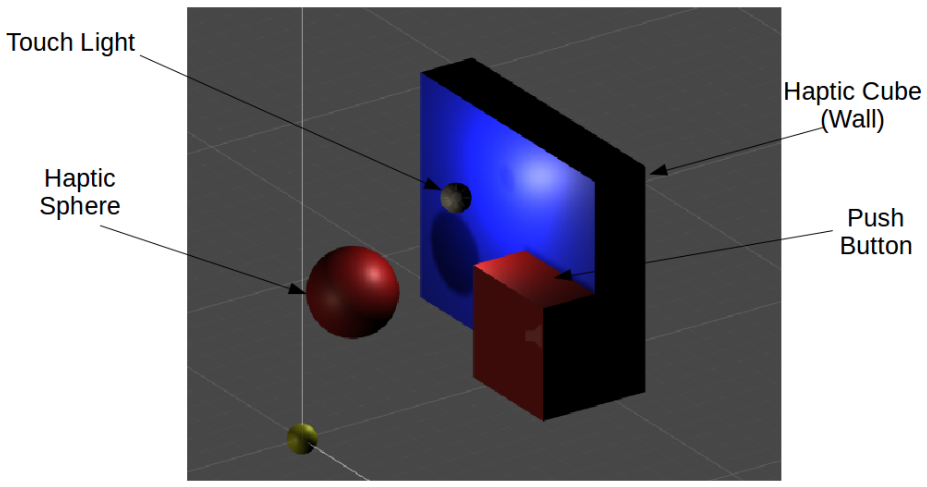


Figure 2: Screen shot of the Haptic World program.

- The program is modified to allow the operation of both arms simultaneously. This is done by created 2 RobotConnections and 2 player objects with minor modifications to the scripts.
- The haptic sphere object is the same as the original program.
- The size and position of the haptic cube is modified to simulate a wall at the back.
- A touch light is created using a sub-class of the haptic sphere. This requires overriding some functions of the parent class which was interesting to do. When a contact between the touch light and the hand is detected, the light is turned on by setting the light color to white. When touched again the program sets the light color to black which effectively turns it off.
- A (very big) push button is created near the bottom-right hand side of the wall. Similar to the touch light, this comes from a sub-class of the haptic cube. When been pressed in the z-direction deep enough, the program toggles the status of the switch and turns a directional light on/off. The length of the push button in the z-direction is different depending on its on/off status similar to a real push button.

3 Notes

- The C sharp repository is here. It is fairly self-explanatory so I will not provide discussions here. Note that the "homing" function does not command the robot to return to home position; instead it records its current configuration as the home position. DO NOT SET THE HOME POSITION TO OTHER CONFIGURATIONS, or the robot will be confused when applying gravity compensation and be unstable. If you are not sure, manually put the robot to the initial configuration and press "h" before quitting. The explanation that I received from Amy is as follows:

"The main advantage of this method is that now you don't have to reboot the console to change configurations. If you flip between right- and left-handed configurations, you can re-home the robot from one of the console applications. As long as you manually re-home before enabling, the robot should function normally. (The Unity applications don't have this feature, but you can run a console application before starting the Unity application.) We are still working towards having the robot detect its home position automatically, but that feature won't be available for at least a few more weeks."

- You can also specify the IP address to use either arm in C sharp programs by modifying the line

  robot = new RobotClient ();

  to

  robot = new RobotClient (IP Address, Port);

  For our robots, it is

  robot = new RobotClient ("192.168.100.200", 5683);

  for the left arm, or

  robot = new RobotClient ("192.168.100.201", 5683);

  for the right arm.

- Again, use extra caution when developing programs. I had experiences of the BURT going unstable and it was quite scary when it operates at high torques. Also, tune the gains gradually to ensure stability.

- One known issue is that the right arm slightly overcompensates for gravity. You can observe this by turning it on and see the robot slowly move upward instead of staying still. I believe this is because it uses a pre-set mass value but it is slightly different in our set-up. Our WAM robot has a function of gravity calibration that corrects these values, and I believe Barrett will develop the same algorithm for BURT.

- The ROS interface is broken after the firmware update for Unity. I am not sure if they will work on this again, but most likely your project can be accomplished by Unity and C sharp.

4 Firmware Update Instructions

Below is the firmware update instruction that I got from Amy in case if you need it in the future.

Upgrade Instructions:

To set up for an upgrade you will need a PEAK CAN USB to CAN adapter (provided by us). This should be plugged into the service port on the front of the console and then into one of the USB ports on the front of the console.

To upgrade you first need to install btflash. This will only need to be done the first time you upgrade, if you don't have btflash installed. To check if you have this installed open up a terminal and type:

```
btflash –version
```

If you have btflash you will get a response similar to the below text. Please note that the version number may be different.

```
btflash, version 0.2.0
```

To install btflash follow the installation instructions in the first section from our btflash repository. Do not follow the "setting up for development" section: https://git.barrett.com/software/btflash. You need to be connected to the internet to install btflash.

Before using btflash you will need to guarantee that the CAN adapter is plugged in and then run the following commands. These commands will configure:

```
sudo ip link set can0 type can bitrate 1000000 sudo ip link set up can0
```

To perform an upgrade you will need a .bin file provided by Barrett. You can carry out the upgrade using the following series of commands:

```
btflash flash 10 path/to/main-board-comm.<version>.bin btflash reset 10
```

For some context the Main Board, which is what runs the control firmware and ROS talks to, has an id of 10. In the above series of commands you are telling btflash to flash (write to the program memory of the device) the firmware from the defined file to the device with id 10. The reset commands provides a soft reset for the firmware on the Main Board. You should not need to power cycle, but if issues occur a power cycle is recommended.

You should follow these instructions for both consoles to upgrade both Main Boards. You may use the same console USB port, but you will need to plug the USB to CAN adapter into the CAN Service port on the front of both consoles to perform the upgrade.