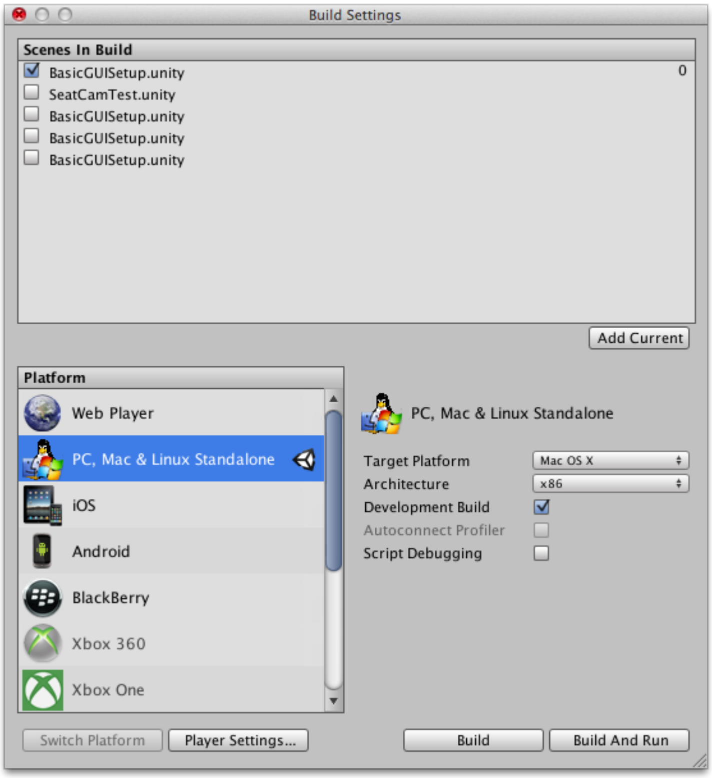
Instructions to OZ system

What you need

- Node.js
- Ternimal or something else to run Node.js
- Unity

Step by Step Set-up

1. Download “OzMachine-master” and “ozUnity-master” from github. The OzMachine-master will run as server and the ozUnity will run as clients.
2. Open Unity and open OzUnity-master project. “File → open project → OzUnity-master”
3. Build the project to your platform. Here we chose to build an application for Mac OS. “File → Build settings → pc, Mac, Linux → Build → rename: ‘pc1’ --> save ”.



Build Settings

Scenes In Build

☑ BasicGUISetup.unity 0
☐ SeatCamTest.unity
☐ BasicGUISetup.unity
☐ BasicGUISetup.unity
☐ BasicGUISetup.unity

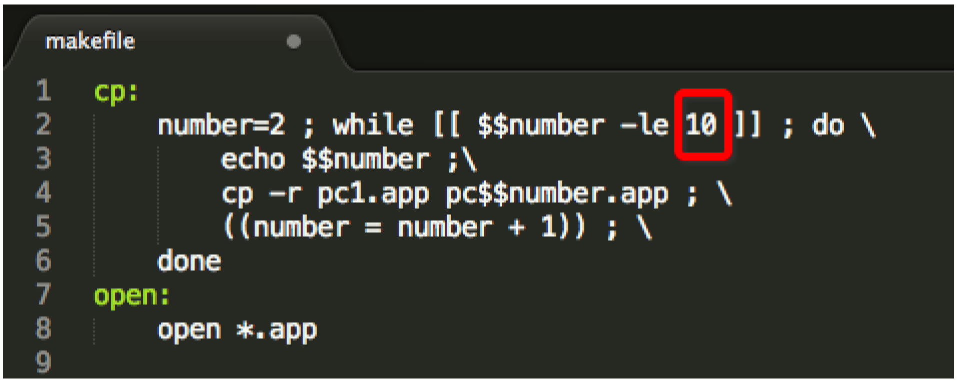
Platform

PC, Mac & Linux Standalone

Target Platform: Mac OS X
Architecture: x86
Development Build ☑
Autoconnect Profiler ☐
Script Debugging ☐

4. Now you will have an application named “pc1.cpp” in your OzUnity-master folder(you may save it anywhere but in the same folder is much better).

5. Open makefile in the ozMachine-master folder



makefile

```
1  cp:
2      number=2 ; while [[ $$number -le 10 ]] ; do \
3          echo $$number ;\
4          cp -r pc1.app pc$$number.app ; \
5          ((number = number + 1)) ; \
6      done
7  open:
8      open *.app
9
```

In line 2 you will find a number 10, which is the number of clients you want to test. You may modify this number.

6. After modify the makefile, type “make cp” in terminal. And now you will have multiple clients named “pc2.app”, “ pc3.app”, and so on.
7. Type “make open” in terminal, all the applications will run.
8. Open terminal and direct to the folder of ozMachine-master and open “ozNodeServer” folder
9. Type in “node socketServe.js” in terminal and the server will be running.
10. You may type in a number of scene to change to that scene. Each clients will be set to the same scene.
11. When the server is running the clients will connect to it and get a unique client ID. The server will find who have connected and print the ids on terminal.
12. You may type “send ” + your message in terminall to send a message to all the clients, like “ send Hello!”. And the clients will reply with its unique Id. The server will collect these replies and write them into a file called “message.txt”, which will be created automatically if not exist.
13. You can open that file and check whether the clients have responsed correctly.
14. You can also turn to a python script there to check, if there are too many clients. In the same folder, simply type “python check.py” + the number of clients in the terminal. For example, “python check.py 10”. If there is no output, you are good. If you see the error log, then you know which one is missing.