MADFINGER Games a. s.

SHADOWGUN - DeadZone
Game Master's Kit

'Howto' guide

Version:	1.0.3
Date:	2015-07-02

Table of contents

Foreword

The project we've prepared for you is a reduced version of the real SHADOWGUN:DeadZone project which was under development for several years. It is more or less a full and functional multiplayer game. It is not a technical example or tutorial but a real and complex project.

You are free to learn from, modify or redistribute it as long as you respect the SG:DeadZone Public License Agreement.

This guide is little bit Windows platform oriented but it should not be difficult to apply it for OSX platform as well.

Prerequisites

You really do not need any special hardware or software to play with our project. All you need is either a Windows or OSX based system with a few gigabytes of RAM and an average GPU. All examples shown here are taken from Windows environment, but you should be able to follow these steps on OSX operating system, if you have developer's license.

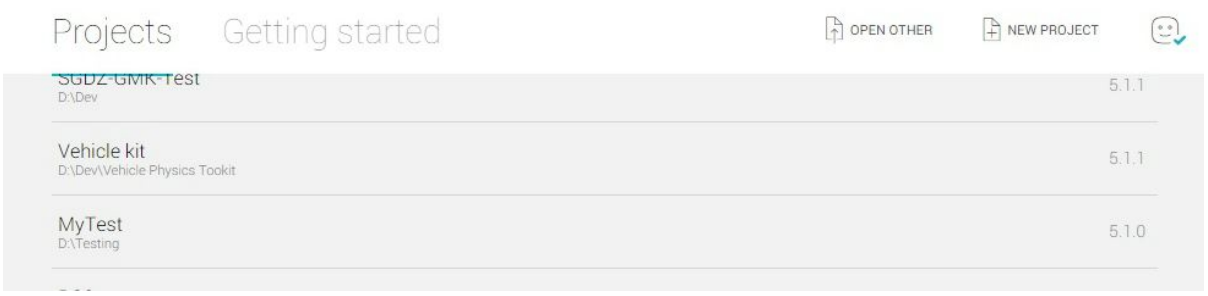
The only software you really need to have installed is Unity 3D development platform. You can find the installer here. The minimum version is 5.1.0 but the project should also work on later versions of Unity. The Professional edition is not really required, you should be able to make all the builds and use all the game features using the Unity Personal Edition.

We are assuming that you have previous experience with Unity and development tools related to target platform, if differs from a host system - for example `adb` for Android build. Additionally, we expect that you are familiar with (but not a professional) game industry and networking problematics.

Get the project and make it ready to use

First of all, make sure there is no Unity Editor instance running at your system. If there are any, close them. Are you ready? Let's start then...

Click the following link and visit the GMK project Unity asset store web page using your favourite browser. Now find the Open in Unity button and click it. It should launch your Unity editor and display the "Open project" dialog. Press NEW PROJECT here:

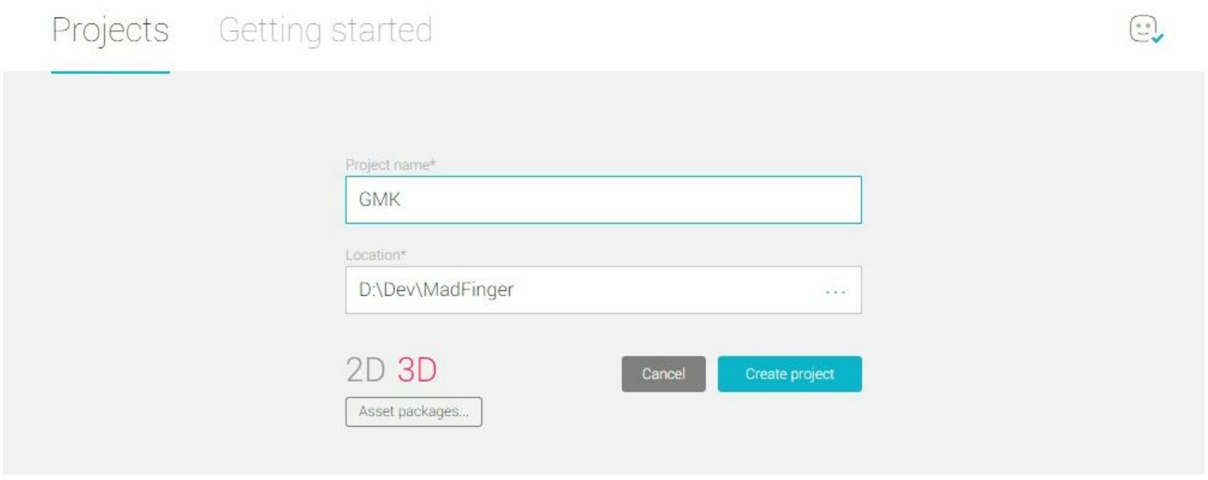


SGDZ-GMK-Test
D:\Dev
5.1.1

Vehicle kit
D:\Dev\Vehicle Physics Tookit
5.1.1

MyTest
D:\Testing
5.1.0

In the “New Project” dialog fill in the project name, choose a project location (folder) and switch it to 3D in a similar way as shown below:



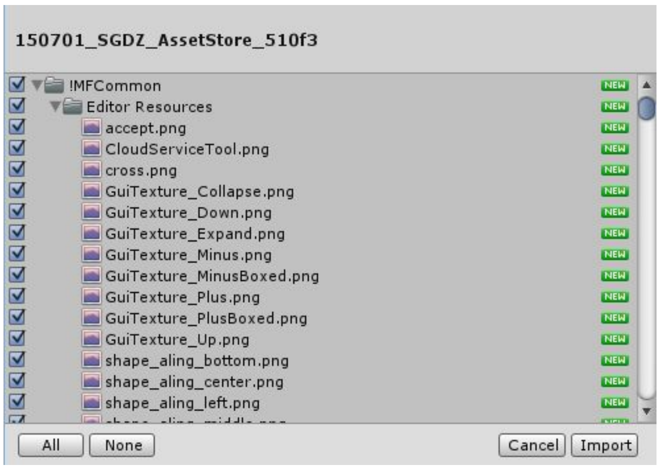
Project name*
GMK

Location*
D:\Dev\MadFinger

And press the Create project button.

The new project should be created and opened automatically. Additionally, there should be the exactly same asset store page opened in the Unity editor built-in asset store browser. The only difference is that the button **Open in Unity** had changed to **Download**. Press it and wait until the GMK complete project package is downloaded and decompressed. That will take couple of minutes depending on your internet connection speed and CPU performance.

After a while the “Import” dialog appear:

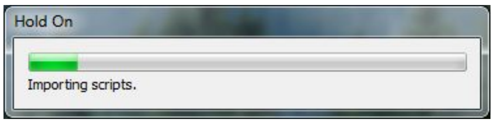


```
150701_SGDZ_AssetStore_510f3

☑ ▼ !MFCommon                          NEW
☑   ▼ Editor Resources                 NEW
☑       accept.png                     NEW
☑       CloudServiceTool.png           NEW
☑       cross.png                      NEW
☑       GuiTexture_Collapse.png        NEW
☑       GuiTexture_Down.png            NEW
☑       GuiTexture_Expand.png          NEW
☑       GuiTexture_Minus.png           NEW
☑       GuiTexture_MinusBoxed.png      NEW
☑       GuiTexture_Plus.png            NEW
☑       GuiTexture_PlusBoxed.png       NEW
☑       GuiTexture_Up.png              NEW
☑       shape_aling_bottom.png         NEW
☑       shape_aling_center.png         NEW
☑       shape_aling_left.png           NEW
```

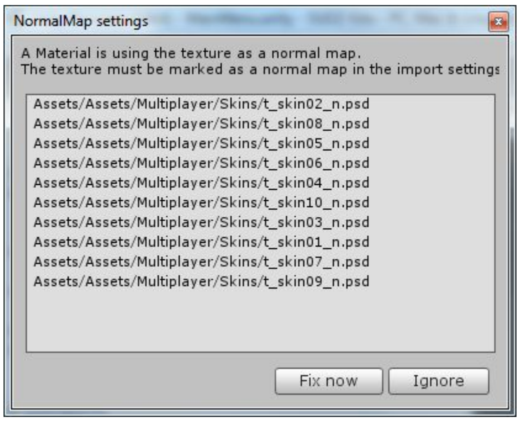
Just for sure press the “All” button and then click Import.

It will take some time to import all of the various resources used in the game. Please be patient - this happens only once.



```
Importing scripts.
```

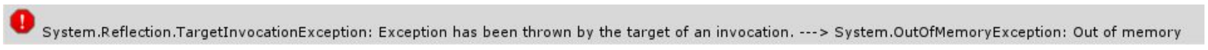
We do have several special shaders/materials in our project. They cause Unity to think that there is a mistake in the settings but they are actually correct. Once you are prompted about the "NormalMap settings", make sure you click “Ignore”:



```
A Material is using the texture as a normal map.
The texture must be marked as a normal map in the import settings
Assets/Assets/Multiplayer/Skins/t_skin02_n.psd
Assets/Assets/Multiplayer/Skins/t_skin08_n.psd
Assets/Assets/Multiplayer/Skins/t_skin05_n.psd
Assets/Assets/Multiplayer/Skins/t_skin06_n.psd
Assets/Assets/Multiplayer/Skins/t_skin04_n.psd
Assets/Assets/Multiplayer/Skins/t_skin10_n.psd
Assets/Assets/Multiplayer/Skins/t_skin03_n.psd
Assets/Assets/Multiplayer/Skins/t_skin01_n.psd
Assets/Assets/Multiplayer/Skins/t_skin07_n.psd
Assets/Assets/Multiplayer/Skins/t_skin09_n.psd
```

A couple of warnings might appear during the import process, but they should not stop you from your being able to do your work. If you haven’t received any compilation errors (we can’t really guarantee compatibility with future versions of Unity), you are done. If you find a serious issue, you will have to fix it or check if there is an updated version of the project.

There is one known error triggered during the import process. It is probably related to an Unity internal pipeline (dll patching) and we are unable to fix. Surprisingly, this error has no visible consequences and you can safely ignore it.



```
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. ---> System.OutOfMemoryException: Out of memory
```

There might be more warnings or errors on OSX, but all should run flawlessly as well.

We’ve identified a weird bug in the Unity editor resulting in faulty builds (pink screen because of missing/broken pixel shaders) if you continue to work at the Unity instance you’ve used for the project import. Please restart Unity editor and reopen the project to workaround it.

Now you are ready for your real work!

Make your build

There are several ways how to make a client or a server build.

In-Editor build

This type of build is best for development purposes because it is fast and allows you to inspect all the game objects and their components at runtime.

Also, this method represents the simplest way how to make a build and launch it. All you need to do is open the project, compile your modifications and load a proper scene. For the client build you have to load the MainMenu.unity scene, for the server build load the DedicatedServer.unity scene. Afterwards, just press the play button.

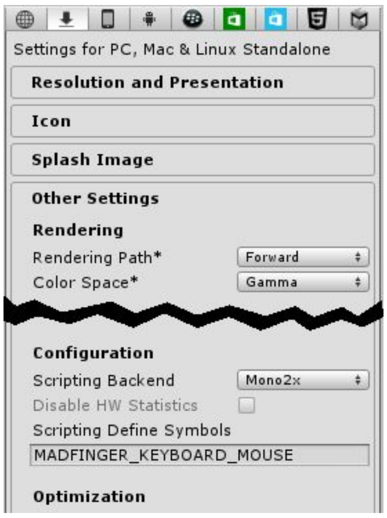
This method should work without any visible limitations on all development platforms (Windows, OS X), in both the Unity Personal and Professional licenses.

Manual build

This build is useful for the testing or development purposes, especially when you need to launch a server and one or more clients on a single computer or on several computers connected into the same LAN.

Symbol defines

Before you start building, you might want to review already defined scripting symbols, but if you leave all as is, you are safe, unless building for mobile platform.
For changes, press Player Settings in a build dialog and set symbol(s) in a rightmost Unity pane as seen on a picture.



MADFINGER_KEYBOARD_MOUSE

Here are the most common client ones:
DEADZONE_CLIENT is required for all mobile (client) builds.
MADFINGER_KEYBOARD_MOUSE switch is required for all client standalone (desktop) builds. It enables the keyboard and mouse input devices.

Server defines:
DEADZONE_DEDICATEDSERVER is useful for server builds to improve the server performance - might be used in a Windows build as well.

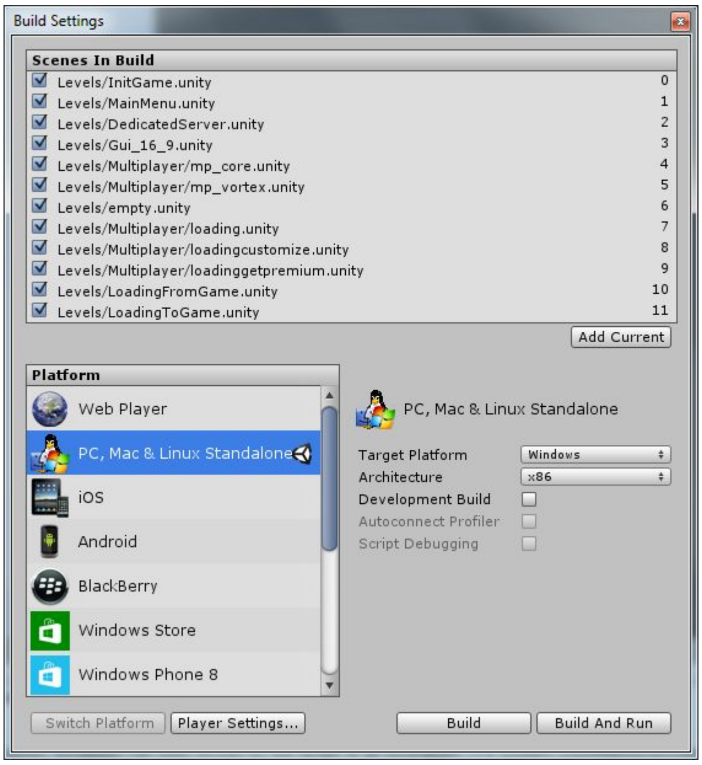
Please note, that Linux build is not tested at all. For any other platforms, we suggest having a dedicated project copy for each platform because the platform switching is a painfully slow process. The target platform specific pipelines are beyond the scope of this document. Check the Unity platform specific documentation for more details.

To make the build you have to just to load the project and open the Build Settings dialog (File / Build Settings). Then you have to choose the list of scenes you would like to have in your build.

Please remember that Unity always makes the scene with the index 0 the default scene, i.e. it will be loaded first once you launch the build.

Windows client build

To make a client build, you have to select all the scenes in the list. InitGame.unity must have the index 0 assigned as shown below:



Build Settings

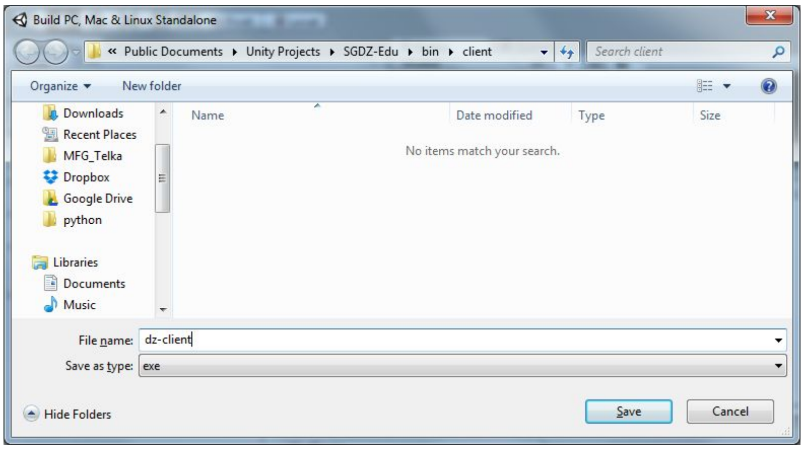
Scenes In Build

✓ Levels/InitGame.unity	0
✓ Levels/MainMenu.unity	1
✓ Levels/DedicatedServer.unity	2
✓ Levels/Gui_16_9.unity	3
✓ Levels/Multiplayer/mp_core.unity	4
✓ Levels/Multiplayer/mp_vortex.unity	5
✓ Levels/empty.unity	6
✓ Levels/Multiplayer/loading.unity	7
✓ Levels/Multiplayer/loadingcustomize.unity	8
✓ Levels/Multiplayer/loadinggetpremium.unity	9
✓ Levels/LoadingFromGame.unity	10
✓ Levels/LoadingToGame.unity	11

Platform: PC, Mac & Linux Standalone

Target Platform: Windows
Architecture: x86
Development Build: ☐
Autoconnect Profiler: ☐
Script Debugging: ☐

Then press Build button and choose directory, where to store client files. We recommend to create special directory for client and another one for server. Then name client binary (`dz-client` is OK) and point your dialog to client directory just created:



Build PC, Mac & Linux Standalone

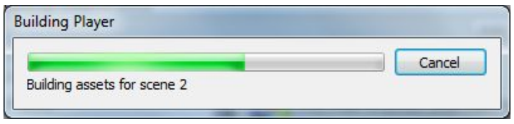
« Public Documents ▸ Unity Projects ▸ SGDZ-Edu ▸ bin ▸ client

No items match your search.

File name: dz-client
Save as type: exe

Press Save.

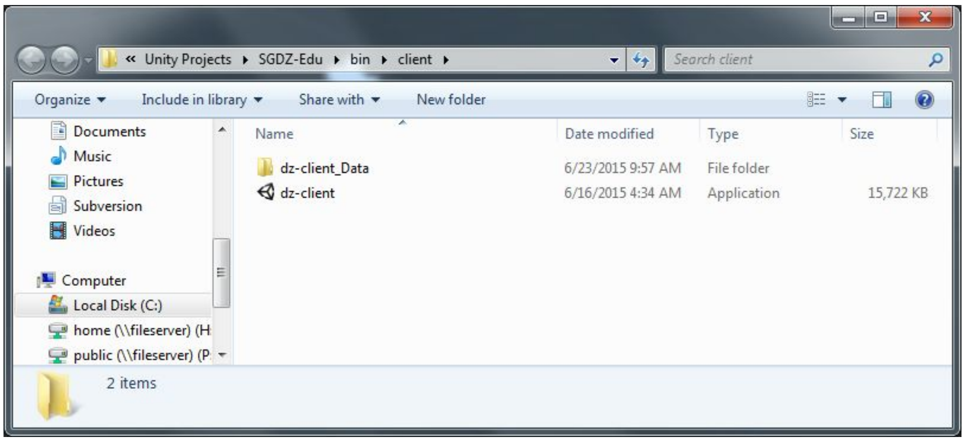
Window appears:



Building Player

Building assets for scene 2

Once done, window with client executable appears:



« Unity Projects ▸ SGDZ-Edu ▸ bin ▸ client ▸

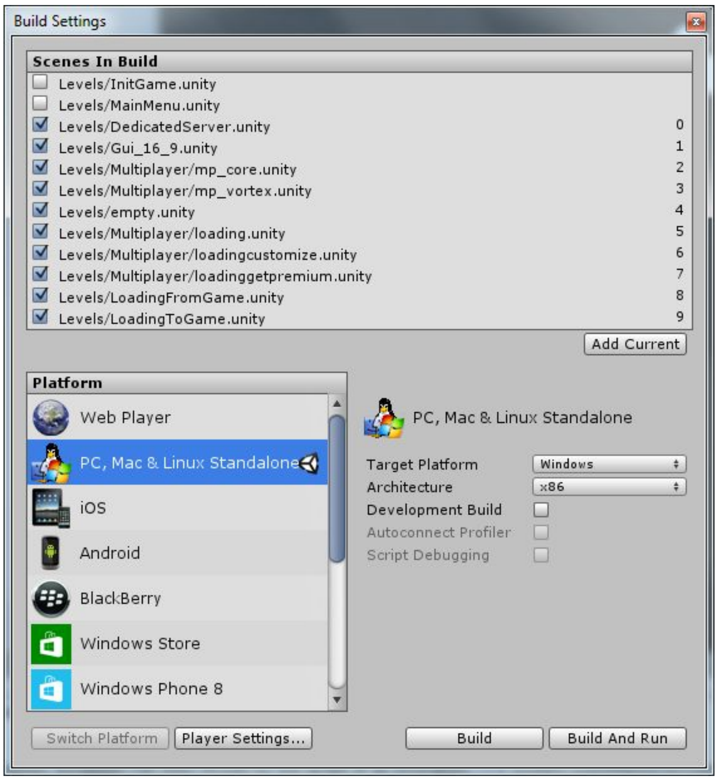
Name	Date modified	Type	Size
dz-client_Data	6/23/2015 9:57 AM	File folder	
dz-client	6/16/2015 4:34 AM	Application	15,722 KB

2 items

Leave it open for now - will be useful when starting a client later.

Windows server build

To make a server build, you have to uncheck the InitGame.unity and MainMenu.client. DedicatedServer.unity must have the index 0 assigned:



Build Settings

Scenes In Build

☐	Levels/InitGame.unity	
☐	Levels/MainMenu.unity	
☑	Levels/DedicatedServer.unity	0
☑	Levels/Gui_16_9.unity	1
☑	Levels/Multiplayer/mp_core.unity	2
☑	Levels/Multiplayer/mp_vortex.unity	3
☑	Levels/empty.unity	4
☑	Levels/Multiplayer/loading.unity	5
☑	Levels/Multiplayer/loadingcustomize.unity	6
☑	Levels/Multiplayer/loadinggetpremium.unity	7
☑	Levels/LoadingFromGame.unity	8
☑	Levels/LoadingToGame.unity	9

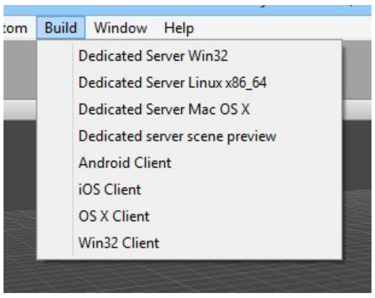
Platform: PC, Mac & Linux Standalone

Target Platform	Windows
Architecture	x86
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐

Continue with the build the same way as you did for client - remember to create dedicated directory for server and choose suitable executable name (`dz-server` is a good option).

Automated/script build

Automated or script based builds are best for QA testing but you can use them for the development purposes as well (depends on the build and its settings).



The benefit is that everything happens automatically, minimizing the chance of human error. Additionally, some of the builds take extra pre or post process actions which is not possible in manual build.

You can find them in the Unity Editor at the Build menu. Most of the items in the menu are quite self-explanatory and do not require any additional descriptions. There are few special builds, though.

The build scripts store their builds into $ProjectDir/../_Packages folder.

And now something for advanced game developers: There is a special set of dedicated server builds which do support data stripping. The data stripping is a very effective way how to reduce server's memory footprint and optimize the performance at the same time. Data stripping builds can be enabled in the BuildPlayer.cs source file. Check it for more details.

Establish a game session

SG:DeadZone is a server-client online multiplayer game. It does not support a listen server (server running in the same process together with a client), it strictly splits into clients and dedicated servers.

Contrary to the full version of the game it does not support matchmaking. Thus, you need to have a server running. The server needs to be network accessible to all potential clients. It has to run either on a server machine with a public IP address or in a LAN together with other clients.

Server

The server can be launched either in-editor or as a standalone build on a desktop platform. In theory nothing can stop you from running the server on a mobile device but in reality it is not the best option because of the performance and the lack of command line.

The server listens on a UDP port. The server can be launched in two ways

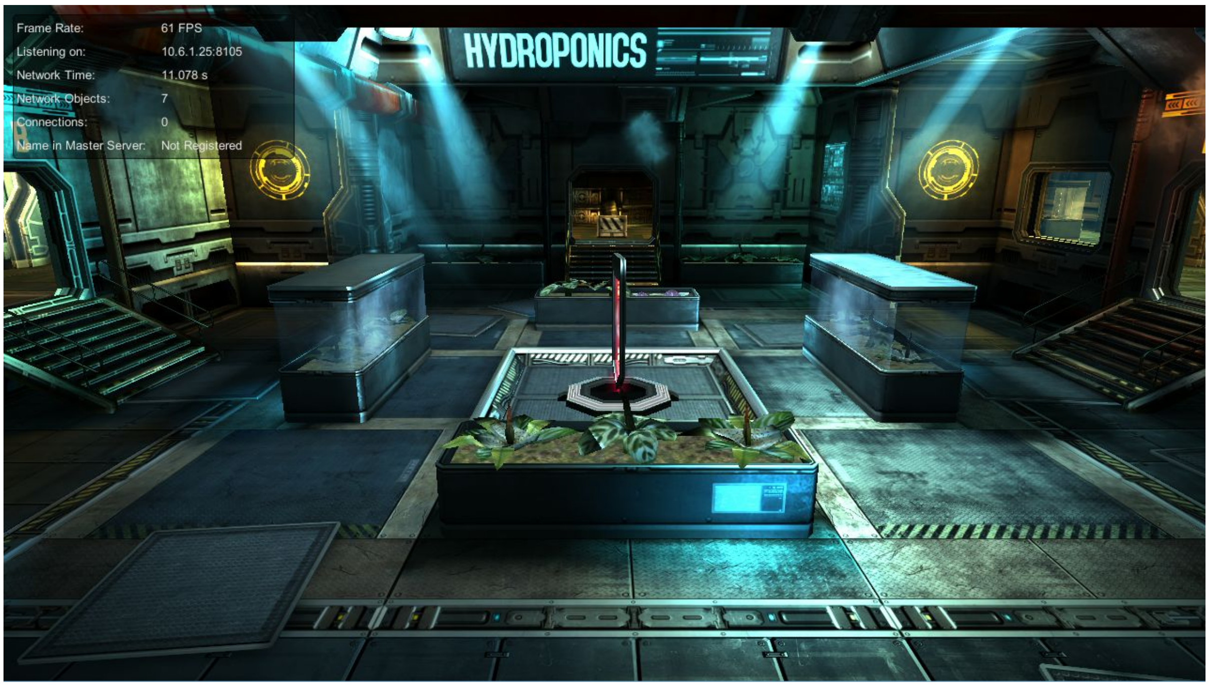
- With the server port automatically chosen from a pre-defined port interval (8101-8105). First server on a machine binds to 8101, second to 8102 etc.
- With a manually assigned port number

Example command line of automatically assigned port (Windows):
```
dz-server.exe
```

Example command line of manually assigned port (Windows):
```
dz-server.exe -serverport=8101
```

See the command line arguments chapter for more info. Additionally, do not forget to configure your firewall properly to make sure your server is accessible to other clients.

After a server launch, you can see a window with server status information:



```
Frame Rate:              61 FPS
Listening on:            10.6.1.25:8105
Network Time:            11.078 s
Network Objects:         7
Connections:             0
Name in Master Server:   Not Registered
```

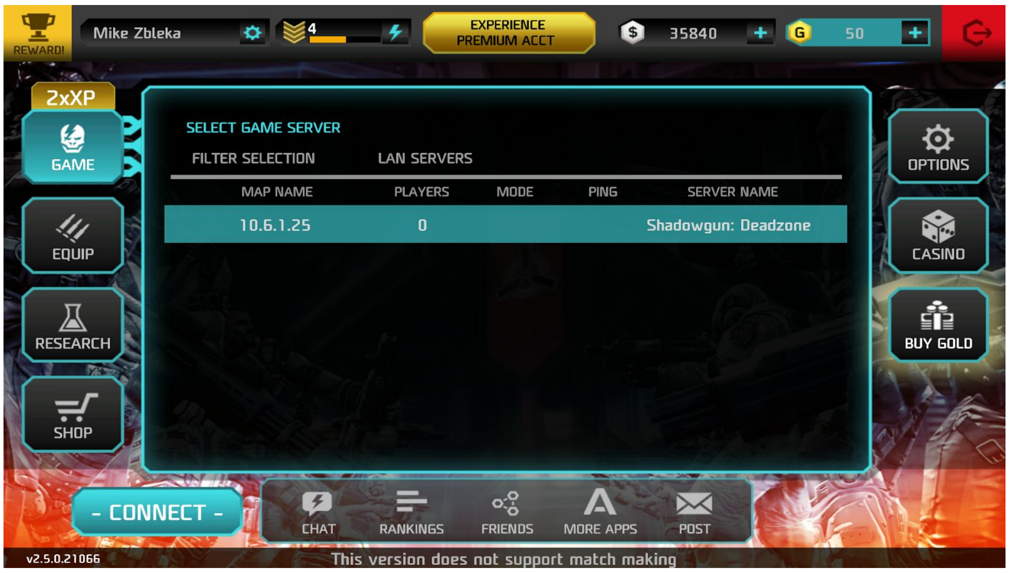
Client

Client can run on each supported platform including the Unity editor. There are currently two ways how to connect a client to the server:

Server list dialog

This option is supported by all platforms.

Simply, launch your client and sign in using your username and password. In the main menu, press the “Game” button. Unlike the official game, it opens the server list dialog:



SELECT GAME SERVER

FILTER SELECTION   LAN SERVERS

MAP NAME	PLAYERS	MODE	PING	SERVER NAME
10.6.1.25	0			Shadowgun: Deadzone

This version does not support match making

The server list in this project version is not powered by a dedicated master server running somewhere on the internet but it is implemented using UDP packet broadcasting. Thus, you can't use it to join an unknown game server running somewhere on the internet. It is moreless connected with localhost or LAN. To use it, you must meet the following conditions:

- packet broadcasting must be enabled in your network
- game server must listen on an UDP port within the interval 8101-8105

Command line

This option is supported by all standalone builds (Windows, OSX, Linux). It is an effective way how to get quickly into a game. If you combine this option with the "Login me automatically" feature at the login screen you can join into your game without a single click!

You can join a game server running on a localhost and known ip:port

```
dz-client.exe -join=127.0.0.1:8101
```

Also, you can join a game server running somewhere on the internet. In such case you need to know its ip and port.

```
dz-client.exe -join=123.124.125.126:8101
```

Command line arguments

All the following command line arguments apply to a standalone client or server builds only. They can't be applied to in-editor or mobile builds.

Server

`-gametype=type`
There are two game types supported: `zc` (Zone Control) and `dm` (DeathMatch)
In both cases, the server takes the first map of the given type from a list and loads it.
Example: `-gametype=zc`

`-serverport=port_number`
With this argument, you can explicitly define the server port the game listens on. If the argument is omitted, the server port is chosen from the default port range as before (8101 - 8105).

`-batchmode`
This is one of the built-in arguments but a very useful one. It runs the game server in a so called headless version. This does mean that the game runs without any visible output (window) and input. This is very useful especially on server class machines. On Windows, it is a good practice to combine it with the `-nographics` argument. For more info check Unity Command-line reference.

Client

`-join=ip_address:port`
When specified, the client tries to connect to a given server automatically after the user signs in. To use the maximum potential/benefit of the feature it is good practice to combine it with the “Logon automatically” feature (check-box) at the sign-in screen.

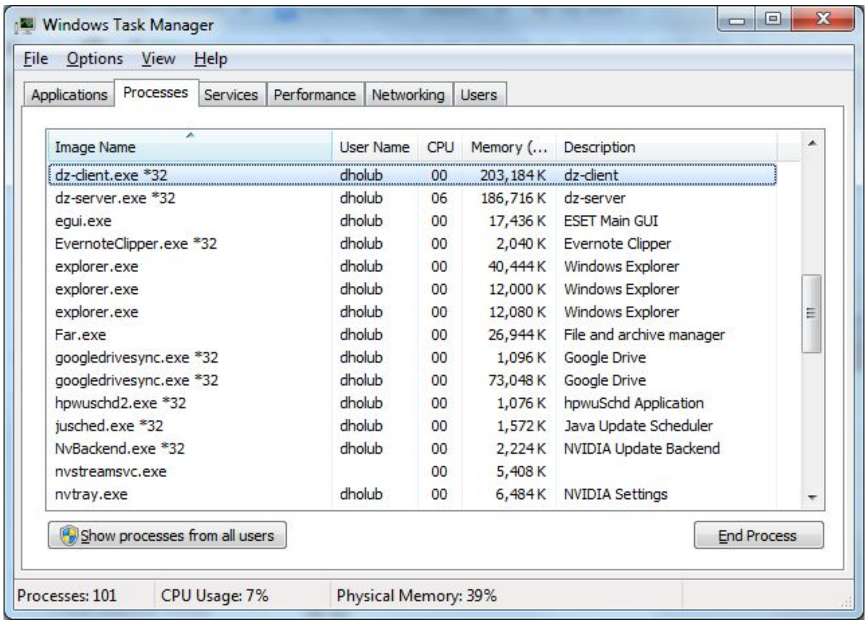
Support

Please note that we do not provide direct support for this project. However, you might find a help or an answer to your questions at our game forum.

If you want to actively participate on maintenance or development with access to full game version, please visit the community forum for more details.

FAQ

Q: After I start server, window disappears and nothing happens. What’s wrong?
A: You probably started your server using the `-batchmode` command line argument. If so, your server is running in background. Try to find it in a list of running processes:



Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Show processes from all users
End Process

Image Name	User Name	CPU	Memory (...	Description
dz-client.exe *32	dholub	00	203,184 K	dz-client
dz-server.exe *32	dholub	06	186,716 K	dz-server
egui.exe	dholub	00	17,436 K	ESET Main GUI
EvernoteClipper.exe *32	dholub	00	2,040 K	Evernote Clipper
explorer.exe	dholub	00	40,444 K	Windows Explorer
explorer.exe	dholub	00	12,000 K	Windows Explorer
explorer.exe	dholub	00	12,080 K	Windows Explorer
Far.exe	dholub	00	26,944 K	File and archive manager
googledrivesync.exe *32	dholub	00	1,096 K	Google Drive
googledrivesync.exe *32	dholub	00	73,048 K	Google Drive
hpwuschd2.exe *32	dholub	00	1,076 K	hpwuSchd Application
jusched.exe *32	dholub	00	1,572 K	Java Update Scheduler
NvBackend.exe *32	dholub	00	2,224 K	NVIDIA Update Backend
nvstreamsvc.exe		00	5,408 K	
nvtray.exe	dholub	00	6,484 K	NVIDIA Settings

Processes: 101 CPU Usage: 7% Physical Memory: 39%

Should you need kill the server process, this is a good place to do so.

Q: I can not see my server in the server list dialog. Why?
A: There might be more reasons:

- You are not running any server. Run a server as described above.
- You are running a server with port out of range of 8101 - 8105. See above how to configure your server port properly.
- Your LAN configuration might not properly support UDP broadcasting. If you combine WiFi (Wireless) and wired LAN network (for example: server connected via LAN, client connected via WiFi), make sure that the WiFi <-> LAN broadcast forwarding is enabled. Check your router configuration carefully.

Q: Both automatic and manual server and client builds renders in magenta color. What went wrong?
A: Probably you did not restart Unity editor after the project import. Please restart it and re-make your builds.

Known issues

Exploding barrels

Exploding barrels sometimes fail to explode. It is an annoying but rare bug. It looks like a server side problem. The problem was never observed in the official build, though.

Server enumeration

Only servers listening to a port in the range of 8101-8105 are displayed in the server list screen. The "bug" is caused by a bad design decision in the underlying technology. Of course, it is possible to make the interval wider but it could have very negative impact to the network load (UDP broadcast flooding). Do not increase the interval unless you have a really good reason for that.

Document history

Date	Version	Author	Status	Note
2015-06-22	0.1	Mike Zbleka	Draft	First version
2015-06-23	0.2	Dushino	Reviewed	More screenshots, a bit more descriptive for beginners
2015-06-23	1.0	MFG	Approved	
2015-06-23	1.0.1	Mike Zbleka	Update	Correction of technical information
2015-06-24	1.0.2	External	Language review	
2015-07-02	1.0.3	Mike Zbleka	Update	Known issues, screenshots