Full Game Kit - Operation Desert Road

Version 1.0.0

Thanks for buying Full Game Kit - Operation Desert Road!

Documentation
Documentation can be found here:
http://xform.nl/projects/AssetStore/OperationDesertRoad/OperationDesertRoadDocumentation.pdf
This file is also located in the Project Assets folder of the kit, right beside the ReadMe file!

Tutorial Videos
These can be found here:
https://www.youtube.com/user/xformgames
Also be sure to check the Tutorials chapter of this document.

Support
Send your (support) questions to support@xformgames.com
There may also be a thread on the Unity forum, although this is not an official page.

License notes
You may use all source codes and assets in your own projects and games, also if they are for commercial use. However, unchanged copies or clones of Operation Desert Road are not allowed. We are a very flexible when it comes to commercial use, but a quick unchanged build is not ok. Change some assets, like the name, textures, sound (effects), levels, players, etc. and you will be fine. You may NOT distribute the source code or assets directly, for instance in the Asset Store. If you've obtained this software from any other source than the Asset Store, your license is effectively invalid, and Xform cannot provide support for it.

Index

1 General Information

1. Installation

Download and import the package from the Unity Asset Store into a preferably new Unity Project.
Alternatively you can try copying the already unpacked contents into a Unity Project.

2. Play the game in the editor

Open the Game scene in the Scenes folder.
Press Play.

3. Create a build

Choose File > Build Settings
Choose your platform.
Press Build.

Supported and tested platforms:

- Standalone (Win, Mac*, Linux*)
- WebGL
- iOS
- Android

* not tested

4. Controls

Keyboard

Control:	Action:
Drag mouse	Move
WASD/Arrowkeys	Move
LMB/Z/Space	Fire
RMB/X	Use special attack
Esc/P	Pause

Touch

Control:	Action:
Drag left / right	Move
Tap screen	Fire
Tap special attack button	Use special attack
Tap pause button	Pause game

Cheats*

Key:	Cheat:
K	Mission Complete
L	Game Over
C	Give 100 credits in shop
I	Toggle interface
O	Toggle cursor visibility
U	Toggle cursor lock

**Not available on touch devices. Cheats need to be enabled in the GameScript on the _Game object.*

5. Version history

For the version history of the project, please refer to the ReadMe.txt in the Project Assets folder. This article refers to the version history of this document.

1.0.0 Release
Added this version history list
Added Tutorials section
Removed Mobile build section
Added tutorial video references in FAQ
Added question 16 to FAQ

2 The Game

1. Introduction

– A brief history of the game

Operation Desert Road is inspired by casual mobile games such as Crossy Road. We wanted to do something with a little more action, and before we knew it we had guns, explosions, collapsing buildings and airstrikes.

We added in lots of fun vehicles to try out and made a procedural mission generator. To top things of we also made a system that bakes in vertex colors to make it all look nice and shiny.

Then we thought it would make a great second Full Game Kit project. Due to the game's smaller scope, we could simplify the structure we had for our previous kit. Making it more accessible to beginning Unity users!

2. Instructions

– The goal of the game and how the controls work

Use your tank to complete missions to free the world from evil! Shoot everything that stands in your way! Avoid obstacles like red barrels and mines. Watch out for enemy vehicles. Your tank cannot swim. And it cannot drive through buildings.

Swipe to move your tank and tap or fire to shoot. Use the onscreen button or the right mouse button to use a special weapon if you've picked up one.

Clear procedurally generated missions by reaching the finish line. Complete levels by clearing missions. The game has an infinite amount of levels.

3 Example project

1. General

– How to use this project and how to learn about it.

We've decided to sell this game on the asset store as an example project. To be honest, the main reason for this was that we thought we could make some money for all the time we had invested in creating it. We're hoping that there is a demand for more polished and complete example projects. Most of the projects out there are pretty basic and miss features to really publish them 'as is'.

Providing such a 'complete' project has the downside that you cannot make every aspect of it as modular and foolproof as some other simpler kits are. But for this project we made an effort to keep things as simple as possible. Most changes you'd want to make to the game can be done without any scripting.

Unfortunately, it's not do-able for us to describe every script in the project in a lot of detail here in the documentation. So we've created a brief (but necessary) overview of how the game does what it does.

Our first tip is to play the game for a bit. Stop the game and see what's going on in the hierarchy manager. Change something in the code and see where you've broken it. Start debugging. Most of the scripts have a summary what it does and a lot of coder comments.

Also, we're going to create a tutorial video to create your own game with this kit! So keep your eyes peeled on our Youtube channel.

And of course don't forget to read the rest of this document.

2. Flow

– Describing the coding flow

The game consists of one scene, the Game scene, which only contains three objects.

_Game: This gameobject has the GameScript component in which you can change important game variables. This scripts works closely with the static scripts Game.cs which actually starts up the game. The rest of all game related object are instantiated dynamically and only visible during runtime.

_Interface: This gameobject has the InterfaceScript component (amongst some other Unity required components) and it also contains most interface elements. So. these are already present in the scene for easy editing, and not instantiated dynamically.

_Eventsystem: This gameobject is required by Unity if you use a GUI and you want to catch input. Not important. Forget this object is even there!

If you want to follow the game flow from screen to screen and into the game, it's best to inspect the InterfaceScript. This activates all panels and takes care of the button controlled and automatic navigation between screens.

The Game.cs script loads up the level / missions, and sets up all the required dynamic object you see appearing when starting up the game.

3. Data Management

– How important variables are set and used.

The engine and game variables are defined as public variables that can be found on the **GameScript** on the **_Game** gameobject in the **Game scene**. There are a couple of categories with distinct characteristics. All of which can and are accessed by scripts. All of which are saved and persistent. All of which refer to classes in scripts, so for more information refer to the scripts themselves. These variables (except for Settings and User categories) are saved inside the Game scene file.

Details: These are some global variables, especially the Helpers are handy for debugging.

Platform: Used in script to set platform specific stuff.

State: Current state of the game.

Branding: These are variables with which you can create a particular brand (or version) of your game. For example, if you deploy the game on various game portals, you want to have different logos implemented. This is not interesting for all users. There's always a default branding active.

Settings: User preferences are kept here (i.e. quality, mute, tutorials, cheats etc). These are saved locally on a players device. These settings are NOT saved in editor.

User: Use progress variables are kept here (i.e. credits, current mission, current vehicles, unlocked vehicles, achievements etc.). These are also saved locally on a players device. UserData is NOT saved in editor.

Other **game related variables**: A lot of gameplay related variables such as player **health** and **speed** are defined on the Player prefabs, the Enemy prefabs, Projectile prefabs, PickUp prefabs etc.. Change those there, these are automatically 'saved'. Also see the scripts mentioned in chapter 5 of this document.

Important note: If you nest prefabs inside other prefabs, or if you place them manually in a scene, the prefab variable connection is lost. So if you change anything in the prefab after that, it will not be processed throughout the before-used prefabs.

Localization text files such as **\Game\Assets\Project Assets\Resources\English (US).txt** contains only textual editorial information, which is read only. Of course, only the currently selected language and corresponding text file can be accessed. The current localization file that is used can be set runtime using XLocalization.language or on **_Game** object **Settings -> Game Specifics -> Language**

Note: You can easily add your own entries. Make sure the key is unique. And that you add it to other languages (if you have those) as well.

You can also easily add new languages by coping English (US) and rename that to e.g. Dutch. New languages will automatically show up under ***_Game*** *object* ***Settings -> Game Specifics -> Language*** *popup.*

4. Code

– Describing some of the game's important systems.

As you might have seen, the scripts are divided into two main categories and folders:

Engine: All scripts that we want to share between the different games we produce.
Game: Specific code for Operation Desert Road.

And then there's a third called **XGUI** inside the Game/2D folder: Our quick solution for handling stuff previously done by third party software such as tweening of colors, scale, position of 2d objects and handling the localization of text elements.

Describing all scripts in this document is not possible, so please refer to the comments in the scripts themselves for more information.

5. Assets

– How the resources are structured, and how they are used.

Resources

Tip: For a quick overview of some of the assets please refer to the Overview scenes located in the **\Game\Assets\Project Assets\Scenes** folder**.**

The game assets source files are located under **\Game\Assets\Project Assets\Resources** in the various folders. These files are all necessary for the game to run, so remove or rename them at your own peril. You can of course adjust them or add files. And if you are up to it, maybe make changes in code to change which are used.

Note: We've changed the structure of our Project Assets a bit compared to the previous Full Game Kit. The materials and textures are now located closer to the 3d files themselves (this change had to do with import workflow). There are still separate Materials and Textures folders for assets not directly related to a particular 3d object.

Prefabs

Almost all of the assets are instantiated during runtime, so you cannot find them already present in the Game scene. Look for them in the Project Assets folder, primarily under **\Game\Assets\Project Assets\Resources\Prefabs .**

Here's a quick overview of the assets inside the Prefabs folder content:

- **2D**: Contains buttons and 2d element prefabs. Makes it a little bit easier to manage all the interface elements. All sprites can be found in **\Game\Assets\Project Assets\Resources\Sprites .**
- **Blocks**: A chunk of level, also used by the campaign editor. This is just the visual part of the environment, it contains no gameplay objects.
- **Buildings**: These can be used to fill your mission, also used by the campaign editor. AS you can see, the buildings have a **DestructibleBuildingScript** attached which contain variables you can tweak!

- **Cameras**: The game camera is instantiated dynamically, and the script controlling it as well. So, some changes to the camera can only be made by editing this prefab. to change the camera behavior, look for the **CameraScript** (which could use some commenting!).
- **Civilians**: These are spawned when you destroy a building. They run around in panic. You can edit their behavior in the **CivilianScript** that is attached to them. Note: they have an animation attached which is automatically played.
- **Destructibles**: Not really a vehicle and not a soldier. It has a **DestructibleScript** attached in which you can tweak variables.
- **Effects**: This folder contains all effects that are dynamically instantiated during the game. Drag some in the scene to see what they do. Note: Some particle effects are only visible when you move the gameobject that it is attached to!
- **Enemies**: The current enemies all have their own Enemy script, because their behavior is pretty different. You can tweak the settings on the scripts.
- **Foliage**: Not super interesting. Filler objects for your level, also have the **DestructibleScript** attached.
- **Helpers**: A variety of objects used during the game. Finish_Prefab is used for buildings missions.
- **Misc**: More filler objects for the missions.
- **PickUps**: These have a PickUpScript attached, which you can tweak. Note: The pickups handle themselves through this script, there is no manager.
- **Players**: All prefabs in this folder are automatically considered player vehicles. They all have a PlayerScript attached, of which you can tweak the variables.
- **Projectiles**: These are the prefabs used as bullets for the player and the enemy. You can tweak the ProjectileScript that is attached to each prefab. Note: Don't make the projectiles too slow because the speed is relative to the world, not the player ;).
- **Roads**: The prefabs containing only the floor objects (which require a collider!).
- **Soldiers**: These enemies all contain their own script for their particular behavior. Please know that the 'Soldier_Prefab' is used as a friendly for a Special Attack of the player!

Adding assets

Adding stuff to your game is now easier because the gameplay object are more self contained. It's all prefabs with scripts. So just look at a particular prefab type example and just create something similar.

For example, you can easily add your own player vehicle just by attaching a PlayerScript component to a 3d object (don't forget to click the BIG BUTTON on the script). Then, create a player prefab out of it by dragging it into the Players folder.

Please also check the Tutorials section!

4. Tutorials

1. Overview

We've added a series of tutorials which you can find through our Youtube channel. In the first of this series, we're discussing the package in general terms. Most of the information you could also have found in this document.

Link: https://www.youtube.com/watch?v=ZwtbC-urNNs

2. 3D Art

This video shows you how to create your own voxel art and import it into the game! Very handy if you are planning to create something original. Please know that you can of course also use this kit to create a game with regular 3d objects. For example, with 3d objects found in the asset store.

Link: https://www.youtube.com/watch?v=IMU9ucAN_pk

3. Interface and menu

Here, we'll be showing you how to add your own 2d elements to the project. Also, we'll explain how to add buttons and how the buttons work. And we'll be changing the appearance of screens by adding and removing elements, and changing colors and fonts.

Link: https://www.youtube.com/watch?v=ZgPn4kqMNGc

4. Level design

This video covers the preparing of prefabs, creating chunks of environment, placing the assets onto the levels and creating missions out of it. The package comes with its own basic level editor called the 'Campaign Editor'. This video also shows you how to do some camera adjustments.

Link: https://www.youtube.com/watch?v=U9-kz-Ngny0

5. FAQ

1. What is this?
It's a full game AND a Unity example project. Play the game and have fun! Explore the example project and get inspired. Use the scripts and assets in this project for your own work as you see fit.

2. Why isn't feature x included?
The project started out as just a game project, so it only has the functionality we thought was necessary for it. The game already has quite a bit of functionality, maybe even more than it should have for simplicity's sake. Nevertheless, if you think something essential is missing, don't hesitate to let us know, and also why you think it should be included.

3. Can I skip directly into the game?
Go to the Game scene, click on the _Game gameobject and open the GameScript. Here, you need to change a couple of variables...

- Details/Helpers/ShowPreloader: False
- Details/Helpers/ShowMenu: False
- Details/Helpers/ShowGetReady: False
- Branding/ShowSplash: False
- Settings/GameSpecifics/Tutorials: False

Don't forget to change them back before you make your build!

Note: This is also explained in the tutorial videos.

4. What is the easiest way to upgrade to a newer version of the kit?
There is no real easy way unfortunately. Try to keep your own programming outside the Full Game Kit - Operation Desert Road scripts. Make a backup of your work. Import the upgraded package into a new project. Close Unity and Monodevelop. Copy the freshly imported files over the ones already there, making sure you're not overwriting your own work.
If you are already using 1.0.x version of the package, we really recommend importing your own scripts and assets into a fresh project that has the new package imported.

5. Does the game work on platform x?
There's only one way to find out: go for it. Please know that it does not help to bully us into adding features or platform support by giving us bad reviews or ratings. Obvious really.

6. Can I sell games made with this kit or based on this project?
Yes. You can even sell it if you've just made a mod or a cool adjustment to the original game. Don't forget to buy us a beer if you become a billionaire.
> Please do not blame us if you get accused of theft or asset flipping by the community.
> Please do not use the Operation Desert Road' or Xform related names, logos, videos or screenshots in your publications. Mainly because they may imply we support or endorse your game more than someone else's.

7. I found a bug. Can I report it?
Yes. We hope you have time to investigate it first yourself thought, and maybe suggest a solution if you have one. Send it to the support mail address. Do not post it in the review section of the asset store - because communication is easier/faster per email at the forum.

8. I want to change game parameters and settings. Where should I start?
Read the bit about Data Management in this document. Or watch the tutorial videos!

9. I want to add some of my own assets. Where should I start?
Read the bit about Assets!

Note: This is also explained in the tutorial videos.

10. When is the next update planned?
We're still looking at optimizations, improvements and bug fixes on the current build. If we make a critical change, we will update the package as we can get it through the asset store review process!

11. Why is this project so cheap?
Mostly because the systems and scripts are not as modular and easily portable as you sometimes see in other Asset Store packages. This may change in the future.

12. How can I use the cheats?
Go to the Game scene, click on the _Game gameobject and look for Settings/GameSpecifics/Cheats in the GameScript and set it to true.

Note: This is also explained in the tutorial videos.

13. Is the package compatible with Unity 4.x ?
No - not initially. The package contains Analytics and Ads which are only available in 5. We did not use the external package but the internal services. However, if you remove the scripts or the references chances are it might just work.

14. Is the package compatible with Android or iOS?
Yes. At this moment.

15. The game doesn't run or runs poorly on my mobile device x. Will you fix this?
We're only a small company and unfortunately we cannot test on a lot of devices. Although we cannot give guarantees on all devices in the world, we do appreciate it if you provide us with feedback on the support mail address. As long as it is not abusive. Then maybe we can see if we can fix it!

16. Wow, getting something on the Google Playstore or the Appstore looks like a lot of work. Can you come over and do it for me?
Unfortunately, no. Creating and testing mobile builds is indeed pretty tricky. Testing them and deploying them in the mobile stores can also be pretty complex if you have not done it before. We suggest you look for help and information online.