UDC 004.223

DESIGNING THE MOBILE GAME APPLICATION "CUBE" UNDER THE ANDROID OPERATION SYSTEM

IVAN LARIONOV, YURY KRAVCHENKO
Polotsk State University, Belarus

This article describes how to store data for a mobile gaming application running the Android operating system, as well as how to protect them.

Introduction. Unity3D is a cross-platform game engine produced by Unity Technologies. Unity allows you to create applications that run on more than 20 different operating systems. The engine supports many popular formats of models, sounds, materials and textures.

The project in Unity divided into scenes (levels) - separate files containing their game worlds with their own set of objects, scenarios, and settings. Scenes can contain both, objects (models) and empty game objects - objects that do not have a model. Objects, in their turn, contain a set of components which scripts interact with.

All these data must be protected from copying, illegal use, pirated distribution, professional analysis and hacking.

Main section. When compiling a project Unity creates an executable game file, and in a separate folder - game data (including all game levels and dynamic link libraries).

Data storage methods. Information objects, the relationships between them, as well as the ways of their influence on the system should be represented in the final software product using a data model. It means that the data model is a set of data structures and operations for their processing.

The developed application will include a large amount of information prepared for the user in advance, and it is necessary to select a convenient and efficient way of its storage and processing.

Let us point out the main categories of data that need to be stored for further interpretation by the game:

- information about game levels;
- information about prices in the in-game store;
- predefined settings.

Two options are suitable for storing the above data: XML storage and binary serialization.

The ideal option would be not to choose among the proposed technologies, but to use both options, taking the best from each and applying the situation that is ideally suited for the current task.

The big advantage of binary serialization is the small size of the output files. Data is very compactly folded into files and compressed by the most efficient algorithms, but there is one drawback: the inability to read data without special software.

Binary serialization is great for storing a wide variety of settings, prices, and other configuration parameters.

The way the levels are stored requires transparency and easy access to the data. The data should be easily readable using the selected programming language and at the same time be obvious when you simply view the document. That is why the use of documents in XML format was chosen to store game levels.

Data protection. For Windows builds, Unity compiles and saves the source code of all game scripts in the Managed directory. The code is kept in the following libraries: Assembly-CSharp.dll, Assembly-CSharp-firstpass. dll and assemblyunityScript.dll. There are quite convenient and at the same time free utilities for decompiling and viewing the managed .NET code libraries. They are: IlSpy and dotPeek.

The data approach is especially effective for our purposes: Unity optimizes the source code of game scripts quite poor, practically without changing its structure, and also does not hide the names of variables. This makes it easy to read and understand decompiled material.

In such cases, developers have to worry about the security of their code. For such purposes obfuscators are usually used.

Obfuscation is a process due to which the program code takes on a form that is difficult to analyze. Obfuscation is carried out in order to protect the program code and the algorithms it implements from the eyes of others. But in most cases, obfuscation has a lot of side effects. In especially unpleasant circumstances obfuscator

can make the program completely unsuitable for execution, in less severe cases new errors may appear in the program. Therefore, obfuscation should be applied with maximum caution.

In AssetStore there are many ready-made solutions, but most of them are commercial. Free versions, as a rule, are limited or have low efficiency.

Most Unity project resources are packaged in proprietary format files with extensions. assets and.

resources . Despite the closeness of the formats, there are tools for unpacking such files. For example, Unity Assets Explorer is able to extract most textures and shaders from the game. The resulting textures will be in DDS format, which can be opened with Windows Texture Viewer. Shaders are extracted in the already compiled form and there are no solutions for their decompilation. However, this circumstance does not prevent the import and use of the resulting shaders in another Unity project.

Three-dimensional model in Unity build is arranged in different resource, and some of them can be generated at all during the game. One can obtain such data directly from the memory of the graphics accelerator. When the game is running, all the information about the textures and models visible on the screen is in the memory of the video card. Using the 3D Ripper DX utility, you can extract all this information and save it in a format understandable to 3D editors.

PlayerPrefs is a class from the standard Unity library that allows you to save data in the long-term memory of the device. It is a pair of key - value. Developers often use it to store various settings, achievements, player progress, and other information about the game progress. For Windows, this data is stored in the system registry. For other operating systems, data is stored on the device in the local folder of the application in a special file. In most cases, they can be easily accessed and modified using a text editor. For example, in Windows it is enough to use the built-in utility RegEdit to modify any values of PlayerPrefs, thereby changing the configuration and status of the game.

The easiest way to counteract is to encode the stored data, for example, base 64. This method is not very effective, but can provide initial protection against viewing. Checking whether the data was changed without the knowledge of the hash functions will help: by comparing the checksums of the stored data, we can make it sure that nobody and nothing except our code can changed this data.

For more reliable protection, it is necessary to use encryption or various combinations of the listed methods.

You can also implement your own save format. Thanks to Mono Unity keeps working with the file system. Thus, you can serialize all the necessary data, apply strong encryption and save in a safe place.

Conclusion. In this article we examined the methods of data storage for a game application running the Android operating system, as well as some ways to protect them.

Unfortunately, there are not so many ways to protect the game from hacking. Being installed on the user device, it actually reveals all the textures, models and source code. If someone wants to decompile the application and steal resources - it is only a matter of time.

Despite this, there are effective methods that will seriously complicate the lives of hackers.

REFERENCES

1. Scarfone, K. Guide to Storage Encryption Technologies for End User Devices / K. Scarfone, M. Souppaya, M. Sexton. – November, 2007. – 40 p.
2. Unity – Scripting API [Electronic resource]. – Mode of access: https://docs.unity3d.com/ScriptReference/. – Date of access: 22.01.2019.
3. Практическое руководство по взлому (и защите) игр на Unity [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/266345/. – Дата доступа: 22.01.2019.
4. Базы данных и модели данных [Электронный ресурс]. – Режим доступа: http://edu.tltsu.ru. – Дата доступа: 22.01.2019.