INCREDIBUILD

CASE STUDY

FromSoftware

Slashing build time from hours to minutes for more efficient game development lifecycles for PlayStation 3, Xbox 360, and PC Games

Industry

| Game Development

Process

| C++

Results

| Reduced PlayStation 3 builds from 54.2 mins to 13.2 mins

FROM SOFTWARE

FromSoftware is a game development company located in Tokyo, Japan. They have more than 10 years of experience in game development and publishing for different consoles and PC.

They developed the King's Field series, the Armored Core series, Chromehounds, Demon's Souls, Otogi and others.

About the Author

The author graduated from the Graduate School of Science and Technology, Keio University in 2005 and joined FromSoftware in the same year. Since then the author has participated in the design and maintenance of development environment, as well as developing multi platform libraries for internal use until today.

Abstract

At FromSoftware, we develop games for PlayStation 3, Xbox 360, Windows and other platforms. In this article, we would like to present how Incredibuild helped us to shorten the build times of our development cycle.

Overview

In recent years, the number of lines of code has increased dramatically making build times unbearably long. This was the first challenge that we faced. With the help of Incredibuild, we managed to shorten build times to the degree of half to tenth of the original time.

Furthermore, we often need to develop one game for many different consoles or platforms. In order to work with maximum efficiency, a unified development environment that does not vary amongst target platforms is very important. FromSoftware have been using a unified development environment like this for some time. Therefore, the next challenge would be to gain speed while maintaining this easy-to-use environment that we are accustomed to. Although the special requirement in PlayStation 3 development environment has introduced some issues, we managed to solve them by working with Incredibuild.

The Challenge

FromSoftware is developing video games for different platforms such as PlayStation 3 and Xbox 360. Projects for these platforms almost entirely consist of C++ source code. The total size of source code for the core of the game, together with other relevant libraries and frameworks, can easily exceed 2 million lines of code for one single project.

For large projects like this, every build often lasts between several tens of minutes and several hours. To increase efficiency in development, we wanted to decrease the build time down to several minutes.

In addition, since we have a unified development environment which builds our code-base for several target platforms, any addition to this environment had to maintain compatibility and integrate seamlessly with the current functionality.

The Alternatives

We have thought of the following alternatives for speeding up build time:

Review of Source Code Structure: In the C++ building process, the total build time relies much on the dependencies between source files. Therefore, we thought that we could shorten build time by keeping these dependencies at a minimal level. However, a large amount of changes and new code are checked in on a daily basis. It would require a huge amount of time to optimize the dependencies. Furthermore, this kind of optimization limits the ability to tweak code, which represents a significant impediment to game development.

Increasing PC Performance: As we all know, build time depends on PC performance. Therefore increasing PC performance (by buying new machines) is a straightforward option. However, at the time Pentium 4 was the mainstream CPU, and the Intel Core series were not yet available. Although there are faster CPUs around, after evaluating the cost-performance ratio we decided to give up on this option. On the other hand, memory was comparatively cheap so we have given it a try. We gained improvement by adding memory and using RAM disk. Nevertheless, it was not scalable due to the same cost performance problem. The bottom line was that improvement of build times through hardware upgrades was not good enough to achieve our target, not to mention that clearly the scalability of this method was insufficient for future larger projects. Therefore, we would have ruled out this option even if the cost performance was better.

The Solution

Incredibuild And Windows/Xbox360:
In software development, development PCs do not always run at full force. Therefore it is natural to harvest these unused resources for build distribution to fully utilize them.

At the time we were mainly developing for the Windows and Xbox 360 platforms, and just started

development work for PlayStation 3. Therefore, we started off by integrating Incredibuild into our workflow for Windows and Xbox 360 distributed builds.

This was a huge success. We managed to improve build time dramatically for both platforms.

Incredibuild And Playstation 3:
PlayStation 3 introduced a big problem in its build time. Not only did it not benefit from distributed builds, the original undistributed build time was also longer than with other platforms.

Incredibuild was not yet available back then. Therefore, we used another distribution technology to overcome this problem. Through that, we somehow managed to continue our development for the PlayStation 3.

However, running different distributed build system on the same PC caused resource conflicts. It was clearly not what we wanted. Since we were developing for several different platforms, we could not simply divide those PCs into groups, and run each system on different groups. Also, maintaining two systems for the same purpose didn't make sense financially.

After discussing this issue with Incredibuild, they generously let us try their then alpha version. If we can use Incredibuild to distribute PlayStation 3 builds, then we will be able to unify our development infrastructure.

However, we wanted to keep the nice feature of Incredibuild that enables you to run distributed builds within Visual Studio, without the hassle of having to setup every participating PC individually. This is an important characteristic of Incredibuild for efficiency. Although Incredibuild provides a more generic interface compared to the Visual Studio build agent, we could not use it as is because that would break the unified development environment.

The process of building PlayStation 3 projects with Visual Studio is composed of two steps. First, the add-in provided by the development environment generates a Makefile automatically. Then, Visual Studio processes the Makefile to invoke the compiler and the linker.

By default, these two steps are performed as one, but it is possible to insert another step as a Visual Studio build event.

In order to integrate Incredibuild, we inserted a build event which automatically replaces the original Makefile with the one we created. This new Makefile invokes Incredibuild to process the original Makefile, but all compilation tasks are distributed to other machines via Incredibuild's Automatic Interception Interface.

By this, we finally settled down having a development environment which looks and feels the same as before, enabling distributed builds for PlayStation 3 using Incredibuild.

We have been using this method for more than two years now. It has become our de facto standard for PlayStation 3 development.

One Step Further:
As Incredibuild is a generic platform, we further extended its use into other areas.

Calculation of large amount of resource data
Compilation of shaders (it is essential to increase compile speed, as there is a huge number of shaders due to their variations)
PlayStation 3 distributed linking
In PlayStation 3 development, the long linking time is also a problem. We use Incredibuild to distribute the linking process to solve this problem. Usually, linking is the final step of the build in which compiled object files are linked together to create the final product (library or executable), so it was impossible to distribute this process.

Therefore, we introduced another step in-between compilation and the final linking. In this intermediate linking step, several object files are linked into one intermediate object file. In this way, we can distribute the intermediate linking process. In the final linking process, these intermediate object files are linked to produce the final product.

It is much faster to link a few big object files than to link many small object files. Therefore, the total build time was drastically improved by distributing intermediate linking, even though it added another step to the build process.

Although it prevented us from using some of link-time optimization options, those options were way too time consuming to be practical in the first place. Therefore, we gave priority to improving link time instead.

Summary

We integrated Incredibuild into our workflow during the pressure just before the launch of Xbox 360 and PlayStation 3, and it helped us improve our productivity dramatically and enabled us to safely deliver our products on time. Owing to the full compatibility with Visual Studio and the flexibility of Incredibuild, we could integrate them into the workflow seamlessly without too much burden on the developers even during the transition period.

Now, Incredibuild has become a vital part of our game development and all our teams are using them.

The Bottom Line

The following time bars show the build time improvement we managed to achieve in some of our game development projects. The changes are dramatic especially in PlayStation 3 builds. This brought great productivity gain to our projects.

Note that these are different game titles so it is meaningless to compare their build time across platforms.