効果的なラインタイムアセット管理

– 差分アーカイブと自動リロードで制作効率を向上 –

2014 年 2 月 24 日 初稿

板垣 衛

■ 改訂履歴

稿	改訂日	改訂者	改訂内容
初稿	2014 年 2 月 24 日	板垣 衛	（初稿）

■ 目次

■ 概略

別紙の「開発を効率化するためのファイルシステム」で示すファイルシステムを使用することを前提に、効率的に制作を行うためのアセット管理を説明する。

■ 目的

本書は、別紙の「開発を効率化するためのファイルシステム」で示すファイルシステムを活用し、制作業務を効率化させせることを提案することを目的とする。

■ アセット管理について

本書で言うところの「アセット管理」とは、DCC ツールなどで作成するコンテンツデータの管理を指すものではなく、それをオーサリングした後のランタイム用のデータのことを意味する。

アセットのバージョン管理、ファイル共有、（各データ固有の）オーサリング手法については本書の対象としない。

なお、ゲームデータの管理手法についてだけは、別紙の「ゲームデータ管理 DB システム」および「ゲームデータ仕様」にまとめている。

■ 効果的なアセット管理

別紙の「開発を効率化するためのファイルシステム」で示すファイルシステムを活用すること前提に、具体的にどのように制作業務を効率化するのか説明する。

▼ ランタイムアセットファイル

以降の説明部中に用いる「ランタイムアセットファイル」とは、オーサリング後のアセットファイルをアーカイブしたファイルのことである。

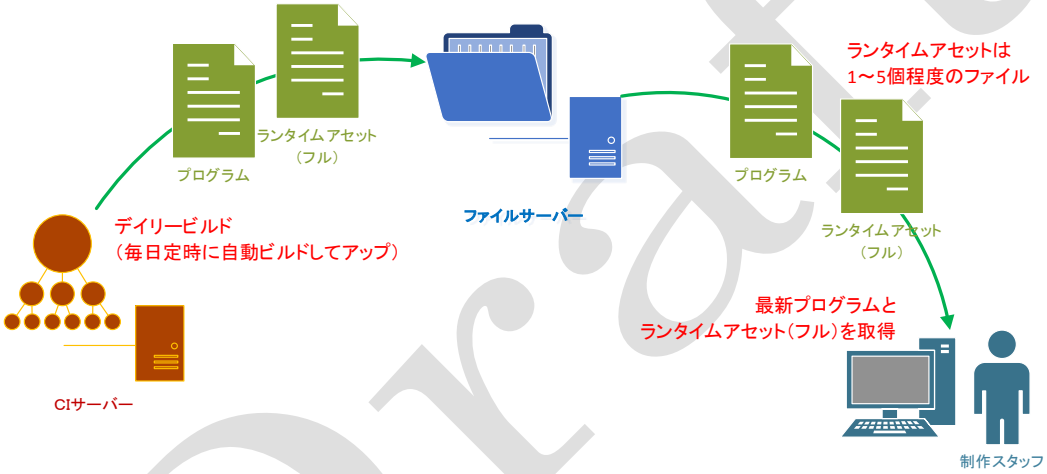
▼ 最新ビルドの利用①：デイリービルド

プログラムの最新ビルドは、CI ツール（Continuous Integration：継続的ビルドツール）により、毎日深夜などの定時に自動的に作成され、ファイルサーバーにアップされる。

この時、プログラムと共にその時点の最新データをオーサリングし、ランタイムアセットを生成する。ランタイムアセットファイルは、1～5 個ほどの大きなアーカイブファイルである。

制作スタッフは、昨晩の内にアップされている最新プログラムファイルとランタイムアセットファイル数個をローカル環境にダウンロードすることで、手早く最新プログラムを実行することができる。

デイリービルドの利用イメージ：



▼ 最新ビルドの利用②：日中更新された最新ビルド

日中プログラマーが更新した最新ビルドを配信する際は、最新プログラムと共に、差分のランタイムアセットファイルをファイルサーバーにアップする。

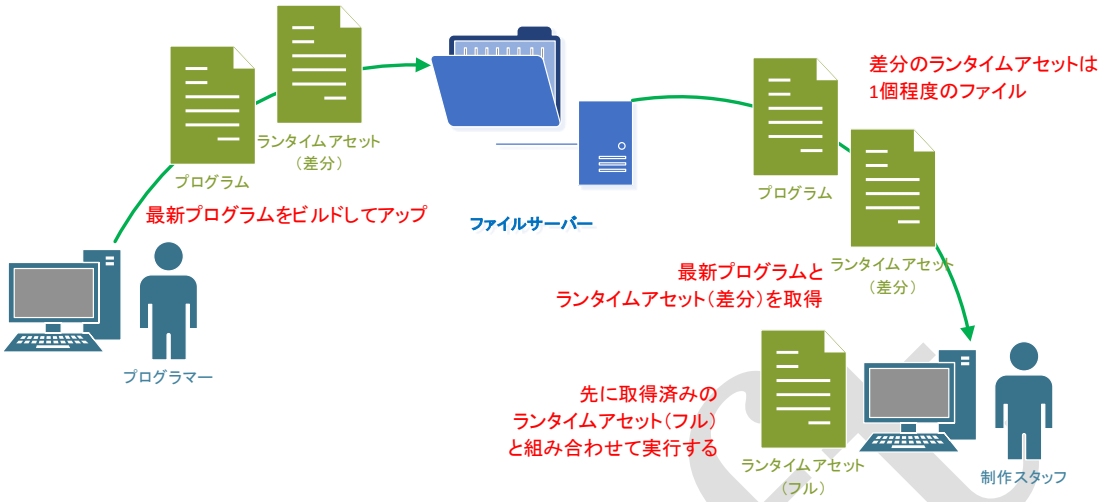
差分ファイルでサイズも小さいので、制作スタッフはすぐにダウンロードして使用できる。

先にダウンロードしているフルランタイムアセットと組み合わて実行する。

問題があって前の状態に戻したい場合は、差分ファイルを削除して以前のプログラムを実行するだけで済む。

翌日には新たなデイリービルドがあるので、差分ファイルはその日限りで不要となる。

日中プログラマーが更新したビルドの利用イメージ：

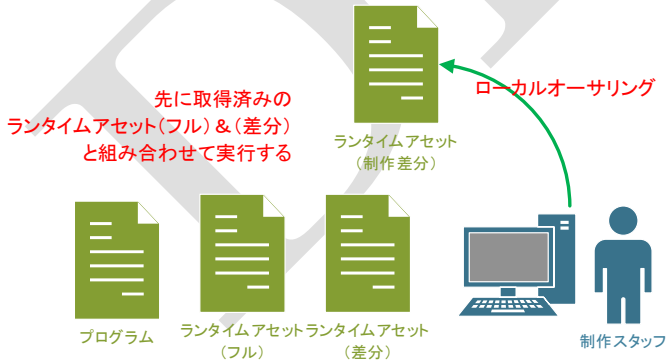


▼ コンテンツ制作業務

日中制作スタッフが作成した最新コンテンツは、ローカルでオーサリングした差分のランタイムアセットファイルを作ることですぐにランタイムで確認できる。

ファイル数はそんなに多くならないはずなので、ローカルオーサリングは短時間で終了し、すぐに使用することができる。

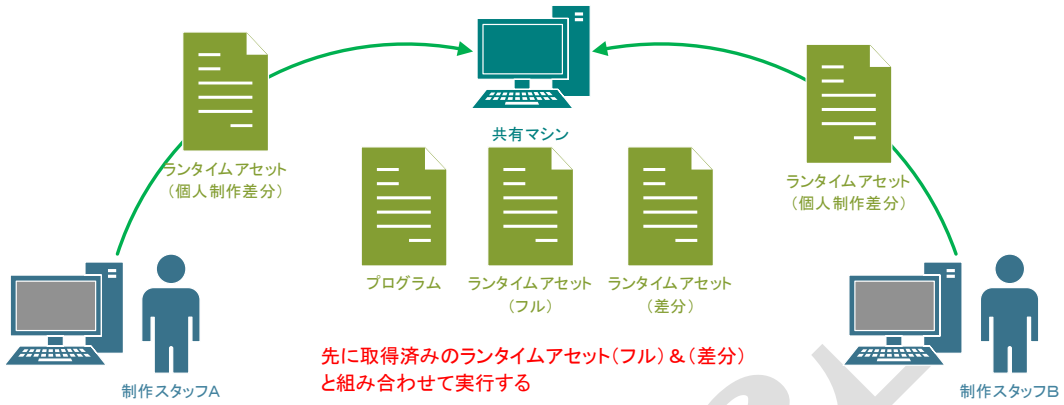
コンテンツ制作業務のイメージ：



▼ 実行環境の共有

複数のスタッフで１台のマシンを共有してゲーム機での実行を確認する場合、ローカルオーサリングした差分ファイル一つ持ち込むだけなので、簡単・安全にマシンを共有し、混乱なく譲り合いできる。

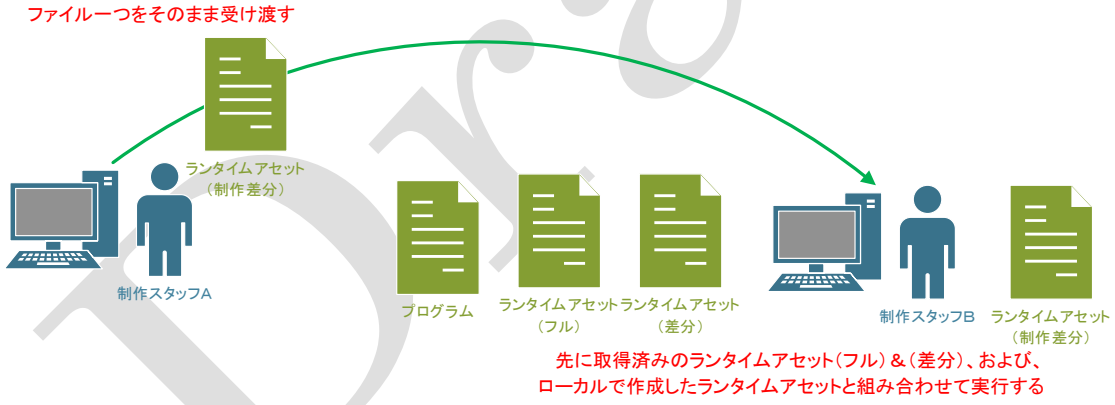
コンテンツ制作業務における、実行環境の共有イメージ：



▼ 他の制作スタッフへの最新データの受け渡し

他のチームの制作スタッフに最新データを渡す必要がある場合、手元の差分ランタイムアセットファイルをそのまま渡せば良い。

スタッフ間でランタイムアセットを受け渡しするイメージ：



▼ アセットの自動リロード

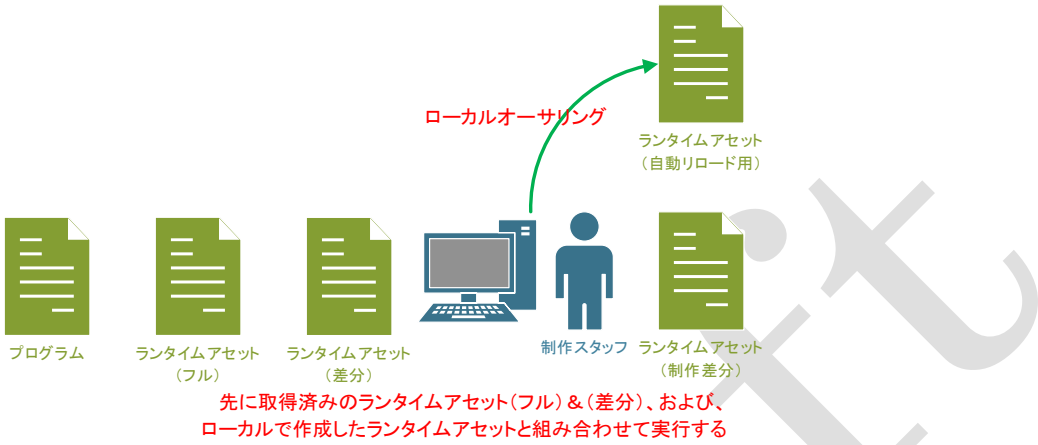
ゲーム実行中に、ゲームを再起動することなく、モデルやテクスチャなどのリソースをリロードすることができる。

リロード専用のランタイムアセットファイルをまとめ、ゲーム上のデバッグメニューなどから「リロード」を実行するだけで、その専用ファイルにまとめたファイルだけが自動的にリロードされる。

自動リロード用ファイルは所定のファイル名で扱うだけで、その中身は通常のランタイムアセットと変わらない。

手続きが少なく扱えるので手軽に利用できる。また、ファイルシステムとリソース管理の基本的な仕組みによって実現するため、スクリプトなどを除くほとんどのリソースが自動リロードに対応する。

自動リロードファイルを使用するイメージ：



なお、リロード完了後は自動的にマウントが外れるので（ゲーム上から扱われなくなるので)、そのままゲームプレイを続行すると、やがてキャッシュが消えて、また古い（リロード前の）データが表示されるようになる。

こうしないと、ゲームがリロード用ファイルを開きっぱなしになり、ファイルを上書きできなくなってしまうためである。リロードファイルを上書きする際の手続きを少しでも減らすために、このようにする。

▼ 多数の差分ファイルを扱う際の手間

ランタイムアセットファイルを多数扱う際は、ファイルシステムの設定ファイルをテキストエディタで編集して、読み込み対象のファイルをあらかじめ設定する必要がある。

しかし、手間はそれだけである。

ファイルシステム設定ファイルのサンプル：

【fsys.json】

```
//ファイルシステム設定ファイル
{
    //マウントするアーカイブファイル
    "mound":
    [
        //優先度が高い順にアーカイブファイルを列挙する
        { "arc": "/user_x.arc" },//個人制作分の差分アーカイブ
        { "arc": "/team_a.arc" },//別チームから一時的に受け取った差分アーカイブ
        { "arc": "/full02.arc" },//フルアセットのアーカイブ02
        { "arc": "/full01.arc" }//フルアセットのアーカイブ01
    ],

    //自動リロード用アーカイブファイル
```

```
    "autoReload": "/user_a_new.arc",
}
```

■■以上■■

■ 索引

索引項目が見つかりません。

効果的なラインタイムアセット管理

以　上