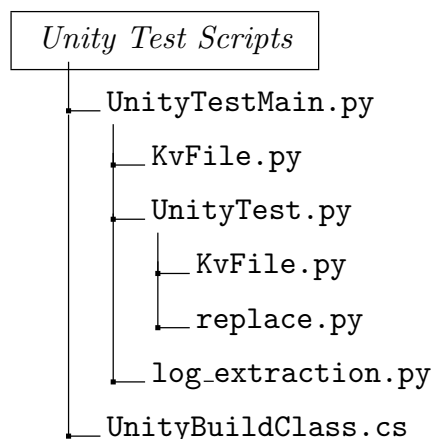


UnityTest システム

UnityTest システムは、unity project 内に存在するシーンファイル (.unity) のビルドと実行を自動的に実施するためのスクリプト群から成る。

1. UnityTestMain.py (1 UnityTestMain 参照)
 - 実行環境および実行するシーンは初期化ファイルで指定する。
 - 初期化ファイルで指定されたシーンを順にビルド&ランする。
 - 個々のシーンのビルド&ランは、スクリプト UnityTest.py で行なう。
 - Unity が出力するログは量が多いので、そのうち compiler が出力した部分のみを log_extraction.py で抽出し、1 つのログファイルにまとめる。
2. UnityTest.py (2 UnityTest 参照)
 - 指定された 1 つのシーンのビルドを行なう。
 - ビルドに成功して exe ファイルが作成されたならばそれを実行する。
 - 最大実行時間を秒単位で指定することができる。
 - 実行環境などは初期化ファイルで指定する。
3. replace.py (3 Replace 参照)
 - ビルド&ラン環境を整えるために、設定ファイルの調整を行なう。
4. log_extraction.py (4 LogExtraction 参照)
 - 2. のビルドにおいて unity editor が出力したログファイルから、コンパイラが出力した部分のみを (重複を除いて) 抽出する。
5. UnityBuildClass.cs (5 UnityBuildClass 参照)
 - 2. のビルドをコマンドラインから実行させるためのクラスを定義する。
 - 引数 -executeMode に指定するのは UnityBuildClass.Build() である。
 - このファイルは UnityProject 下の Assets/.../Editor ディレクトリに置く。

UnityTest のスクリプト構成は次のとおりである。



1 UnityTestMain

1.1 概要

初期化ファイル (起動引数 `-i file` 参照) で指定されたシーン (.unity) を、順次ビルド&ランする。複数のシーンに対してテストを実行するためのラッパである。

1.2 ファイル

UnityTestMain.py	スクリプトファイル
UnityTestMain.ini	初期化ファイル

1.3 関連スクリプト

UnityTest.py †	単一シーンのビルド&ランを行なうスクリプト
KvFile.py	初期化ファイルの解析スクリプト
log_extraction.py †	ログの抽出スクリプト

† 初期化ファイルにて変更可

1.4 初期化ファイル

初期化ファイルは Key-Value-Pair (key と value を空白またはタブで区切って定義した) ファイルであること。次の項目を指定する。

1. 必須項目

Key	Value
UnityTestScript	単一シーンのビルド&ランスクリプト (パス名)
ExtractScript	ログ抽出を行なうスクリプト (パス名)
Python	Python 実行形式 (python.exe) (パス名)
Springhead2	Springhead2 のルートディレクトリ (パス名)
SceneList	テストするシーンの名前を空白で区切って並べたリスト (ファイル名) (.unity はなくてもよい)

備考：特に指定がない限り、パス名は絶対パスで指定することが望ましい。

2. 任意項目

Key	Value
UnityProject	Unity プロジェクトのルート (パス名) default: \$(Springhead2)/src/Unity
TestRoot	UnityTest システムのルート (パス名) default: \$(Springhead2)/src/UnityTest
Inifile	UnityTest.py の初期化ファイル (パス名) default: \$(TestRoot)/UnityTest.ini
ScenesDir	シーンファイルが置かれているディレクトリ \$(UnityProject) からの相対パス名 default: Assets/Scenes
TestMainLogFile	UnityTestMain が作成するログファイル (パス名) default: \$(TestRoot)/log/TestMain.log

備考：特に指定がない限り、パス名は絶対パスで指定することが望ましい。

1.5 ヘルパーメソッド

1.5.1 s16(value)

概要 与えられた 16 ビットのビットパターンを符号付き整数に変換して返す
引数 value 16 ビットのビットパターン

1.5.2 logmsf(msg, file)

概要 指定された msg を file に追加する
引数 msg ログメッセージ
file 書き出すファイル

1.5.3 verbose(msg, level=0)

概要 バーバス情報の表示
引数 msg メッセージ
level 表示制御変数
説明 起動引数で指定した -v オプションの個数が level で指定した数より大きいときに msg を表示する

1.5.4 error(msg)

概要 stderr に msg を書き出す (改行付き)
引数 msg メッセージ

1.6 プログラムの説明

1.6.1 起動形式

UnityTestMain.py [*options*]

1.6.2 起動引数の解析

OptionParser パッケージを用いて解析する

オプション引数：

-i <i>file</i>	初期化ファイル名
-l <i>file</i>	ログファイル名
-t <i>time</i>	タイムアウト値 (秒)
-T	スナップに日付と時刻を付加
-1	SceneList の最初の 1 つのみ実行 (デバッグ用)
-h	ヘルプの出力
-v	バーバス情報出力指定 (複数個指定可)
-V	バージョン情報表示

1.6.3 初期化ファイルの読み込み

初期化ファイル (-i オプションで指定、default: ./UnityTestMain.ini) を読んでパラメータを設定する。指定されなかったパラメータには次のデフォルト値を設定する。

UnityProject	\$(Springhead2)/src/Unity
TestRoot	\$(Springhead2)/src/UnityTest
Inifile	\$(TestRoot)/UnityTest.ini
ScenesDir	Assets/Scenes
TestMainLogFile	\$(TestRoot)/log/TestMain.log

1.6.4 ログファイルの初期化

ログファイル (TestMainLogFile) を空にする。

1.6.5 SceneList に指定されたシーンのすべてについて

1. ログファイル (TestMainLogFile) にシーン名を追加する
2. シーンのビルド&ランを行なう (UnityTest.py)
3. 2 で unity が作成したログファイルから compiler が生成した部分を抽出し、ログファイル (TestMainLogFile) に追加する (4.LogExtraction 参照)

2 UnityTest

2.1 概要

起動引数で指定されたシーン (.unity) を、ビルド&ランする。扱うのは 1 つのシーンのみである。

2.2 ファイル

UnityTest.py	スクリプトファイル
UnityTest.ini	初期化ファイル

2.3 関連スクリプト

KvFile.py	初期化ファイルの解析スクリプト
replace.py	unity 設定ファイルの変更スクリプト

2.4 初期化ファイル

初期化ファイルは Key-Value-Pair (key と value を空白またはタブで区切って定義した) ファイルであること。次の項目を指定する。

1. 必須項目

Key	Value
Unity	unity 実行形式 (unity.exe) (パス名)
Python	Python 実行形式 (python.exe) (パス名)
Springhead2	Springhead2 のルートディレクトリ (パス名)
DllPath	Springhead2 ライブラリのあるディレクトリ (パス名)

備考：特に指定がない限り、パス名は絶対パスで指定することが望ましい。

2. 任意項目

Key	Value
UnityProject	Unity プロジェクトのルート (パス名) default: \$(Springhead2)/src/Unity
ScenesDir	シーンファイルが置かれているディレクトリ \$(UnityProject) からの相対パス名 default: Assets/Scenes
TestRoot	UnityTest システムのルート (パス名) default: \$(Springhead2)/src/UnityTest
OutFile	ビルドで作成する実行形式 (パス名) default: \$(TestRoot)/bin/player.exe
LogFile	unity が作成するログファイル (パス名) default: \$(TestRoot)/log/build.log

備考：特に指定がない限り、パス名は絶対パスで指定することが望ましい。

2.5 ヘルパーメソッド

2.5.1 get_date()

概要 今日の日付・時刻を "yyyy/mm/dd hh:mm:ss" 形式で返す

2.5.2 dir_part(path)

概要 与えられた path のディレクトリ部分を返す

引数 path パス名

2.5.3 s16(value)

概要 与えられた 16 ビットのビットパターンを符号付き整数に変換して返す

引数 value 16 ビットのビットパターン

2.5.4 result_str(code)

概要 与えられたコード値に従い、成功か失敗かを示す文字列を返す

引数 code リターンコード

説明 $code \geq 0$ なら "*code* (success)" を返す

$code < 0$ なら "*code* (fail)" を返す

2.5.5 verbose(msg, level=0)

概要 バース情報を表示する

引数 msg 表示する文字列

level 表示制御変数

説明 起動引数 `-v` オプションの個数が、level で指定した数より大きいときに msg を表示する

2.5.6 info(msg, name=None, has_next=False, continued=False)

概要 与えられた msg を表示する (-T オプション対応、改行制御付き)

引数

msg	メッセージ
name	指定があれば行の先頭に表示する ("name: ")
has_next	ここでは改行したくないとき True とする
continued	ここで改行したいとき True とする

説明 has_next/continued は -v オプションが指定されたときは意味をもたない

- v オプションが指定されたとき：
必ず改行する (-T, name 対応)
- v オプションが指定されていないとき：
has_next が True なら改行せずに "... " と表示する
continued が True なら改行する (-T, name 非対応)

2.5.7 fatal(msg, code=-1)

概要 メッセージを stderr に出力してスクリプトを終了する

引数

msg	メッセージ
code	終了コード

2.6 プログラムの説明

2.6.1 起動形式

UnityTest.py [*options*] scene

2.6.2 起動引数の解析

OptionParser パッケージを用いて解析する

必須引数：

scene	シーン名
	".unity" で終わっていなければ ".unity" を追加する

オプション引数：

-i <i>file</i>	初期化ファイル名
-t <i>time</i>	タイムアウト値 (秒)
-T	スナップに日付と時刻を付加
-h	ヘルプの出力
-v	バーバス情報出力指定 (複数個指定可)
-V	バージョン情報表示

2.6.3 初期化ファイルの読み込み

初期化ファイル (-i オプションで指定、default: ./UnityTest.ini) を読んでパラメータを設定する。指定されなかったパラメータには次のデフォルト値を設定する。

UnityProject	<code>\$(Springhead2)/src/Unity</code>
ScenesDir	<code>Assets/Scenes</code>
TestRoot	<code>\$(Springhead2)/src/UnityTest</code>
OutFile	<code>\$(TestRoot)/bin/player.exe</code>
LogFile	<code>\$(TestRoot)/log/build.log</code>

2.6.4 OutFile/LogFile のディレクトリの作成 存在しないときのみ

2.6.5 unity の ProjectSettings の変更

本テストを実施することによる unity 環境の変更を回避するため、

1. 該当ファイルの退避
2. 該当ファイルの変更
3. テストの実施 (2.6.6 参照)
4. 該当ファイルの復元 (2.6.7 参照)

という手順を踏む。現在判明している限りでは、

`"$(UnityProject)/ProjectSettings/ProjectSettings.asset"` ファイルに
`displayResolutionDialog: 1`

という設定があると、`$(OutFile)` を実行したときにダイアログが開いてしまい、スクリプトが中断 (入力待ち) してしまう。これを回避するために、この行を

`displayResolutionDialog: 0`

と変更してからテストを実施する。

1. ファイルの退避
 上記のファイル `"ProjectSettings.asset"` を `"ProjectSettings.asset.save"` という名前のファイルにコピーする。
2. ファイルの書換え
`"ProjectSettings.asset"` の該当行を書き換える (`replace.py`)

2.6.6 ビルド&ラン

1. `$(Outfile)` を削除する (ビルドの成功を確認するため)
2. `unity` をコマンドラインから次の引数で起動する。

-projectPath \$(ProjectPath)	開く unity プロジェクトのパス
-executeMethod BuildClass.Build	unity で実行するメソッド名
-batchmode	unity をバッチモードで実行
-quit	実行終了後 unity エディターを終了
-logfile \$(LogFile)	エディターが書き出すログファイル
-output \$(OutFile)	作成する standalone 実行ファイル
-target \$(ScenesDir)/scene	ビルドするシーン名

3. ビルドに成功したら (\$(OutFile) が作成されたら)、PATH に\$(DllPath) を加えて \$(OutFile) を実行する。

タイムアウト処理

Windows では SIGALARM が使えないため、本来の意味でのタイムアウト制御は行なえない。ここでは、次のような方式で制御を実施する。

1. サブプロセスを生成して unity を開始する。制御はすぐに戻ってくる。
2. 指定された秒数だけ sleep する。
3. sleep が終了したら、1 で生成したサブプロセスが実行中かどうかを調べる。
 - (a) 実行中ならば、サブプロセスを kill してテストは成功とする。
 - (b) 既に終了していれば、サブプロセスの終了コードを取り出す。

この方式だと、サブプロセスが早く終了しても sleep から戻ってくるまで無条件に待たされることになる。しかし、タイムアウト値に大きい値を指定しない限り、実用上問題はないと思われる。

2.6.7 ProjectSettings の復元

1. ファイル "ProjectSettings.asset" を削除する。
2. 2.6.5 で ProjectSettings を退避したファイル "ProjectSettings.asset.save" を改めて "ProjectSettings.asset" とする。

3 Replace

3.1 概要

入力ファイルの各行について、指定したパターンの置換を施した結果を出力ファイルに書き出す。置換するパターンが見つからなかったときは、入力ファイルと出力ファイルの内容は同一となる。

3.2 ファイル

replace.py スクリプトファイル

3.3 ヘルパーメソッド

3.3.1 replace(iframe, ofname, patterns, sep)

概要 入力ファイルの各行について置換処理 (string.replace()) を施し、結果を出力ファイルに書き出す。置換処理を実施した行数を返す

引数 ifname 入力ファイルのパス
 ofname 出力ファイルのパス
 patterns 置換パターン指定 (複数指定可)
 各パターンは <FROM><sep><TO> の形式とする。
 <FROM> マッチさせるパターン
 <TO> 置き換えるパターン
 <sep> <FROM>と<TO>との分離文字
 sep patterns の分離文字 (デフォルトは'=')

説明 置換処理は行単位で、patterns に指定した順序で実行する
 a=b b=c は a=c と同じ結果となる

3.4 プログラムの説明

3.4.1 起動形式

```
replace.py [options] infile outfile from=to [from=to]..
```

3.4.2 起動引数の解析

OptionParser パッケージを用いて解析する

必須引数：

infile 入力ファイルのパス
outfile 出力ファイルのパス
from=to 置換パターン指定 (複数指定可)
 ヘルパーメソッド replace() での説明を参照

オプション引数：

-s <i>sep</i>	patterns の分離文字 (デフォルトは'=')
-h	ヘルプの出力
-v	バーバス情報出力指定 (複数個指定可)
-V	バージョン情報表示

3.4.3 置換処理の実行

3.4.2 で取り出した引数情報を `replace()` に与えて置換処理を実行する。

4 LogExtraction

4.1 概要

unity が生成したログファイルからコンパイラが生成したログの部分だけを抽出する。

コンパイラが生成したログの部分の判定条件：

"CompilerOutput:" を含む行で始まり

"EndCompilerOutput" を含む行で終る

この判定条件を満たす行を抽出する。

特別な行として

"-target:" で始まり ".unity" で終る

行も抽出する (抽出結果がどのシーンに関するものかを識別できるようにするため)

4.2 ヘルパーメソッド

4.2.1 verbose(msg, level=0)

概要 バース情報の表示

引数 msg 表示する文字列

level 表示制御変数

説明 起動引数で指定した -v オプションの個数が level で指定した数より大きいときに msg を表示する

4.2.2 error(msg)

概要 エラーメッセージとして stderr に表示する

引数 msg エラーの内容

4.2.3 is_new(str, list)

概要 str が list の中で初出かどうかを判定する

引数 str 判定する文字列

list 既出文字列のリスト

説明 str が list の中になければ初出として True を返し、str を list に登録する
str が既に list の中にあれば False を返す

4.3 プログラムの説明

4.3.1 起動形式

log-extraction.py [options] file

4.3.2 起動引数の解析

OptionParser パッケージを用いて解析する

必須引数：

file 解析するログファイルのパス

オプション引数：

-h ヘルプの出力

-v バース情報出力指定（複数個指定可）

-V バージョン情報表示

4.3.3 正規表現の定義とコンパイル

次のパターンにマッチさせる

pattern 1: '^-target: (.+)\.unity\$'

pattern 2: '^-----CompilerOutput:'

pattern 3: '^-----EndCompilerOutput-----',

4.3.4 出力 (抽出) を制御する変数

読み込んだ行を出力するか否かは次の変数で制御する (True のとき出力する)

include ある範囲 (複数行) の出力を制御するために使用
 特定のパターンにマッチしたら on または off する
 初期値は off (False)

include_one 単一の行に関して出力を制御する (次の行には影響しない)

4.3.5 ファイルの各行について

1. include_one を off にする
2. パターンマッチを行ない制御変数を変化させる

マッチ	include	include_one	説明
pattern 1	—	on	シーン名の抽出
pattern 2	on	—	コンパイラ出力開始
pattern 3	off	on	コンパイラ出力終了

3. include か include_one のいずれかが on ならばこの行を出力する

5 UnityBuildClass

5.1 概要

コマンドラインから `unity` を実行させるために定義したクラスである。このクラスには、`unity` のコマンドライン引数 `-executeMode` に与えるメソッドを定義している。

注意 `unity` をコマンドラインから実行するに際して、実行環境を独自に設定する方法はないようである。即ち、`UnityTest` の実行に関して設定した環境が状態として記録されてしまい、それが他の `unity` の実行 (例えば UI からの実行) に影響を与えてしまうことになる。現在のところ、次の環境については退避と復元を実施しているが、他にも同様の処置を施さなければならない環境があるかも知れない。そのような環境が確認されたときは、退避と復元のコードを追加する必要がある。

現在退避と復元を実施している環境：

ビルド対象プラットフォーム シーンを実行させるプラットフォームを選択する
(`UnityTest` では `StandaloneWindows` を選択)

`ProjectSettings` の中の `displayResolutionDialog` exe 実行時に `Screen resolution/Graphics quality` などの確認ダイアログを制御する (`UnityTest` では抑止する) ただし、この環境は UI からでなければ変更できないようである。従って、ビルド実行の直前/直後に設定ファイルを直接編集することで退避と復元を実施する。これについては 2.UnityTest を参照のこと。

5.2 メソッド

5.2.1 Build()

概要 unity から直接呼び出されるメソッド

処理の流れ

現在環境の退避 ビルドの実行 環境の復元 結果の表示

説明 ビルドは BuildPipeline.BuildPlayer() を呼び出すことで実現する
このメソッドに与える引数は次のとおり

levels ビルドに含むシーン (.unity)
プロジェクトディレクトリからの相対パス
unity 起動時引数 -target scene から取得する
5.2.2 GetSceneName() 参照

locationPathName
成果物の保存先のパス (.exe)
unity 起動時引数 -output path から取得する
5.2.3 GetOutputPath() 参照

target ビルド時の対象プラットフォーム
BuildTarget.StandaloneWindows を指定する

options ビルドしたプレイヤーをどのように実行するか
BuildOptions.None を指定する (ビルドのみ)

5.2.2 GetSceneName()

概要 ビルドするシーンの名前を unity 起動時引数 -target scene から取得する

説明 引数の取り出しには GetArg() を使用する

5.2.4 GetArg() 参照

5.2.3 GetOutputPath()

概要 成果物の保存パスを unity 起動時引数 -output path から取得する

説明 引数の取り出しには GetArg() を使用する

5.2.4 GetArg() 参照

5.2.4 GetArg(string key)

概要 unity 起動時引数全体から、指定されたキーに対応するものを取得する

説明 起動時引数の全体は System.Environment.GetCommandLineArgs() で
取得し、この中から指定された key に対応するものを抽出する