

1. 目次

[目次](#)

[はじめに](#)

[インストール編](#)

[Toboggan のインストール](#)

[PuTTY のインストール](#)

[Visual C++ ランタイムコンポーネントのインストール](#)

[Toboggan 基礎編](#)

[新しくプロジェクトを作る](#)

[ファイルを配置する](#)

[新しくテクスチャアセットを作る](#)

[新しく UI アセットを作る](#)

[start.lua の編集](#)

[プロジェクトをパブリッシュ#する](#)

[パブリッシュしたプロジェクトを使ってゲームを実行する](#)

[既存のプロジェクトを開く](#)

[Toboggan を終了する](#)

[Toboggan SVN編](#)

[リポジトリの認証設定](#)

[PuTTYgen で秘密鍵・公開鍵の作成](#)

[秘密鍵を Pageant に登録](#)

[公開鍵をリポジトリサーバに登録申請](#)

[Pageant の常駐化](#)

[プロジェクトをリポジトリからチェックアウトする](#)

[プロジェクトをリポジトリにコミット#する](#)

[最新のプロジェクトをリポジトリから取得する（アップデート）](#)

[付録](#)

[用語集](#)

[ショートカットキー表](#)

2. はじめに

- a. **Toboggan** とは、ゲームの作成に必要なアセットを作成・管理する為のアセットマネージャです。
組み込まれている各種プラグイン（Texture エディタ、UI エディタ等）を使用してアセットを作成します。
- b. 本マニュアルは、Toboggan を使用してプロジェクトの作成・管理を一通り行えるようになる為の基本的な操作方法を説明しています。
本マニュアルで使用するプラグインは以下の通りです。
 - i. Texture エディタ
 - ii. UI エディタ

各プラグインの詳細は Toboggan Plugin Manual (JPN).docx を参照ください。

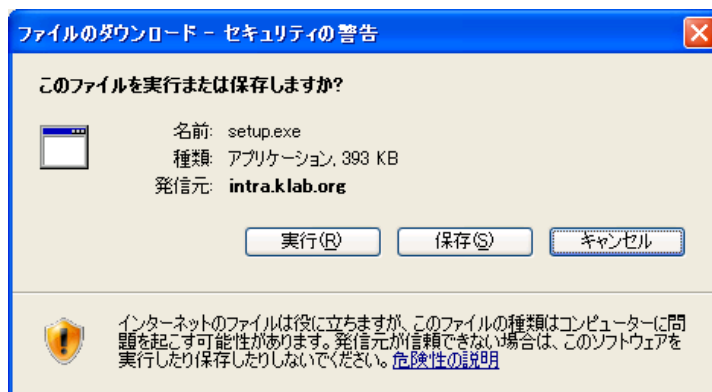
3. インストール編

Notice!

Toboggan その他ツールのインストールを始める前に、**Windows Update** 等で Windows を最新の状態にして下さい。
最低限のセキュリティを確保すると同時に、インストール作業を滞りなく進める為に必要です。

a. Toboggan のインストール

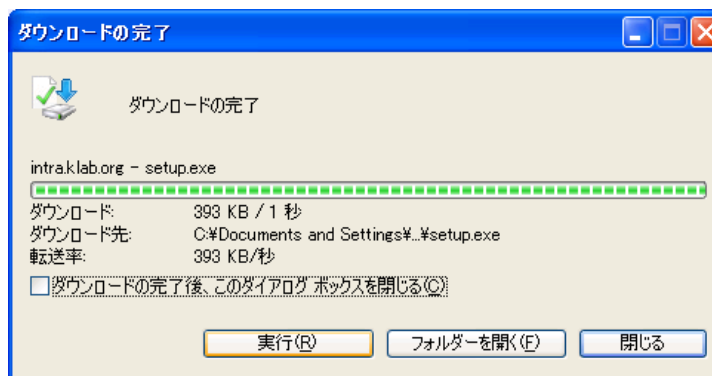
- i. Setup.exe をダウンロードする。



【setup.exe のダウンロードダイアログ】

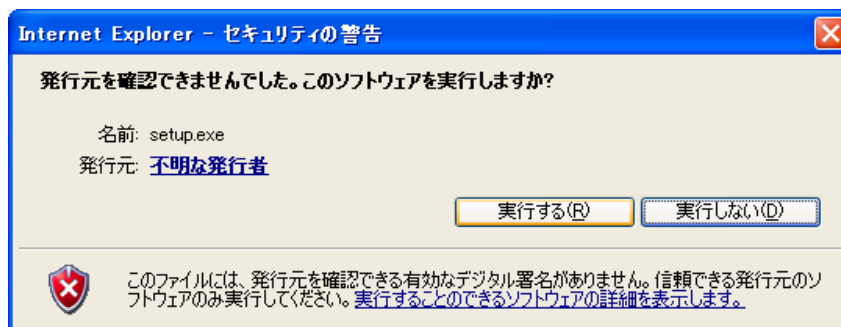
【保存】ボタンをクリックし、任意のディレクトリに保存する。

- ii. setup.exe を実行し、表示されたウィザードに従いインストールを完了する。



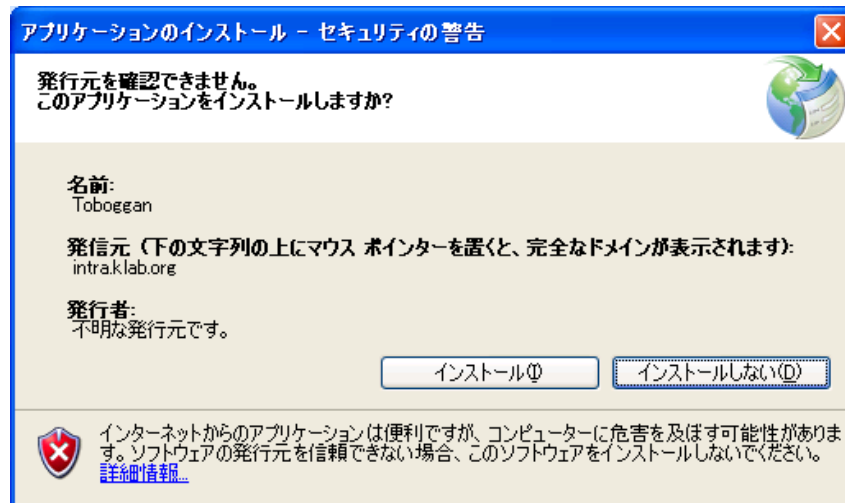
【setup.exe のダウンロード完了ダイアログ】

そのまま **【実行】** ボタンをクリックする。



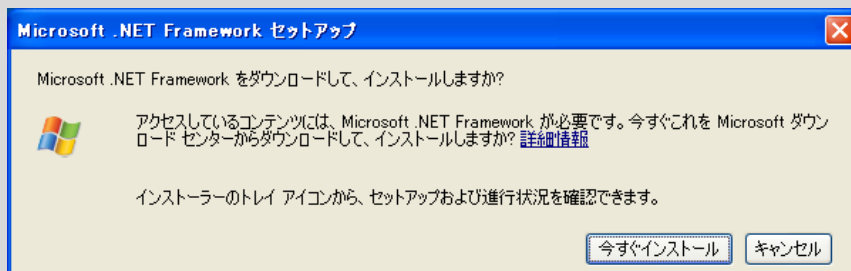
【セキュリティの警告ダイアログ】

今回はイントラネットからのダウンロードなので、このまま **[実行する]** をクリックする。



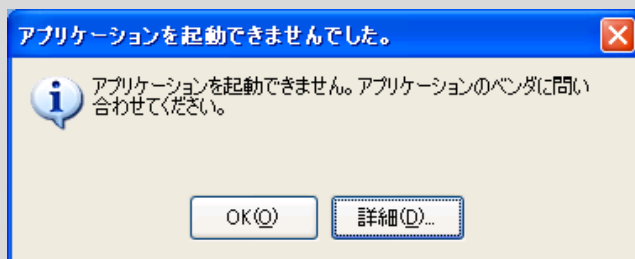
【Toboggan アプリケーションのインストールダイアログ】
発行者が不明の為、ここでもセキュリティの警告が表示されるが、上記同様 **[インストール]** をクリックする。

.NET Framework のセットアップダイアログが表示される場合

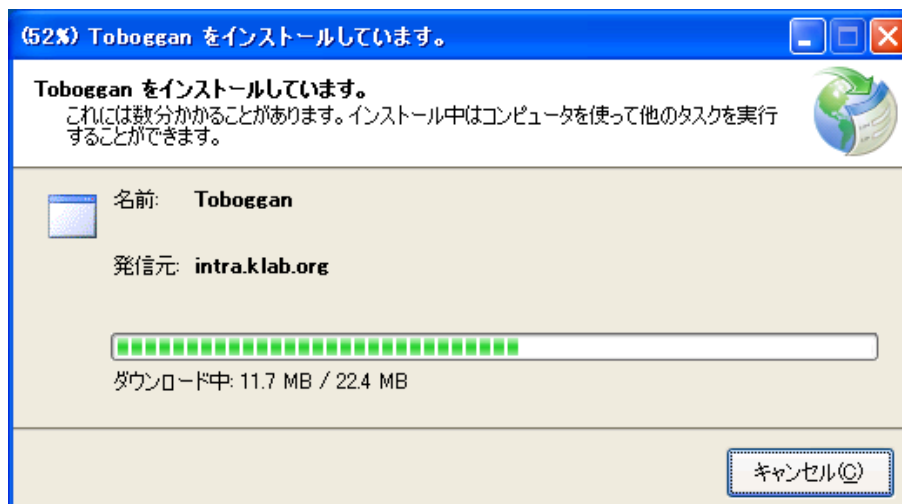


マシンに .NET Framework がインストールされていない場合にのみ、上記ダイアログが表示されます。
Toboggan の実行には .NET Framework が必須ですので、**[今すぐインストール]** ボタンをクリックしてウィザードに従い、.NET Framework をインストールして下さい。

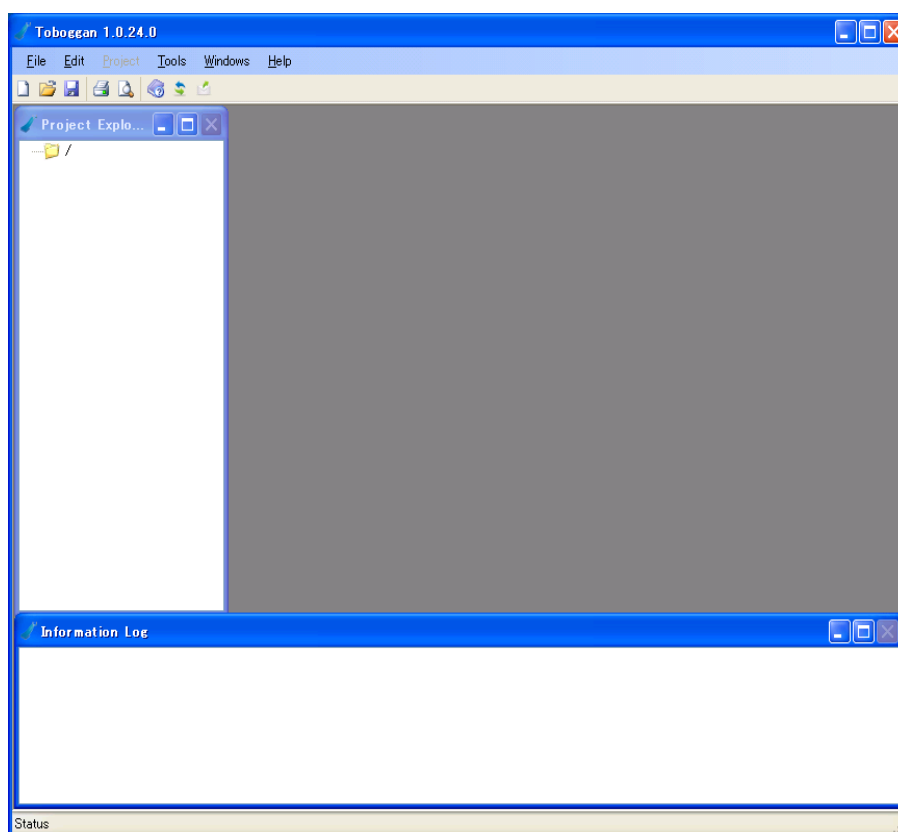
アプリケーションのインストールに失敗する場合



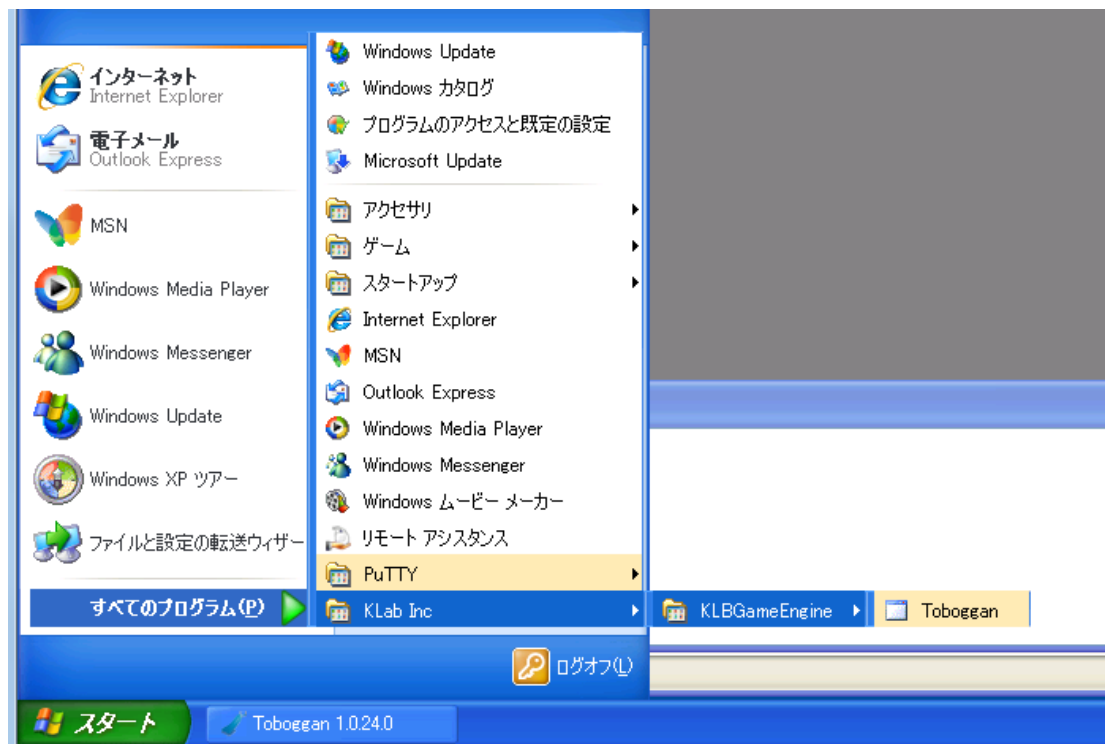
インストールボタンをクリックすると、左記のようなダイアログが表示されインストールが進まない場合、必要な .NET ランタイムがインストールされていない可能性があります。
Windows Update 等でマシンを最新にして再試行してみてください。



【インストールの進捗ダイアログ】
Toboggan のダウンロード及びインストールの進捗率が表示される。



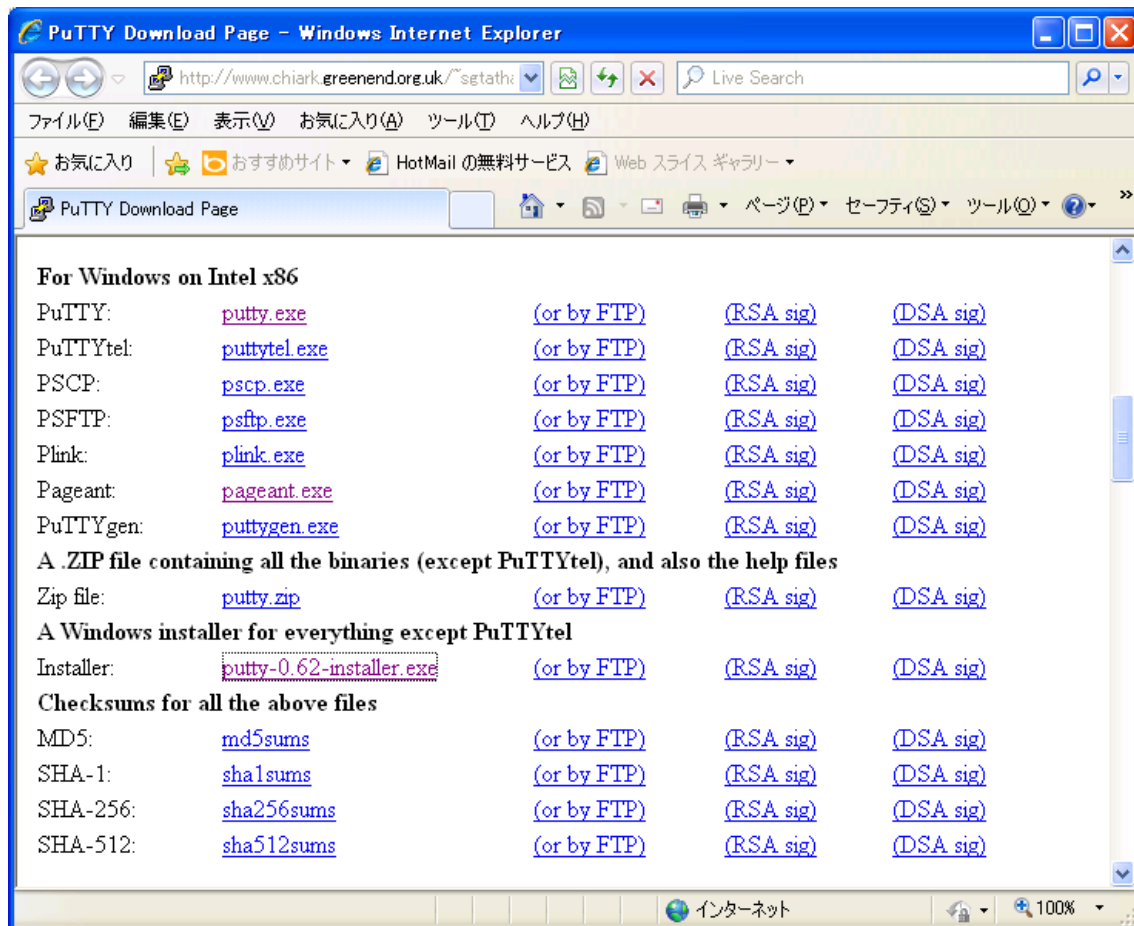
【Toboggan の起動】
インストールが完了すると、自動的に Toboggan が起動し、上図のようなウィンドウが表示される。



【すべてのプログラムの確認】
[KLab Inc] > [KLBGameEngine] > [Toboggan] が表示されていれば、インストールに成功している。

b. PuTTY のインストール [オプション]

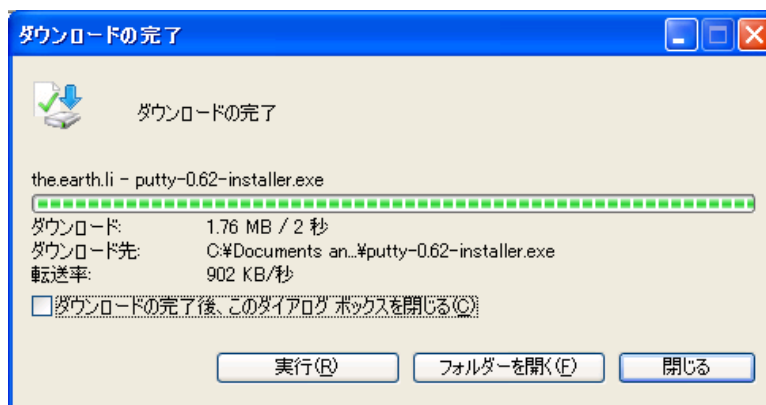
- i. PuTTY のダウンロードページ (<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>) から **putty-0.62-installer.exe** をダウンロード。



【PuTTY のダウンロードページ】

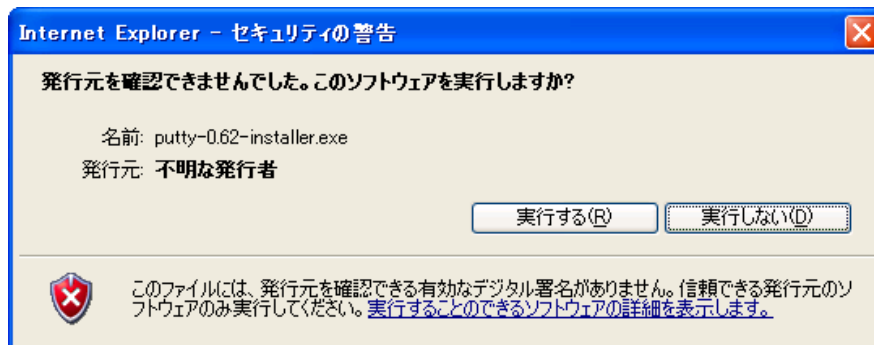
putty-0.62-installer.exe リンクをクリックして、PuTTY のインストーラをダウンロードする。

- ii. **putty-0.62-installer.exe** を実行し、表示されたウィザードに従いインストールを完了する。

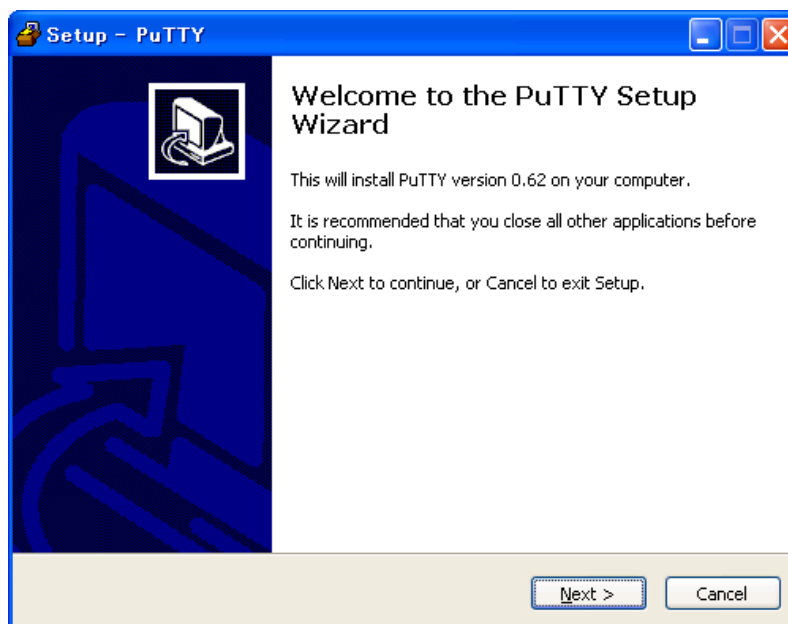


【putty-0.62-installer.exe のダウンロード完了ダイアログ】

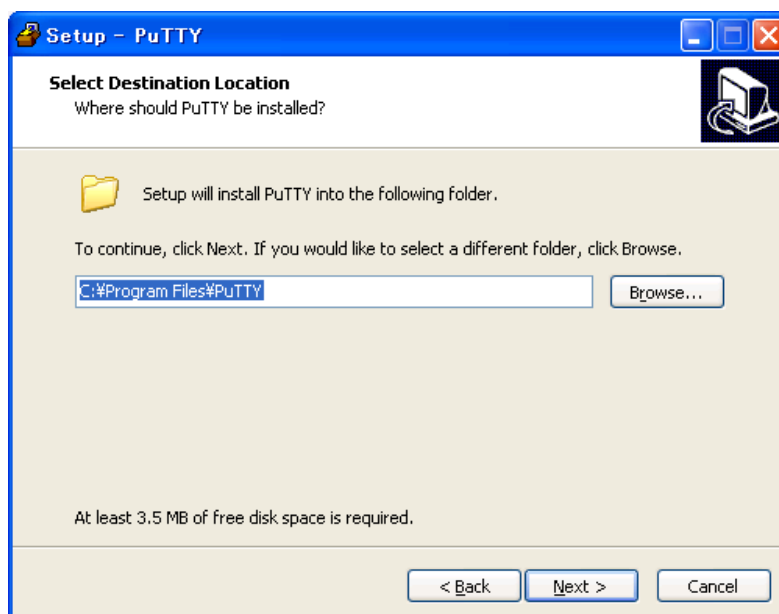
[実行] ボタンをクリックする。

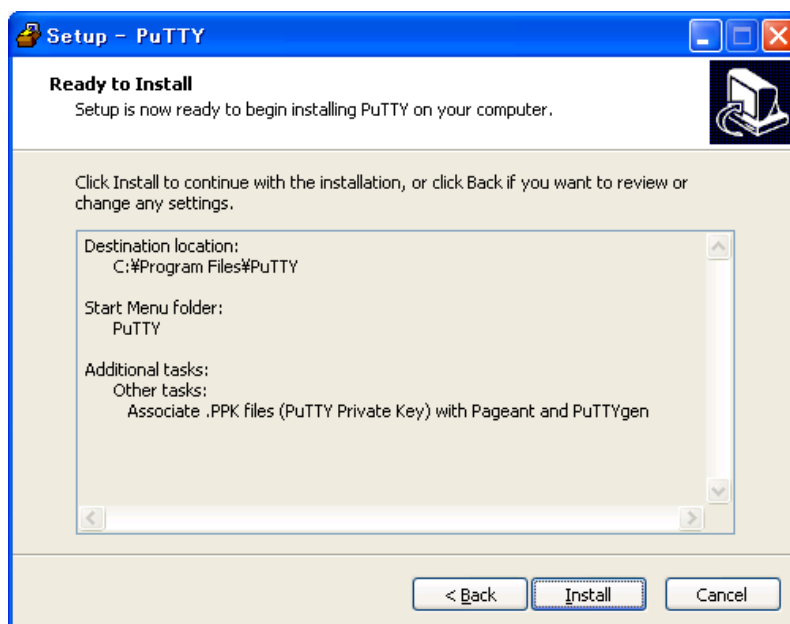
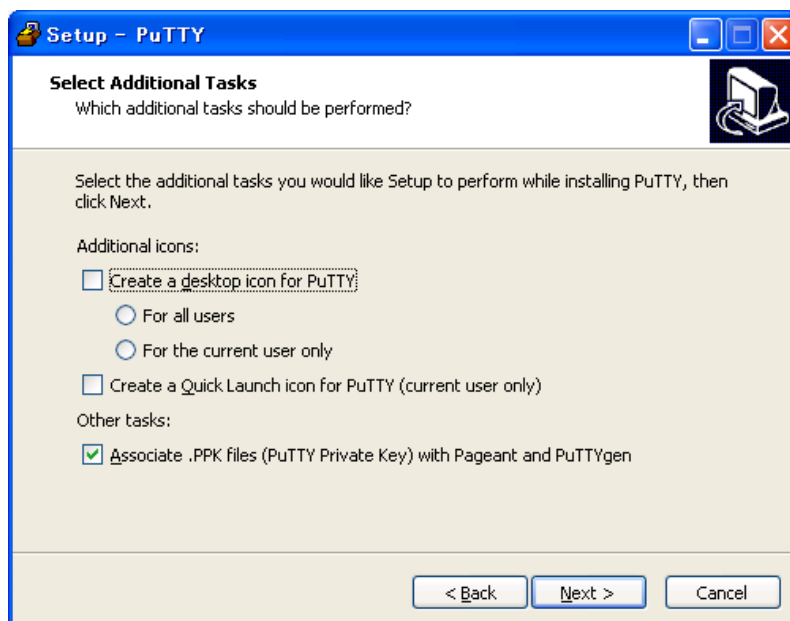
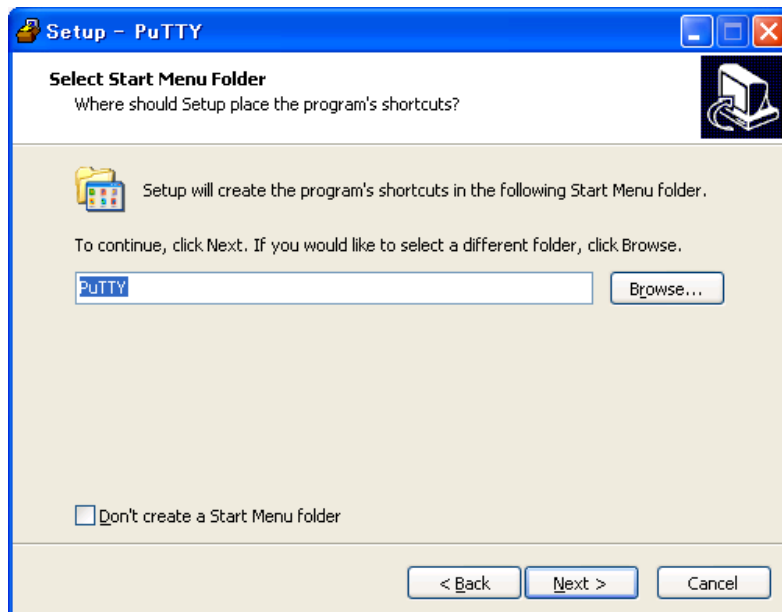


【セキュリティの警告ダイアログ】
デジタル署名が無いいため警告ダイアログが表示されるが、今回はダウンロード元を信用して[実行する]をクリックする。

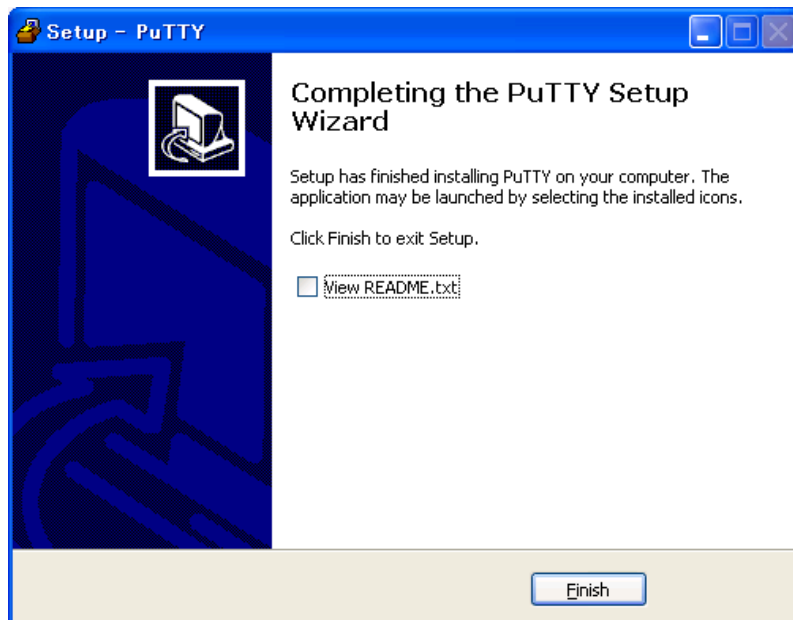
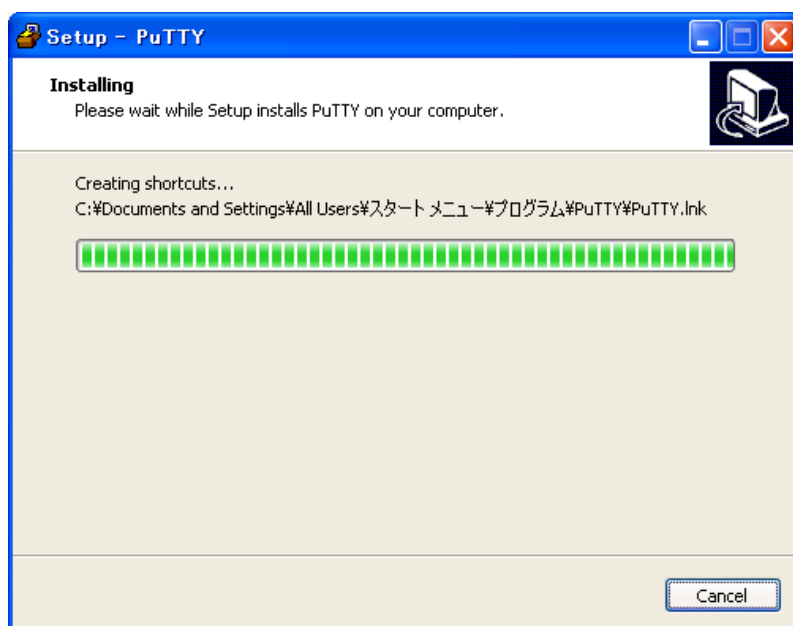


【PuTTY のインストールウィザード】
後は、ウィザードに従ってセットアップする。
基本的には何も変更せず [Next] ボタンをクリックしていけば良い。

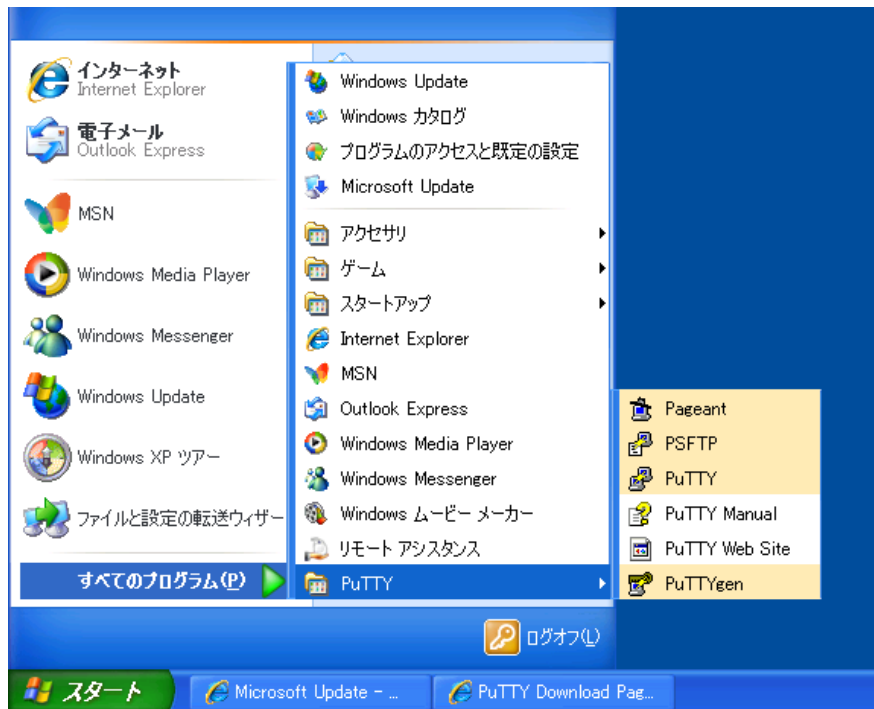




【インストール内容の確認ダイアログ】
ボックス内に表示されている内容であってれば、[Install] ボタンをクリックする。



【インストールの完了ダイアログ】
PuTTY がインストールされたので、**[Finish]** ボタンをクリックしてインストールウィザードを終了する。



【すべてのプログラムの確認】
PuTTY が表示されていれば、インストールに成功している。

c. Visual C++ ランタイムコンポーネントのインストール

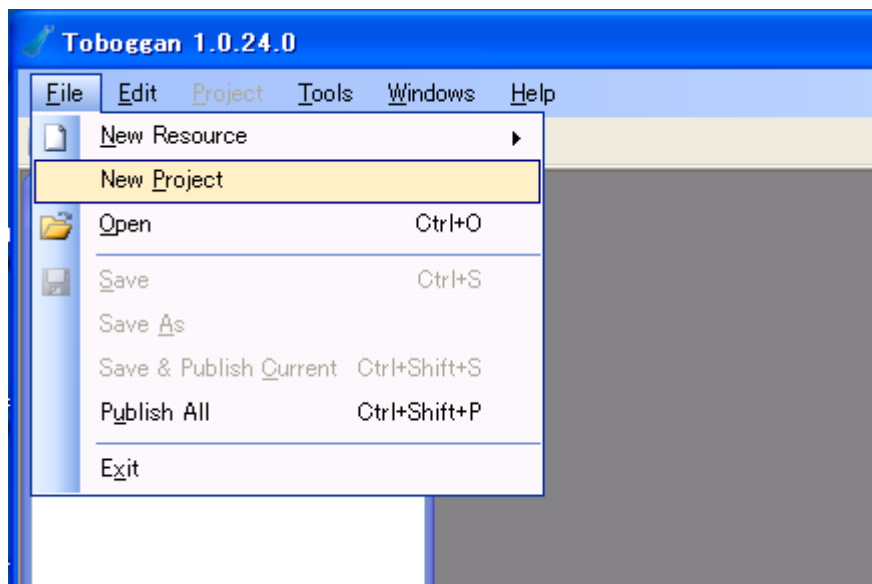
Notice!

Visual C++ ランタイムは、[パブリッシュしたプロジェクトを使ってゲームを実行する](#) の項でゲームエンジンを実行する場合にのみ必要になります。
ゲームエンジンを実行する必要が無ければ、本項は読み飛ばして下さい。

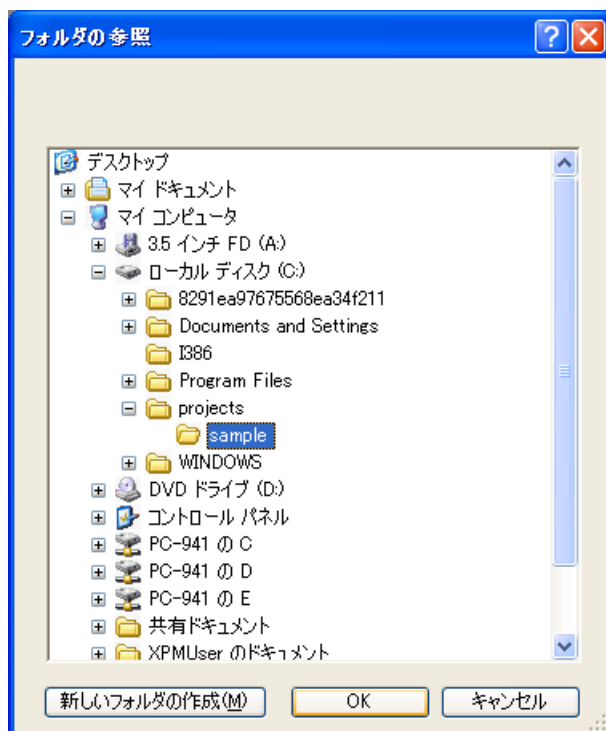
- i. Microsoft Visual C++ 2010 再頒布可能パッケージのダウンロードページ (<http://www.microsoft.com/ja-jp/download/details.aspx?id=5555>) からパッケージをダウンロード
- ii. **vcredist_x86.exe** を実行し、表示されたインストールウィザードに従ってインストールを完了する。

4. Toboggan 基礎編

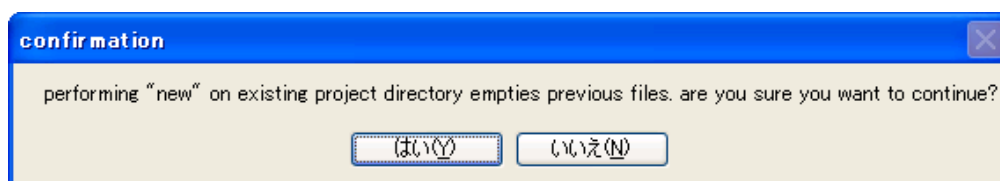
a. 新しくプロジェクトを作る



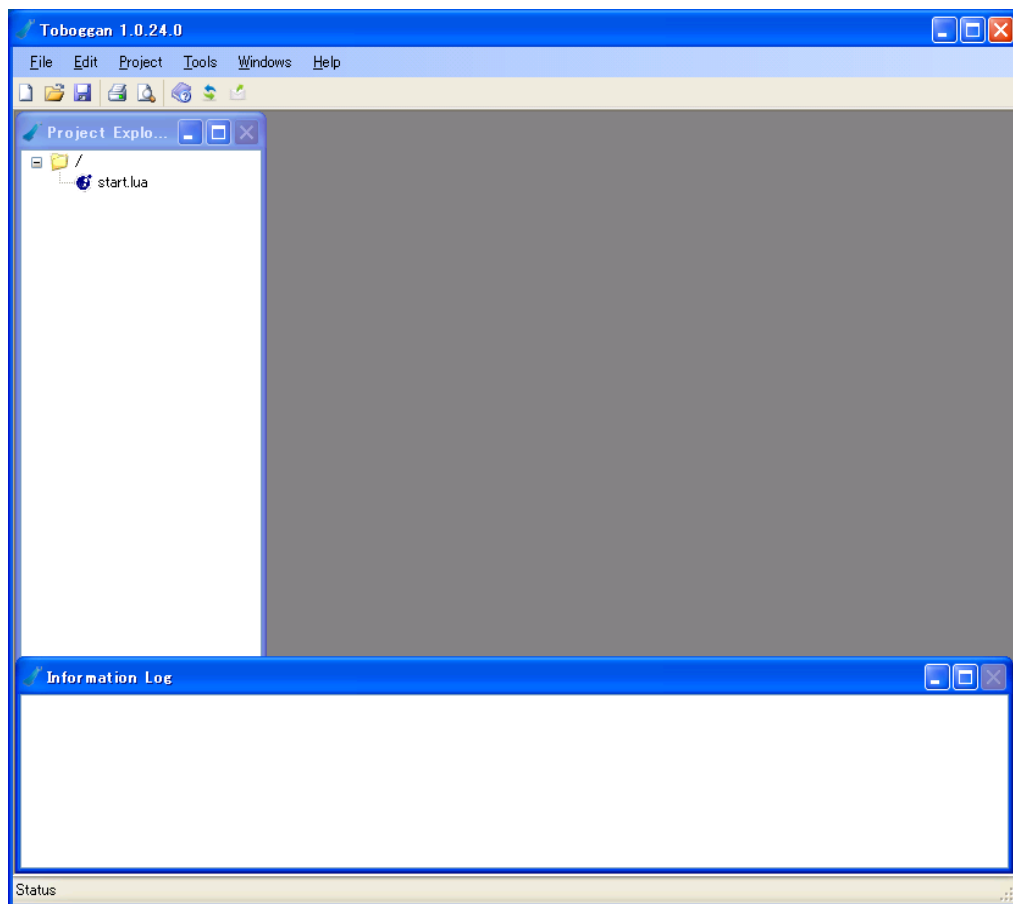
【新規プロジェクトの作成】
[New Project] をクリックする。



【プロジェクトディレクトリの設定】
プロジェクトを管理するディレクトリを選択し、[OK] ボタンをクリックする。
今回は、**C:\projects\sample** を作成し選択。



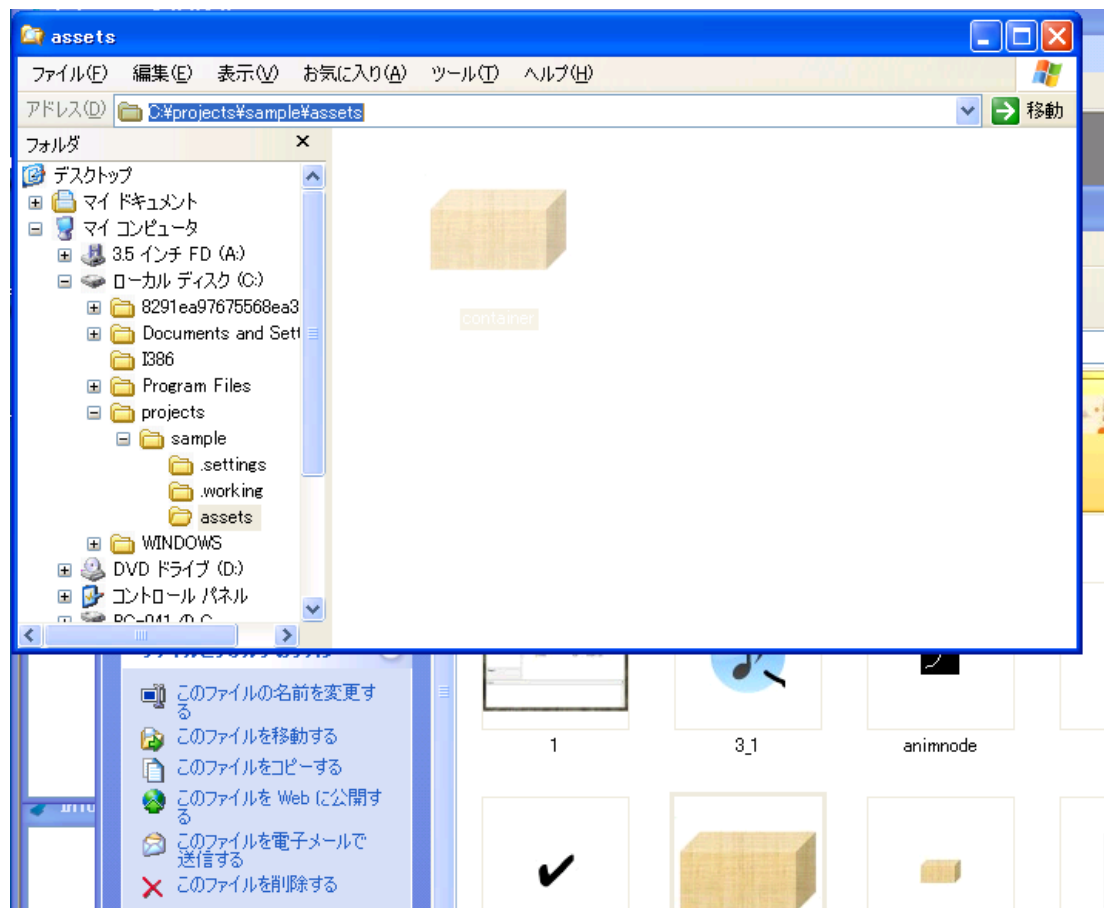
【プロジェクト作成の確認ダイアログ】
プロジェクトを新規に作成すると、選択されたディレクトリの内容が空になる事を注意するダイアログ。
ディレクトリが元々空なら特に問題は無いので、[はい] をクリックする。



【プロジェクト作成後の Toboggan】

プロジェクトの初期化時に作成された、start.lua ファイルのみが Project Explorer（左ペイン）に表示されている。

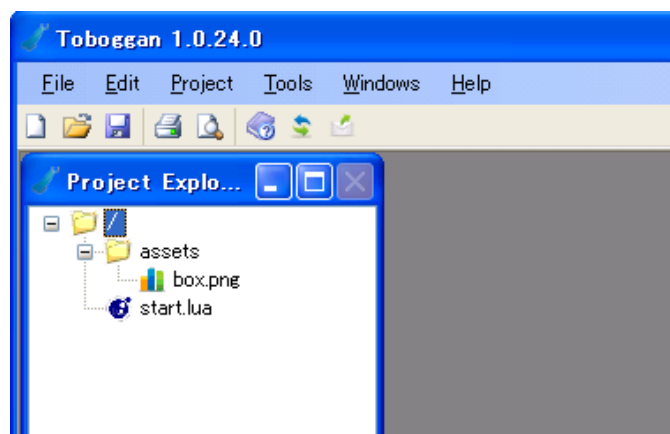
b. ファイルを配置する



【Windows エクスプローラで、画像ファイルの配置】

プロジェクトで使用するファイルをプロジェクトディレクトリに配置する。

上図では例として、**assets** ディレクトリを作成し、その下に画像ファイルをコピーしている

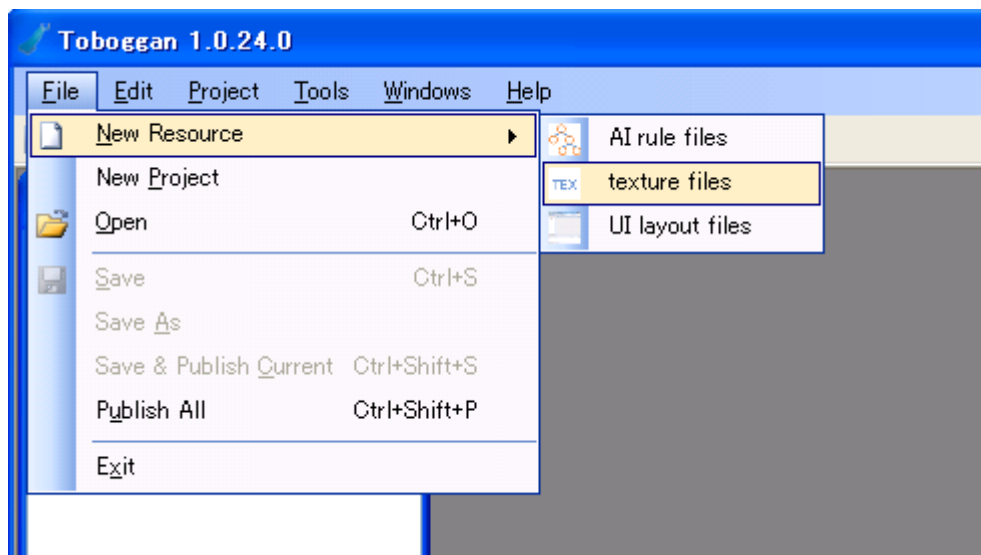


【Project Explorer での画像ファイルの表示】

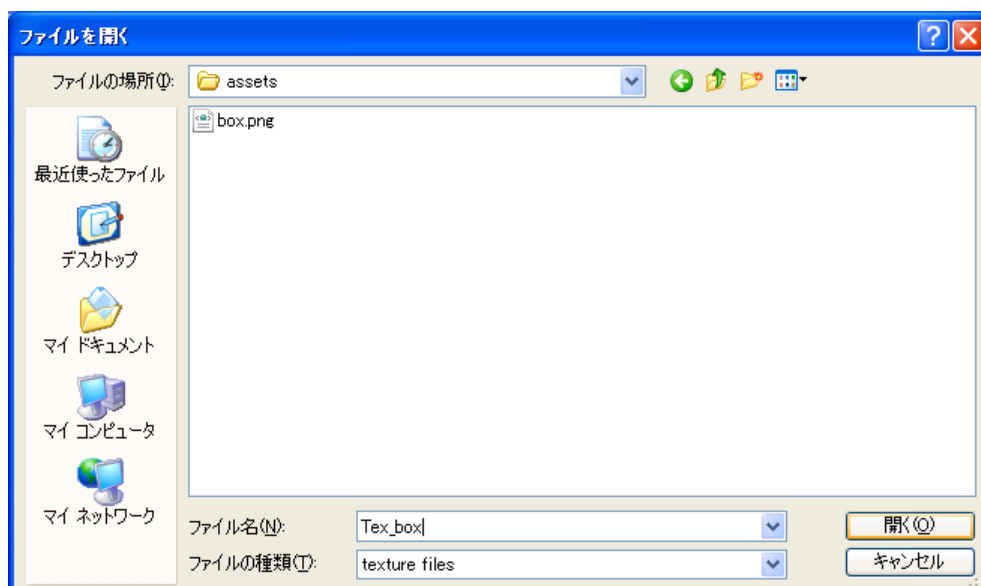
プロジェクトディレクトリに画像ファイルがコピーされた事は、直ちに Toboggan に通知・同期される。

Project Explorer を見ると、**assets\box.png** が表示されている事がわかる。

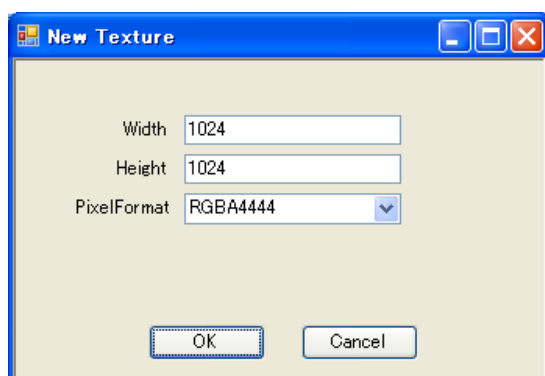
c. 新しくテクスチャ アセットを作る



【テクスチャ アセットの作成】
[New Resource] > [texture files] をクリックする。



【アセットのファイル名を設定】
作成するアセットのファイル名を予め入力し、[開く] ボタンをクリックする。
基本的には貼付するイメージ アセットファイルと同じディレクトリにテクスチャ アセットを作成するので、**assets\Tex_box** とした。

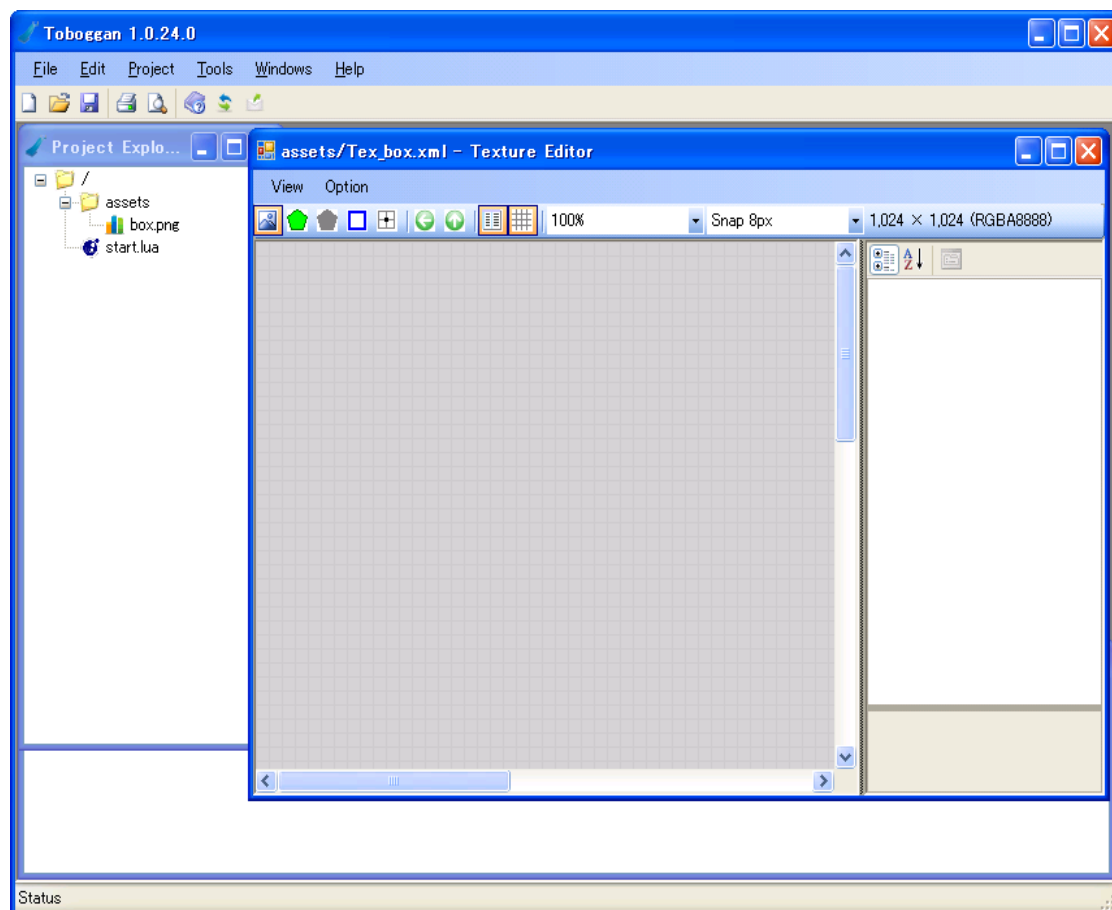


【テクスチャ サイズの設定】
テクスチャ サイズを **1024 x 1024**、PixelFormat を **RGBA4444** に指定して [OK] ボタンをクリックする。

Point!

2^n 2^n

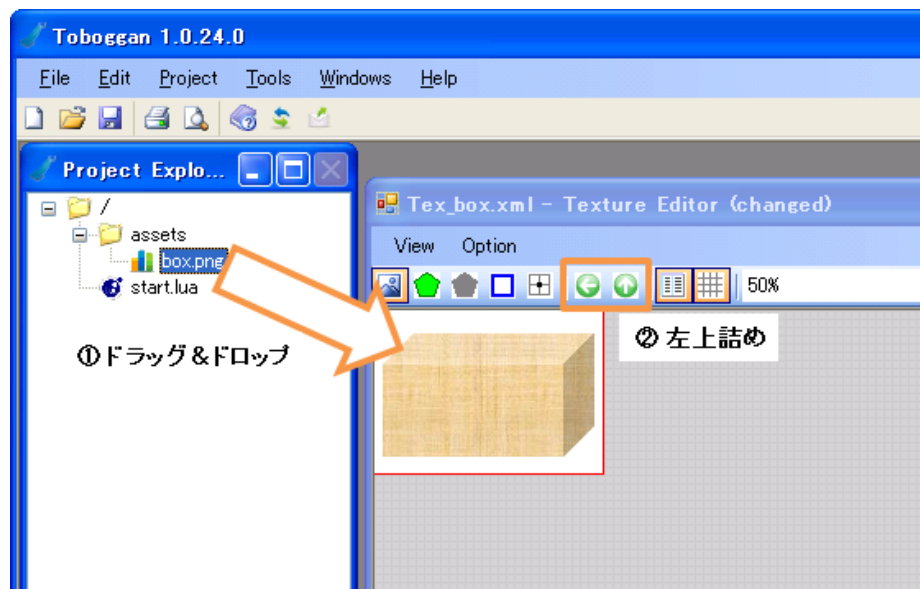
テクスチャサイズは幅・高さ共に2の何乗の倍であればどんな数値でも良いが、現在、ゲームエンジン側でテクスチャを効率的にロードする為には **1024** が適切であるという認識で統一されている為、ここでもそれに倣っている。




【Tex_box.xml を開いた Texture エディタ】

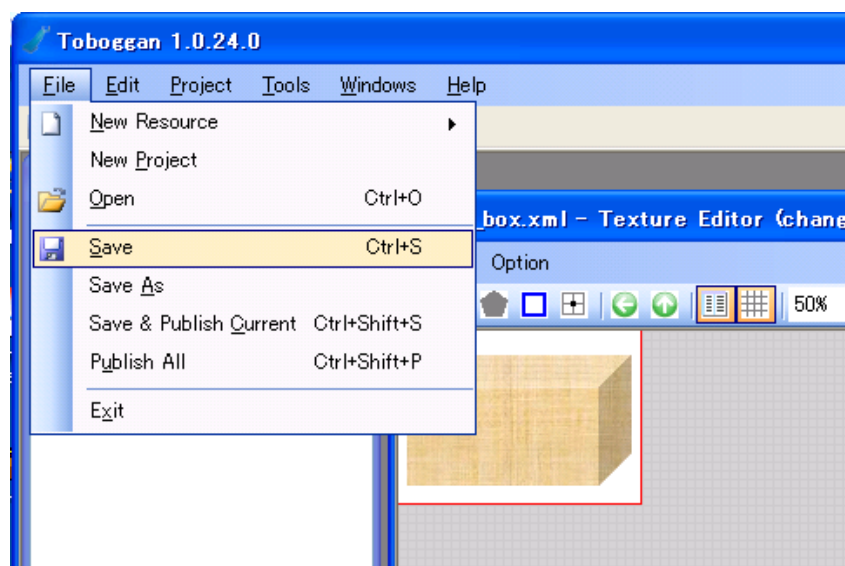
Tex_box テクスチャ アセットを Texture エディタで開いた様子。

新しいテクスチャの為まだ何も無いが、ここにイメージ アセットを貼り付けていく。



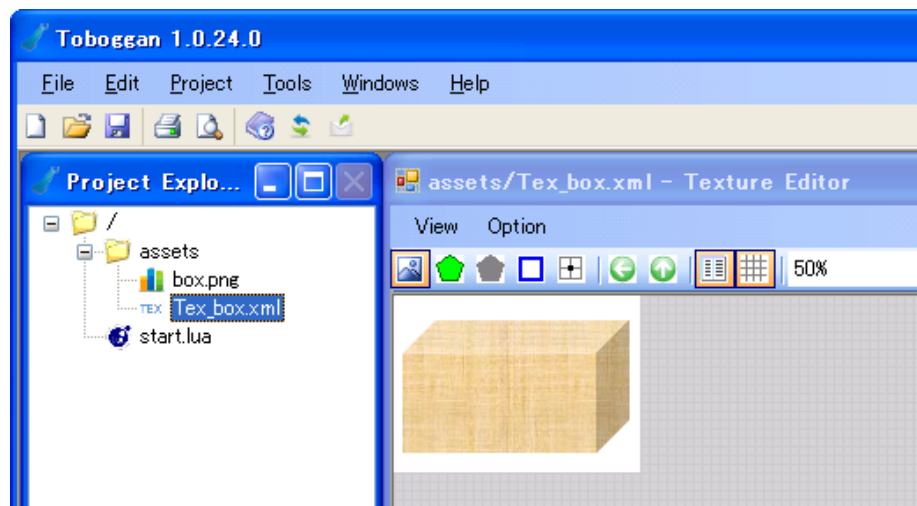
【イメージアセットの貼付】

- ① Project Explorer に表示されている **assets\box.png** をTexture エディタにドラッグ&ドロップして貼り付ける。
- ②  (Packing Space) ボタンをクリックして、貼付したイメージアセットを **TopLeft 詰め (パッキング)** にする。



【テクスチャアセットの保存】

Texture エディタの編集内容をTex_box アセットに保存するには **[Save]** をクリックする。

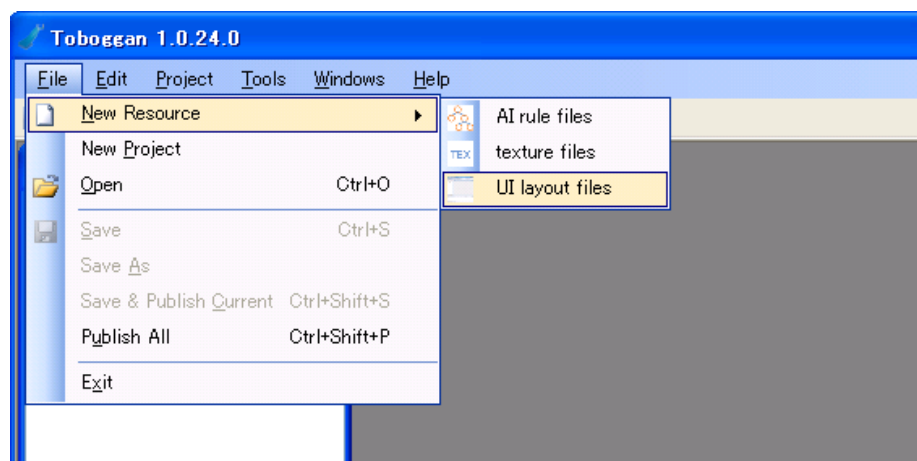


【Tex_box アセット保存後の Project Explorer】
assets\Tex_box.xml ファイルが保存・表示されている事がわかる。

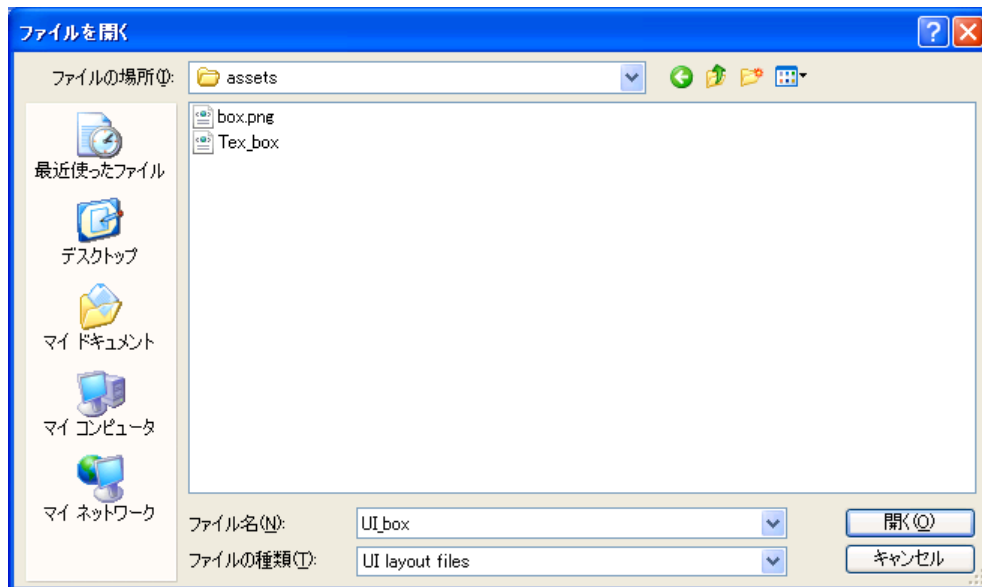
Point!

Ctrl+S ショートカットキーを押下する事で、Texture エディタや UI エディタで編集集中の内容を保存する事が出来る。

d. 新しく UI アセットを作る

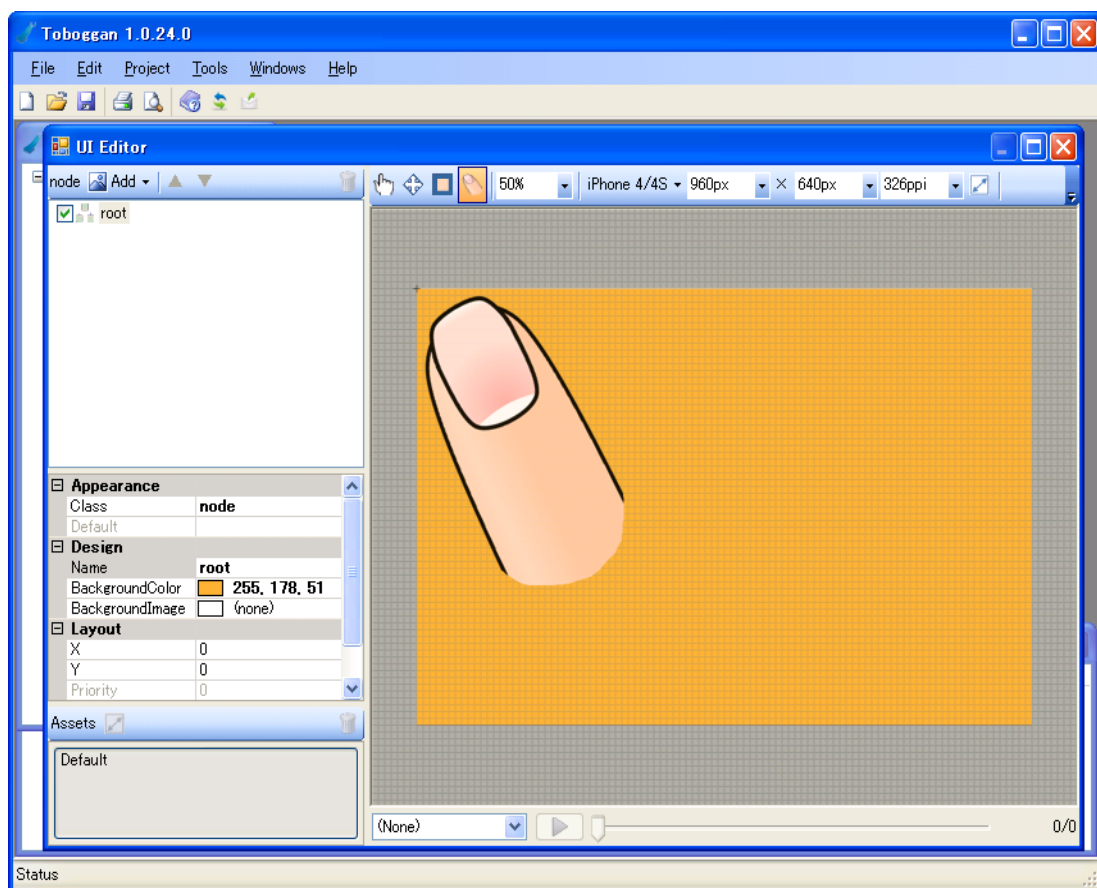


【UI アセットの作成】
[New Resource] > [UI layout files] をクリックする。




【アセットのファイル名を設定】

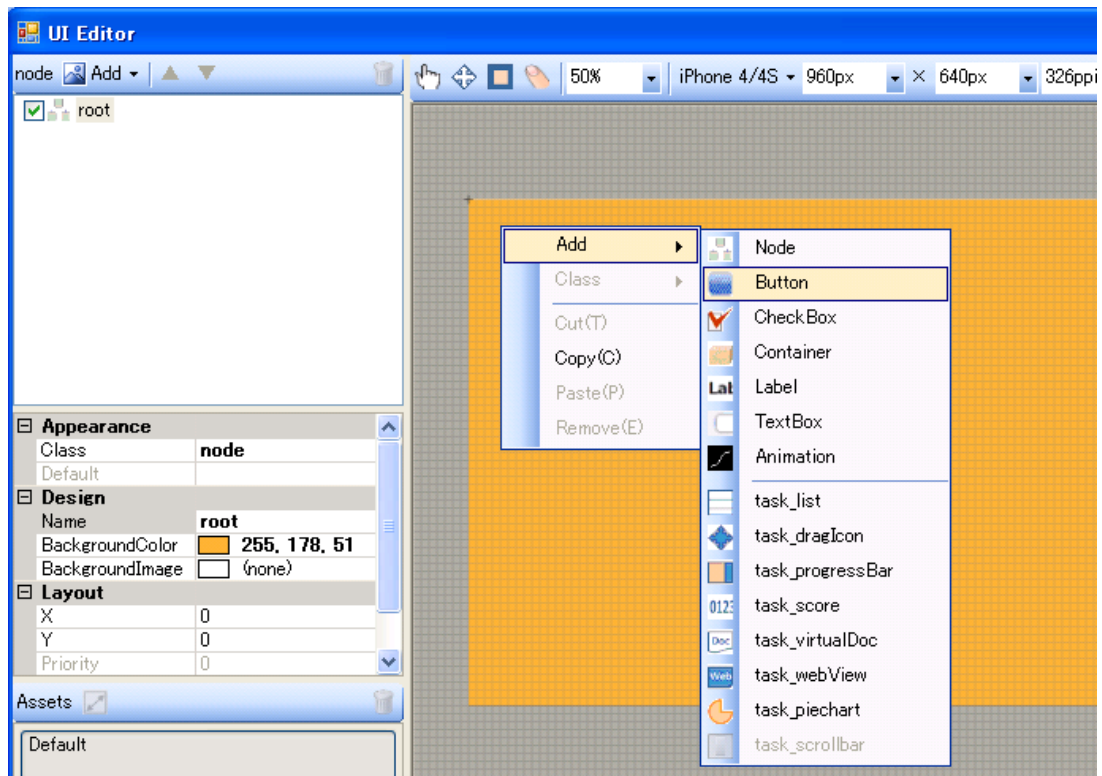
作成するアセットのファイル名を予め入力し、[開く] ボタンをクリックする。
基本的には貼付するアセットファイルと同じディレクトリに作成するので、**assets\UI_box** として。




【UI_box.xml を開いた UI エディタ】

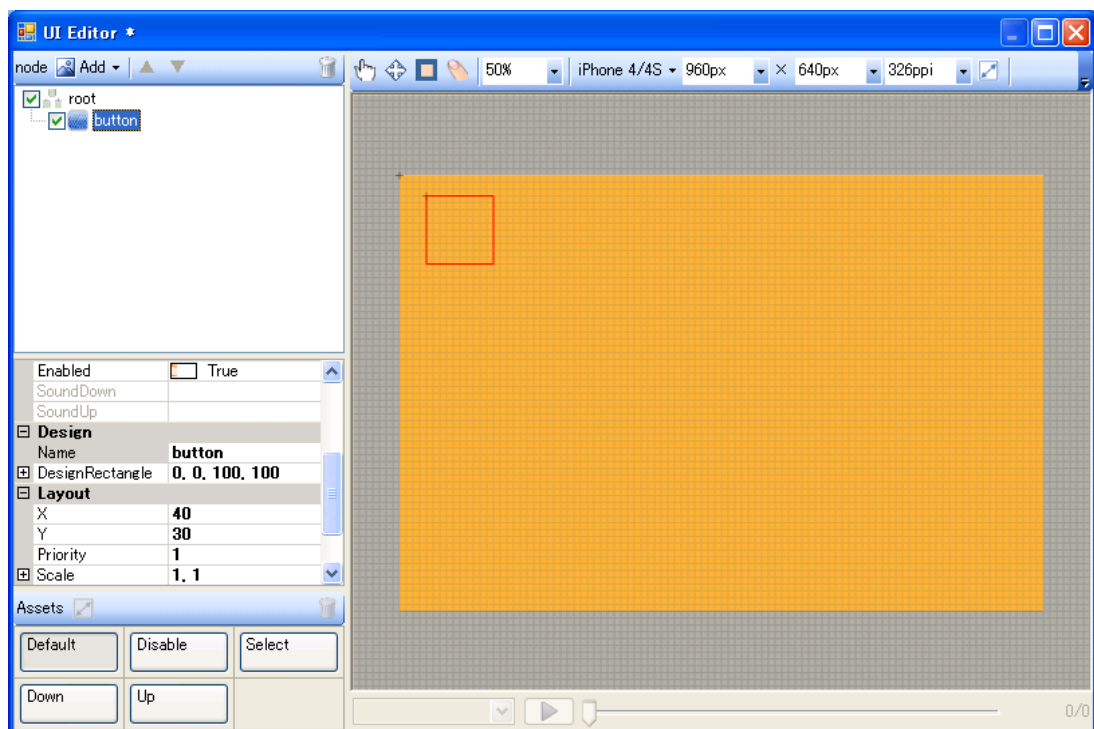
UI_box UI アセットを UI エディタで開いた様子。

新しい UI の為、まだルートノードのみ（指先イメージ  は唯の定規）だが、ここに各種ノードやイメージアセットを貼り付けていく。



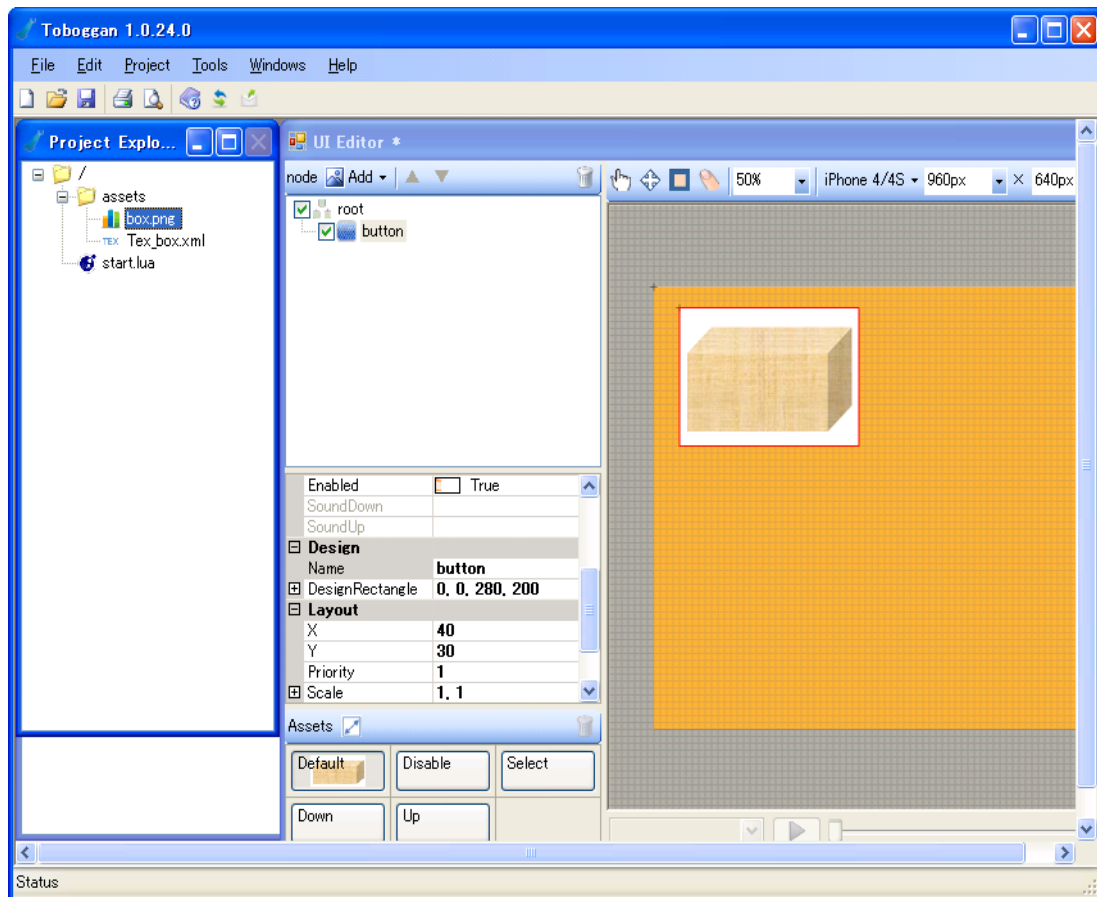
【ノードの作成】

button ノードを作成してみる（ちなみに、指先イメージは今は不要なので非表示にした）。キャンバス（右ペイン）で右クリックする事で表示されるコンテキストメニューから、**[Add] > [Button]** をクリックする。




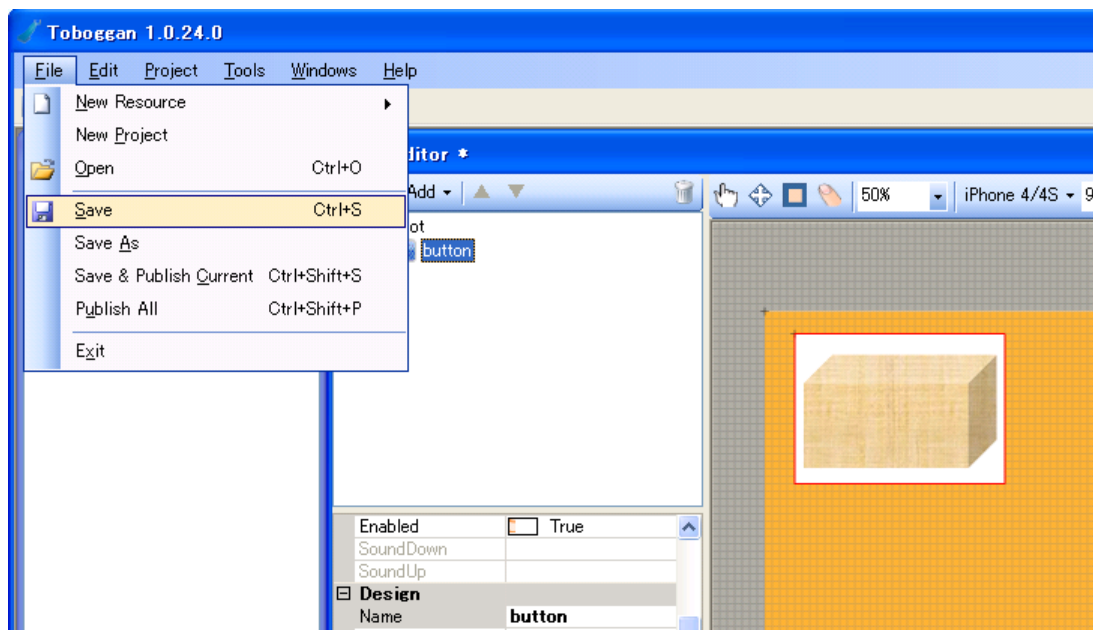
【ノードの選択】

作成した button ノードをクリックして選択する。



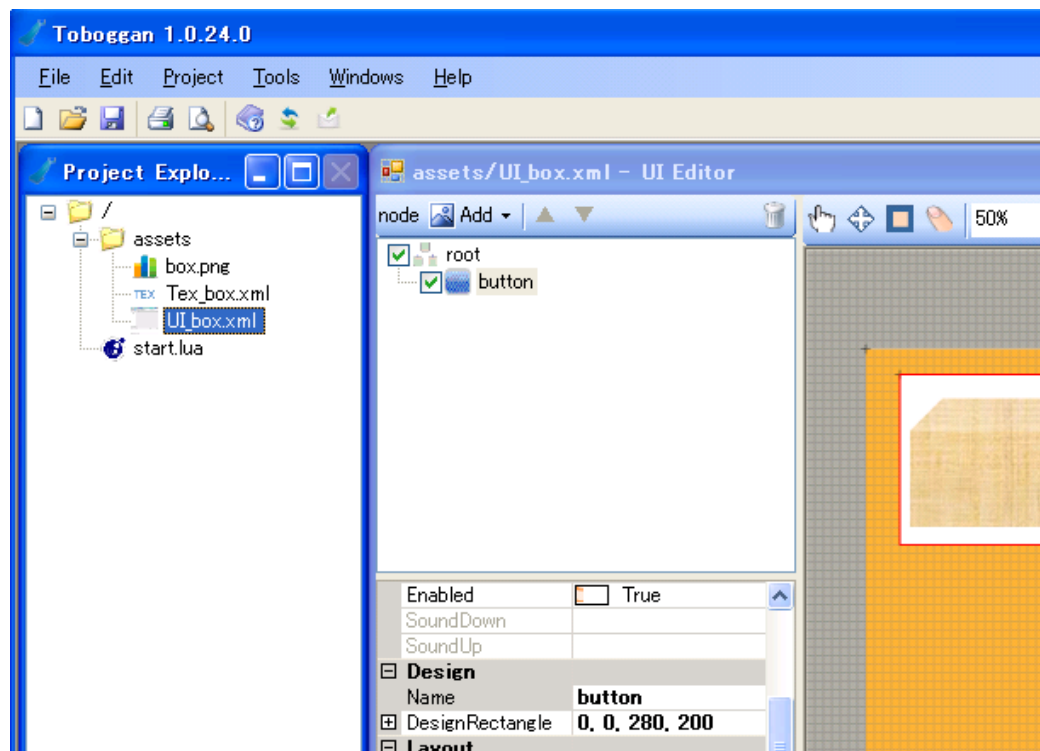
【イメージアセットの貼付】

- ① Project Explorer に表示されている **assets\box.png** を UI エディタの Assets パネル（左下ペイン）の **Default** にドラッグ & ドロップする。
- ② Assets パネルヘッダーの  ボタンをクリックし、ノードのサイズをイメージに合わせる。



【UI アセットの保存】

UI エディタの編集内容を UI_box アセットに保存するには **[Save]** をクリックする。

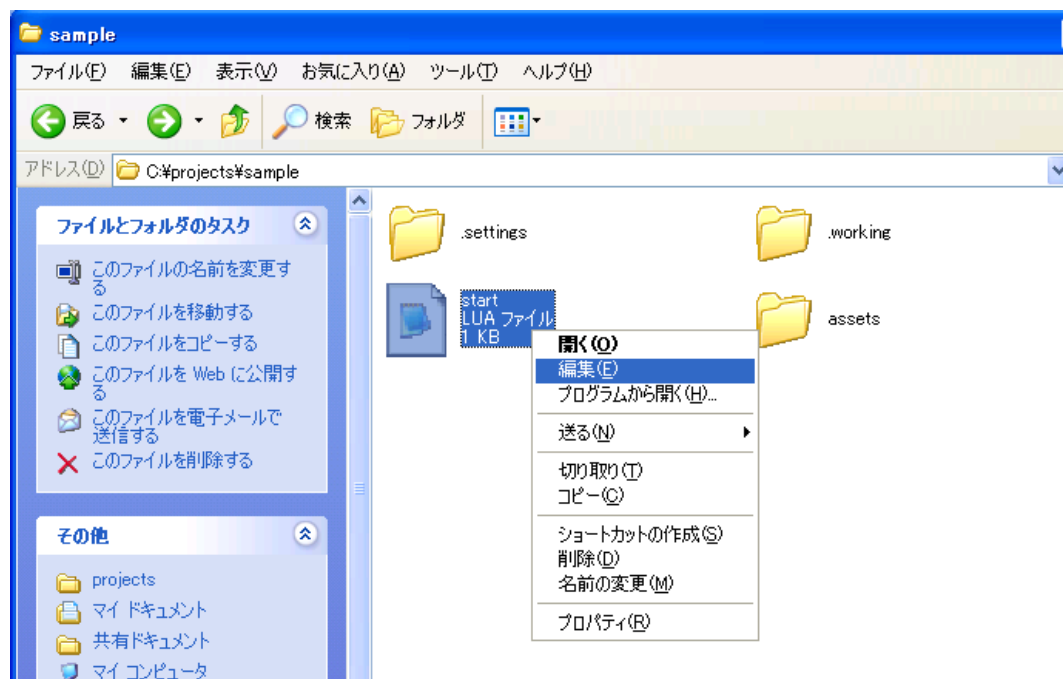


【UI_box アセット保存後の Project Explorer】
assets\UI_box.xml ファイルが保存・表示されている事がわかる。

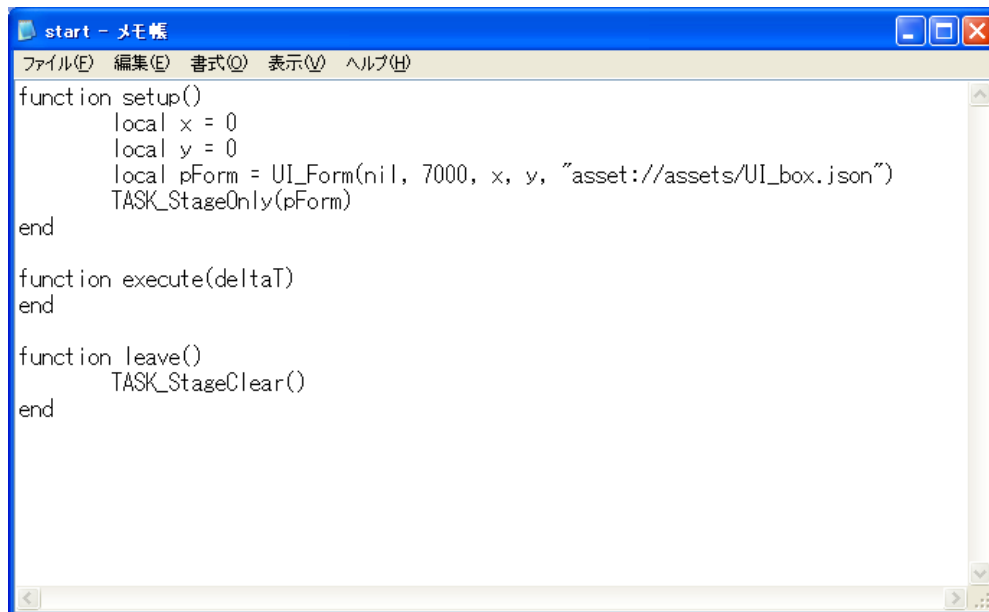
e. start.lua の編集

Notice!

この項では、UI_box/Tex_box アセットをゲームエンジンにロードさせる為のスクリプトを start.lua に記述しますが、lua については本マニュアルの範囲を超えた内容ですので、詳細は省きます。



【start.lua をテキストエディタで開く】
Windows エクスプローラーから start.lua を右クリックし、コンテキストメニューから **【編集】** をクリックしてテキストエディタで開く。



【start.lua にスクリプトを記述】
UI_box アセットをロードする為のスクリプトを記述し、保存する。

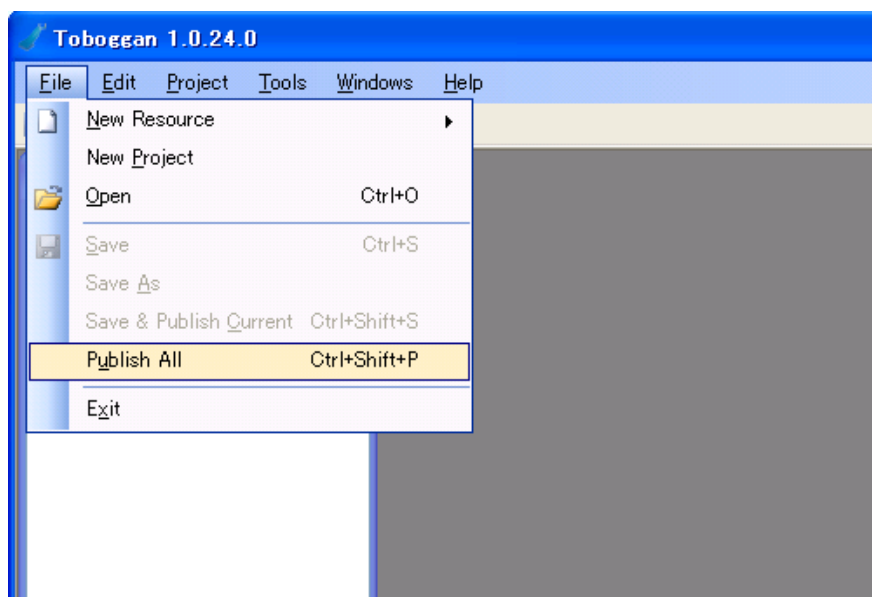
```

-- assets/UI_box アセットをロードする為のスクリプト
function setup()
    local x = 0
    local y = 0
    local pForm = UI_Form(nil, 7000, x, y, "asset://assets/UI_box.json")
    TASK_StageOnly(pForm)
end

-- 毎フレーム実行される関数
function execute(deltaT)
end

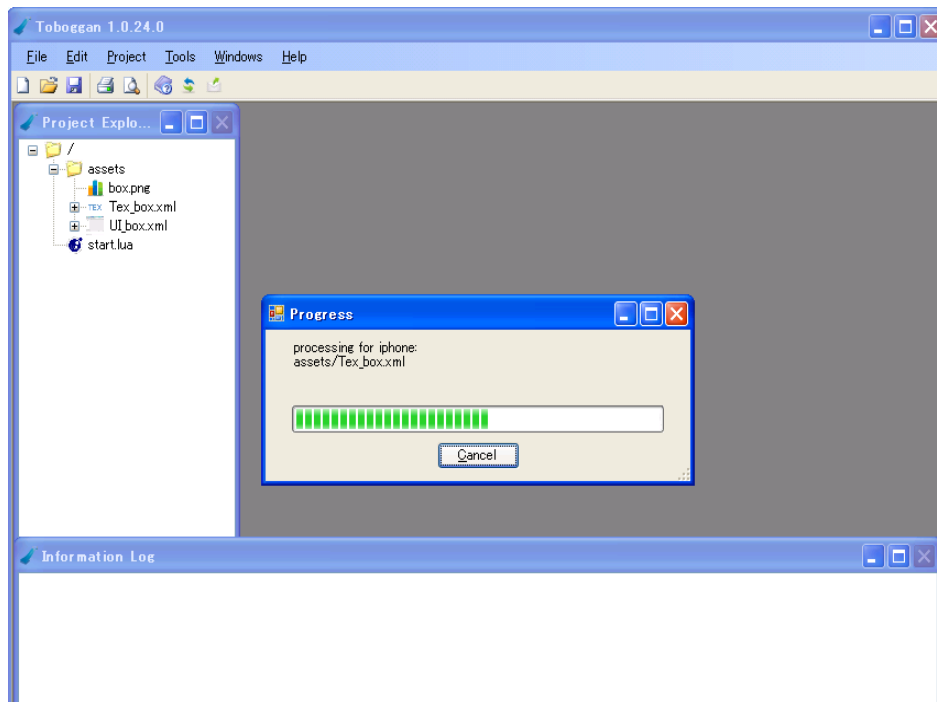
```

f. プロジェクトをパブリッシュ¹する

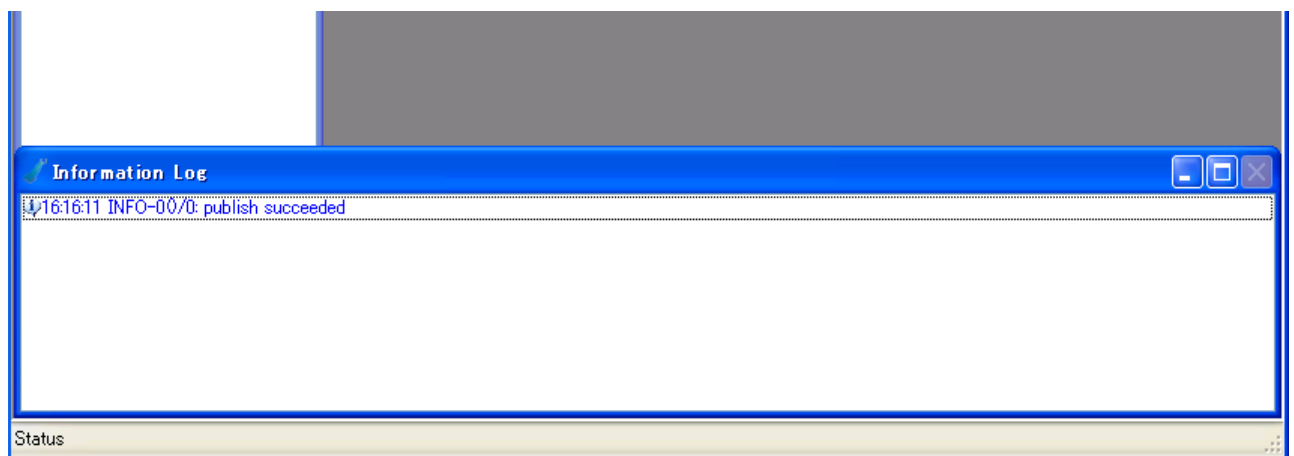


¹パブリッシュ: アセットをゲームエンジンがロード可能な形式へ変換する事。

【プロジェクトのパブリッシュ】
[File] > [Publish All] をクリックする。

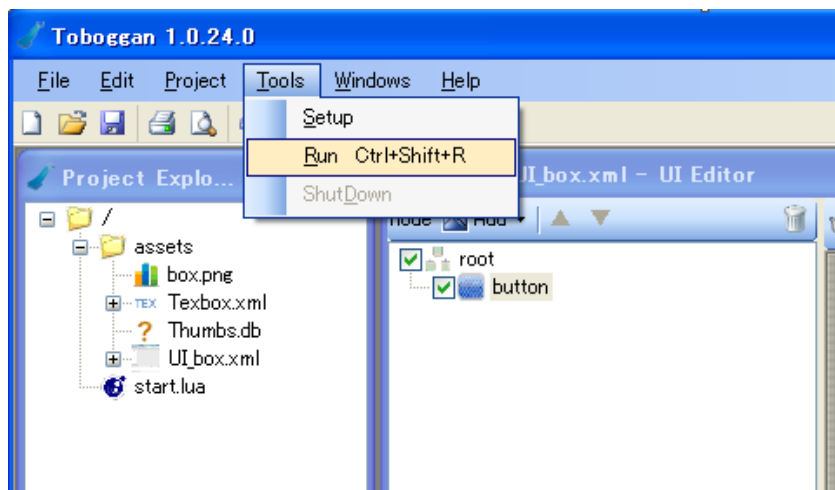


【パブリッシュの進捗ダイアログ】
進捗バーが100%になれば、パブリッシュは完了。
プロジェクトの規模によっては数分掛かる事もある。

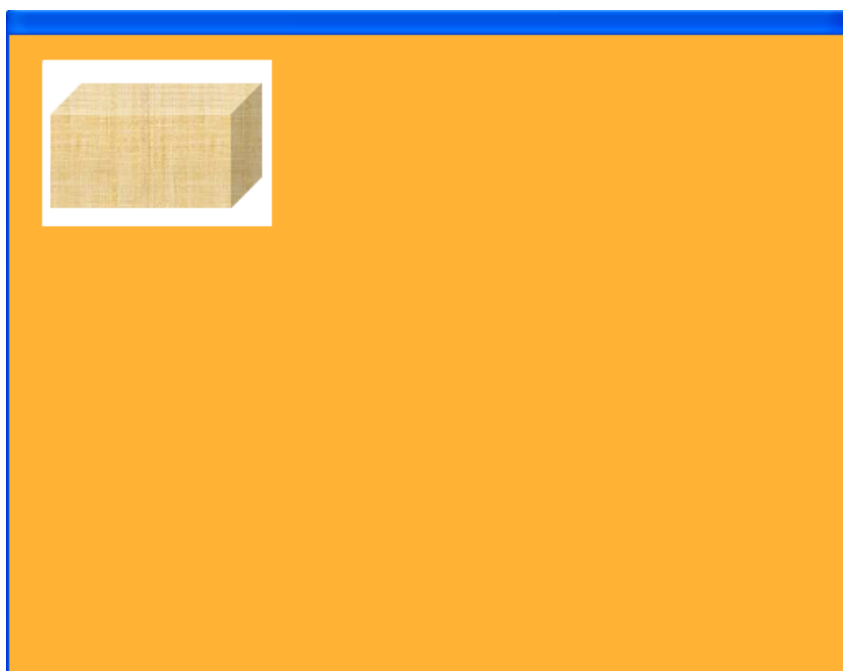


【パブリッシュの完了】
パブリッシュに成功すれば、[Information Log] に *publish succeeded* と表示される。
失敗した場合には、関連するエラーログと *publish failed (N error)* というメッセージが表示される
ので、適宜アセットを修正して再パブリッシュを行う。

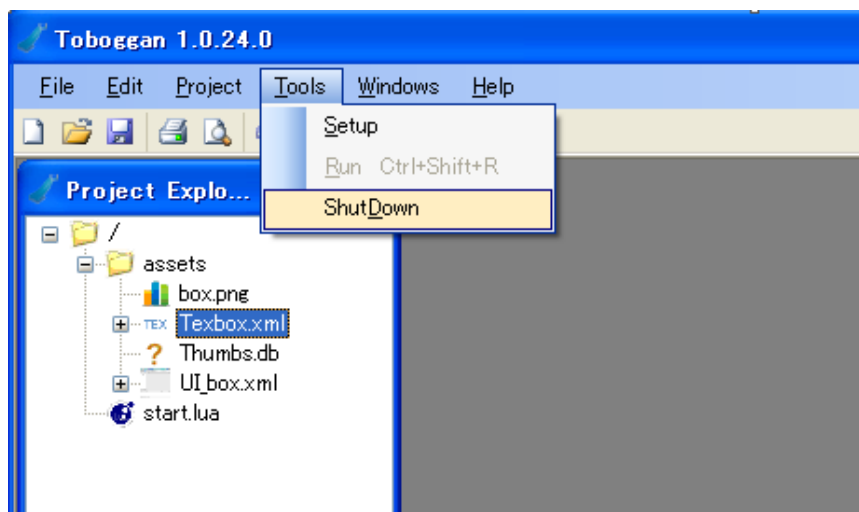
g. パブリッシュしたプロジェクトを使ってゲームを実行する



【ゲームエンジンの起動】
[Tools] > [Run] をクリックして、ゲームエンジンを起動します。
 この時、前項でパブリッシュしたデータがエンジンにロードされます。



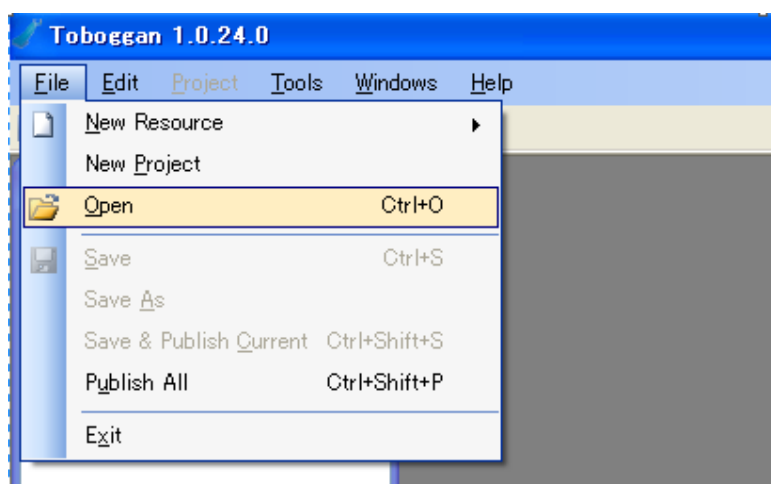
【ゲーム画面】
UI_box UI アセットの内容が表示される事が確認できます。



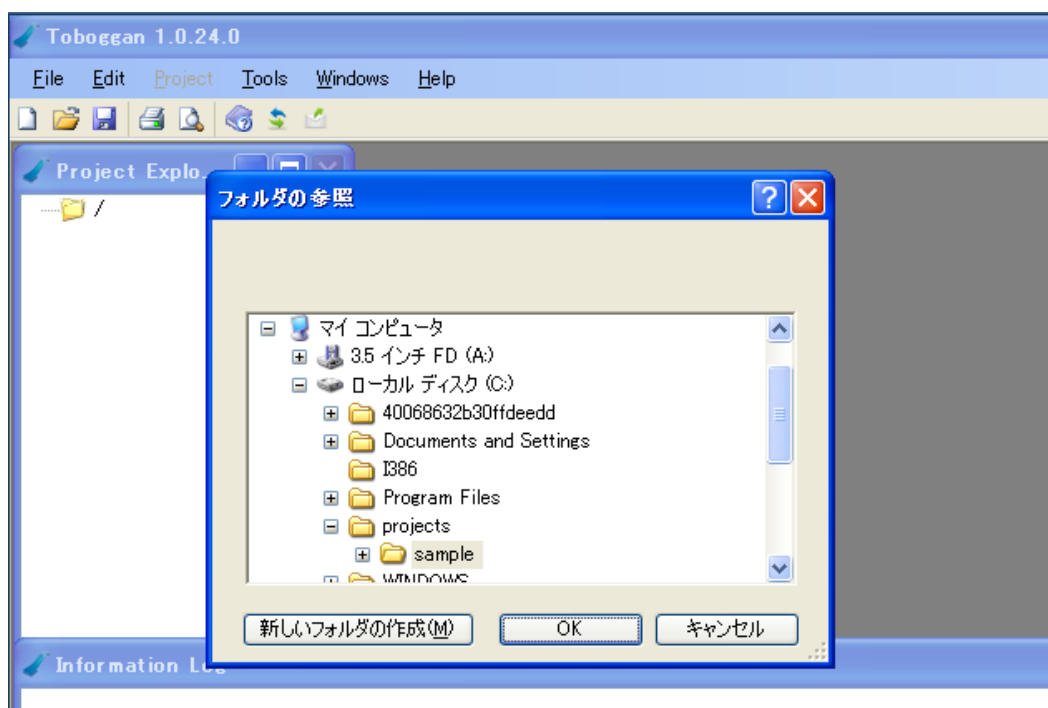
【ゲームエンジンの停止】

[Tools] > [ShutDown] をクリックして、ゲームエンジンを停止します。

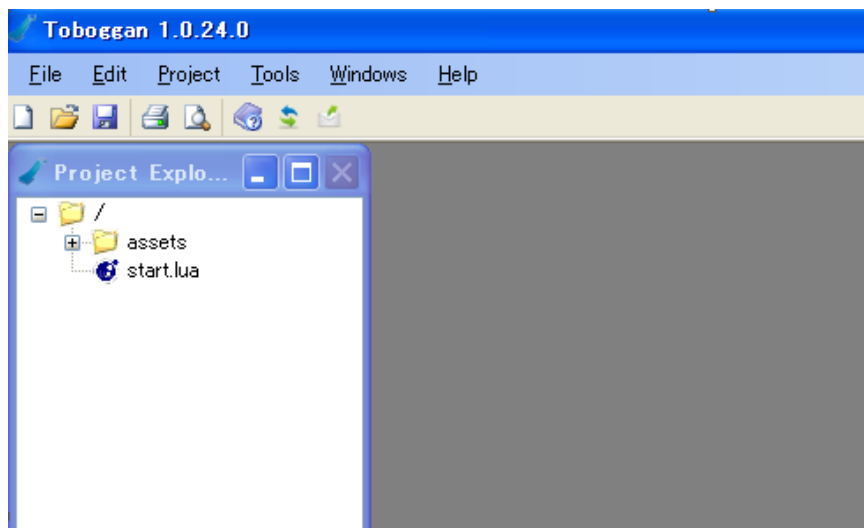
h. 既存のプロジェクトを開く



【既存プロジェクトを開く】
[File] > [Open] をクリックすると、ディレクトリ参照ダイアログが表示される。

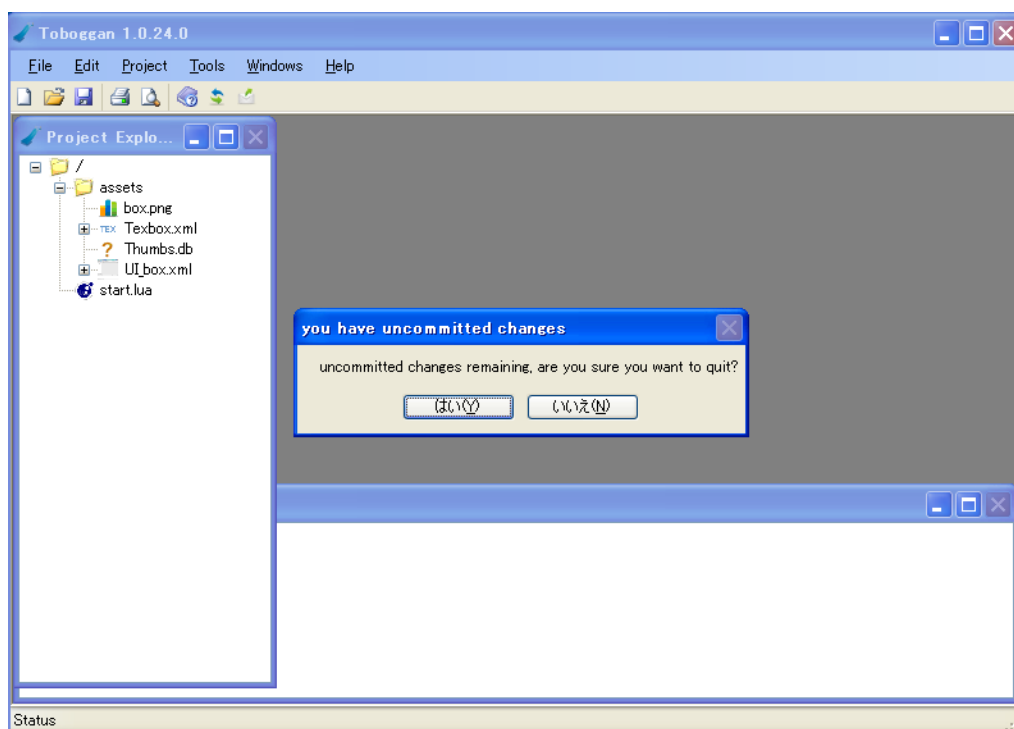


【プロジェクトディレクトリの参照ダイアログ】
既存のプロジェクトディレクトリを選択し [OK] をクリックする。
ここでは、先に作成した C:\projects\sample\ プロジェクトを開いてみる。



【選択したプロジェクトが展開された Project Explorer】
プロジェクトディレクトリの内容が、Project Explorer に展開されている様子が確認できる。

i. Toboggan を終了する



【Toboggan 終了時に表示されるダイアログ】

✖をクリックするなどしてアプリケーションを閉じようとする、上図のようなダイアログが表示される事がある。
変更したアセットファイルをコミット²せずに終了しようとする、と表示されるダイアログで、SVN³を利用していない、又はコミットする必要が無い場合には【はい】をクリックして、アプリケーションを終了する。

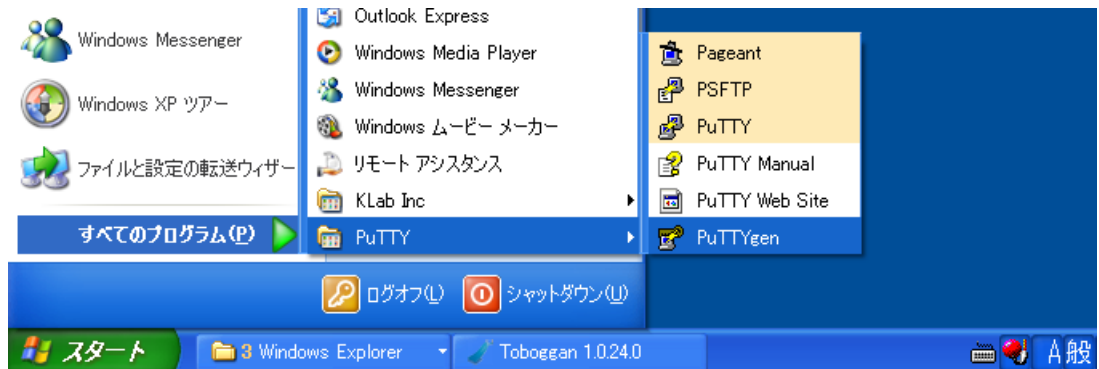
² コミット : SVN リポジトリに変更したファイルを保存する事。この時、変更履歴が自動的に記録される。

³ SVN : バージョン管理ソフトの一つ Subversion の略称。ファイルの変更履歴（バージョン）を管理してくれる。

5. Toboggan SVN編 [オプション]

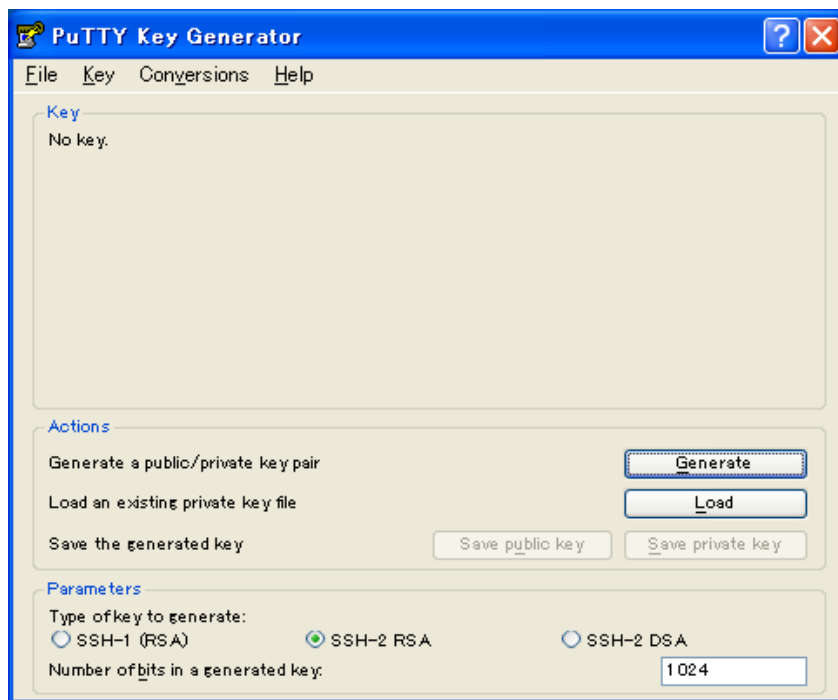
a. リポジトリの認証設定

i. PuTTYgen で秘密鍵・公開鍵の作成



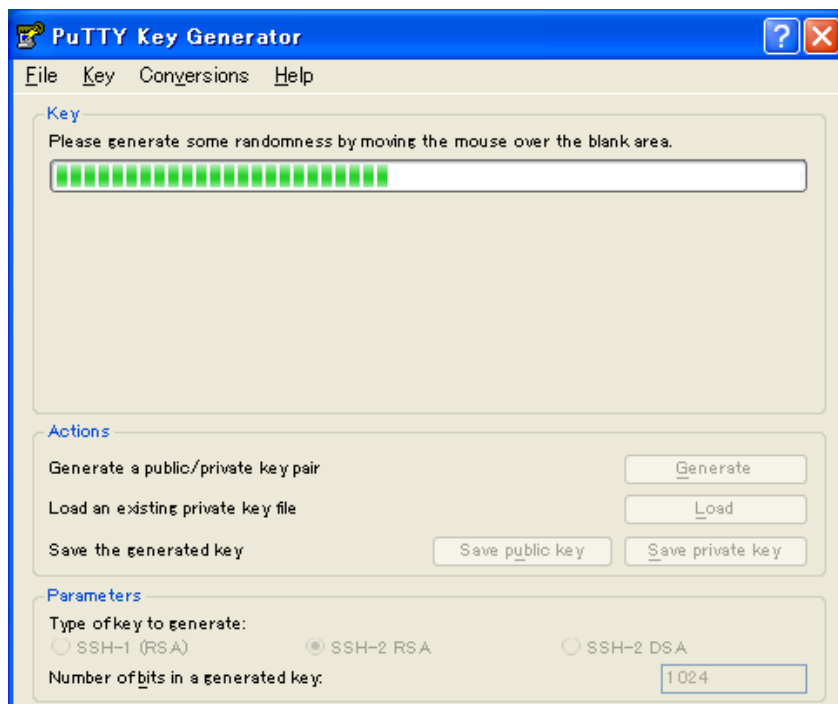
【PuTTYgen の起動】

[スタート > すべてのプログラム > PuTTY > **PuTTYgen**] をクリックして、PuTTYgen を起動。



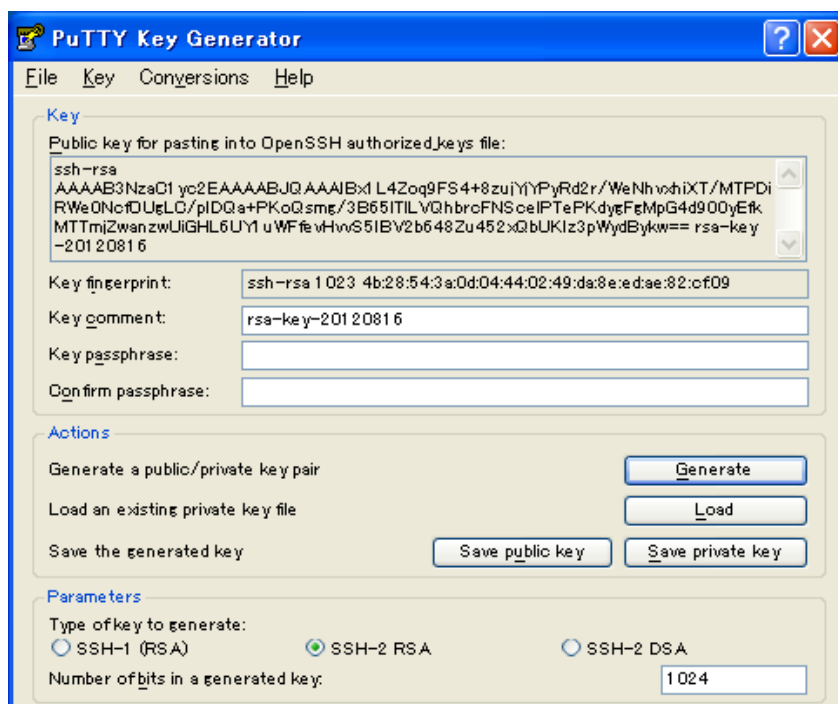
【PuTTYgen で公開／秘密鍵の作成】

[Generate] ボタンをクリックして、鍵の作成を開始。



【鍵の生成中】

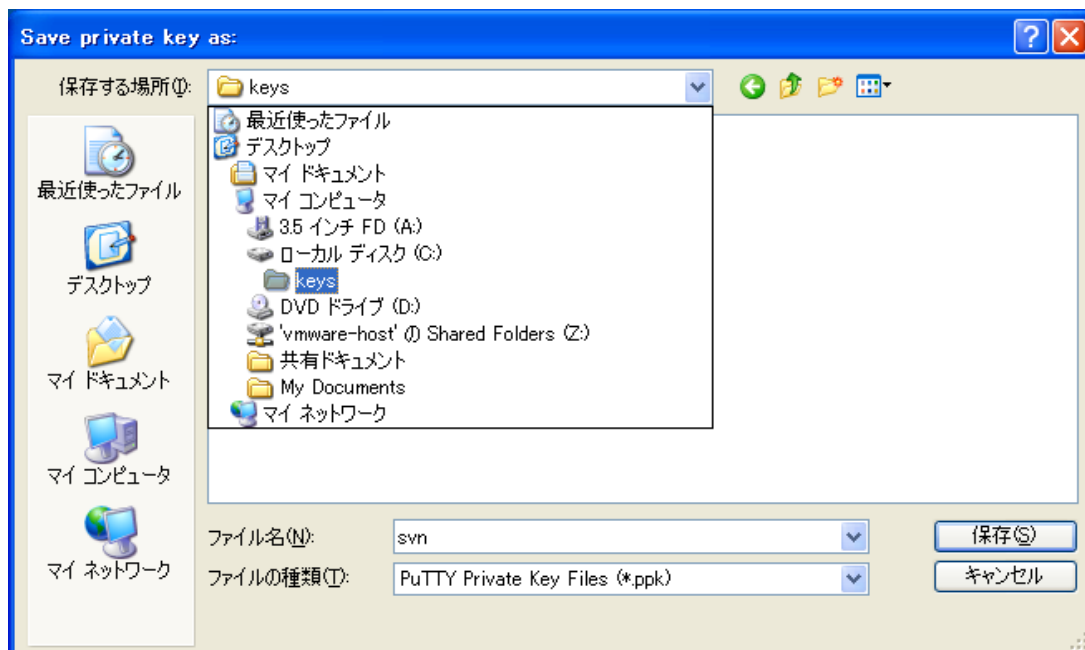
Key ペイン（進捗バー周辺の囲い）内でマウスをランダムに動かし、鍵の生成を促す。進捗バーが最大になると、鍵の生成が完了する。



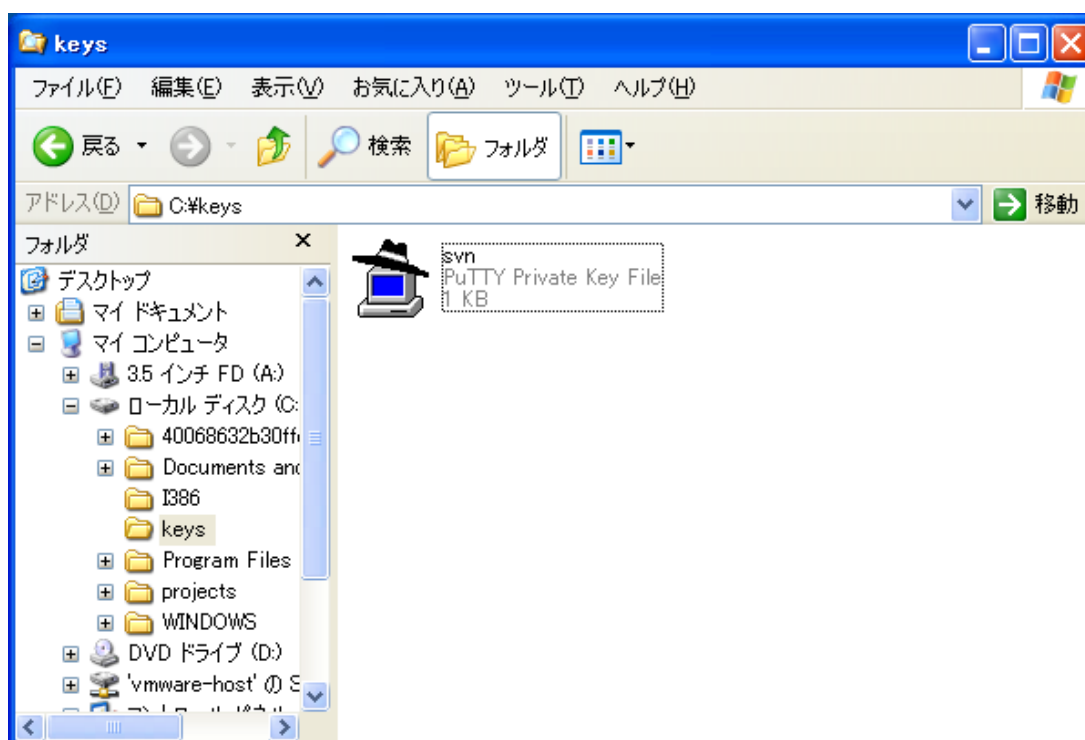
【生成された鍵】

鍵の生成が完了すると、上部のテキストボックスに公開鍵が表示されるので、メモしておく（[公開鍵をリポジトリサーバに登録申請](#) で使う）。

秘密鍵は、[Key passphrase] と [Confirm passphrase] にパスワードを入力して [Save private key] をクリックする。



【秘密鍵の保存ダイアログ】
 保存ダイアログが表示されるので、秘密鍵ファイルを保存する場所を選択する。
 上図では、C:\keys\svn.ppk に保存している。



【秘密鍵ファイルの確認】
 秘密鍵ファイルが確かに保存されているか、Windows Explorer 等で確認する。

ii. 秘密鍵を Pageant に登録



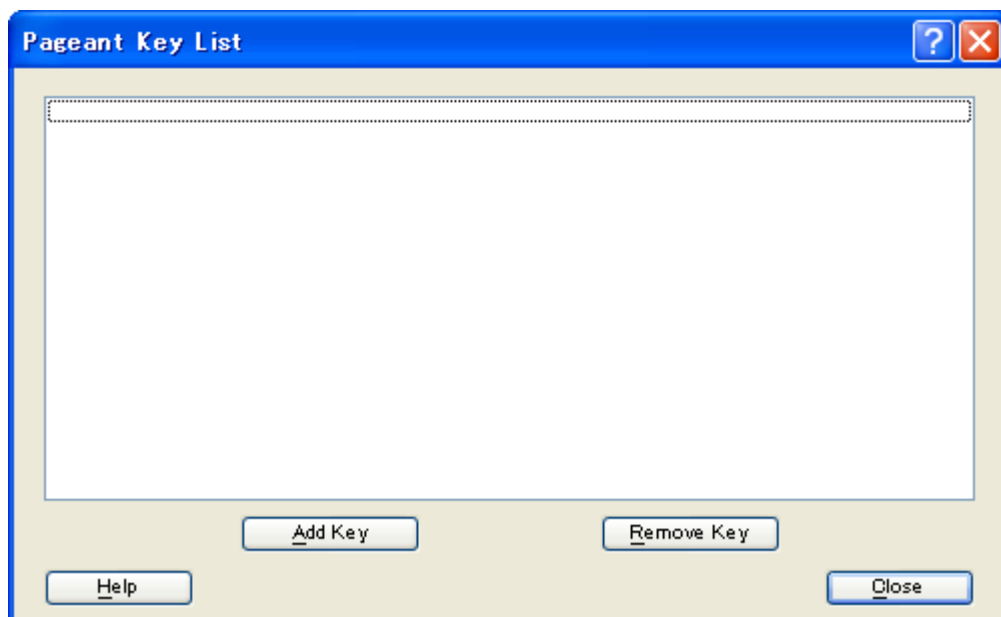
【Pageant の起動】

[スタート > すべてのプログラム > PuTTY > **Pageant**] をクリックして、Pageant を起動。



【タスクバーに潜む Pageant】

タスクバーに起動した Pageant があるので、これをクリックする。

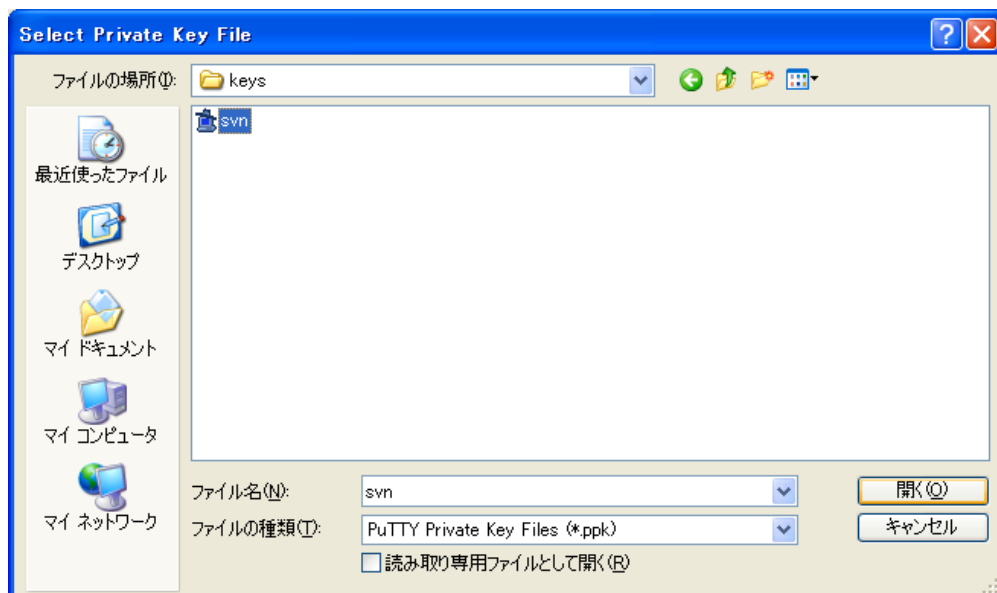


【Pageant のキーリストダイアログ】

秘密鍵の管理ダイアログが表示される。

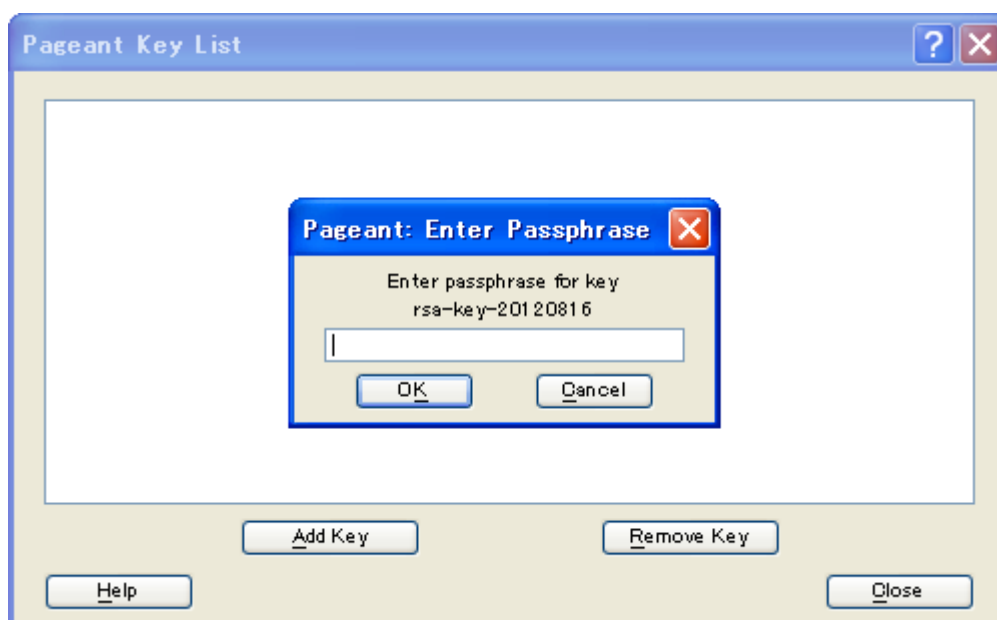
上記はまだ一つも鍵が登録されていないので、一覧は空欄となっている。

[Add Key] ボタンをクリックして、秘密鍵の登録処理を進める。



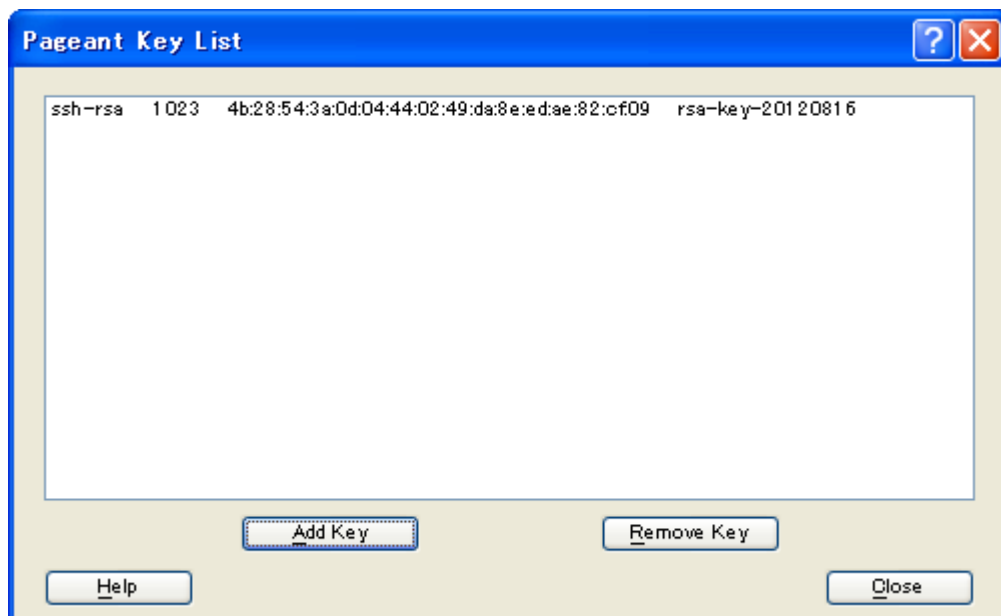
【秘密鍵の選択ダイアログ】

登録する秘密鍵のファイルを選択して、**[開く]** をクリックする。



【秘密鍵のパスフレーズ入力ダイアログ】

PuTTYgen で秘密鍵を生成する際に入力したパスフレーズを入力して、**[OK]** をクリックする。



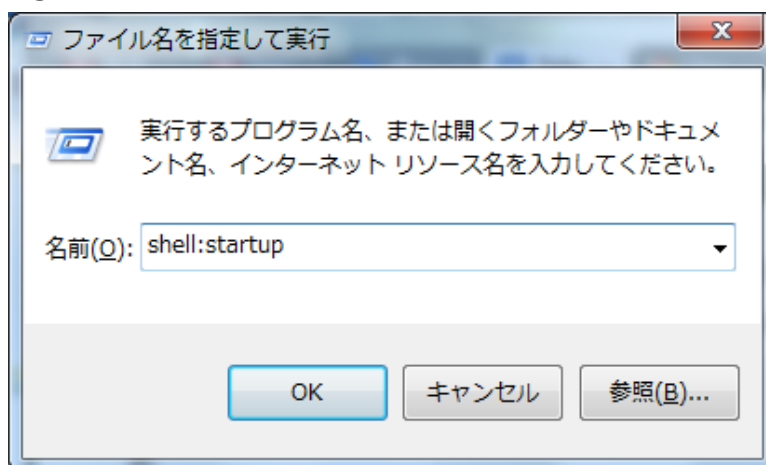
【Pageant に登録された秘密鍵】

パスフレーズに問題が無ければ、上記のように秘密鍵が登録されている事が確認できる。確認したら、**[Close]** ボタンをクリックしてダイアログを閉じる。

iii. 公開鍵をリポジトリサーバに登録申請

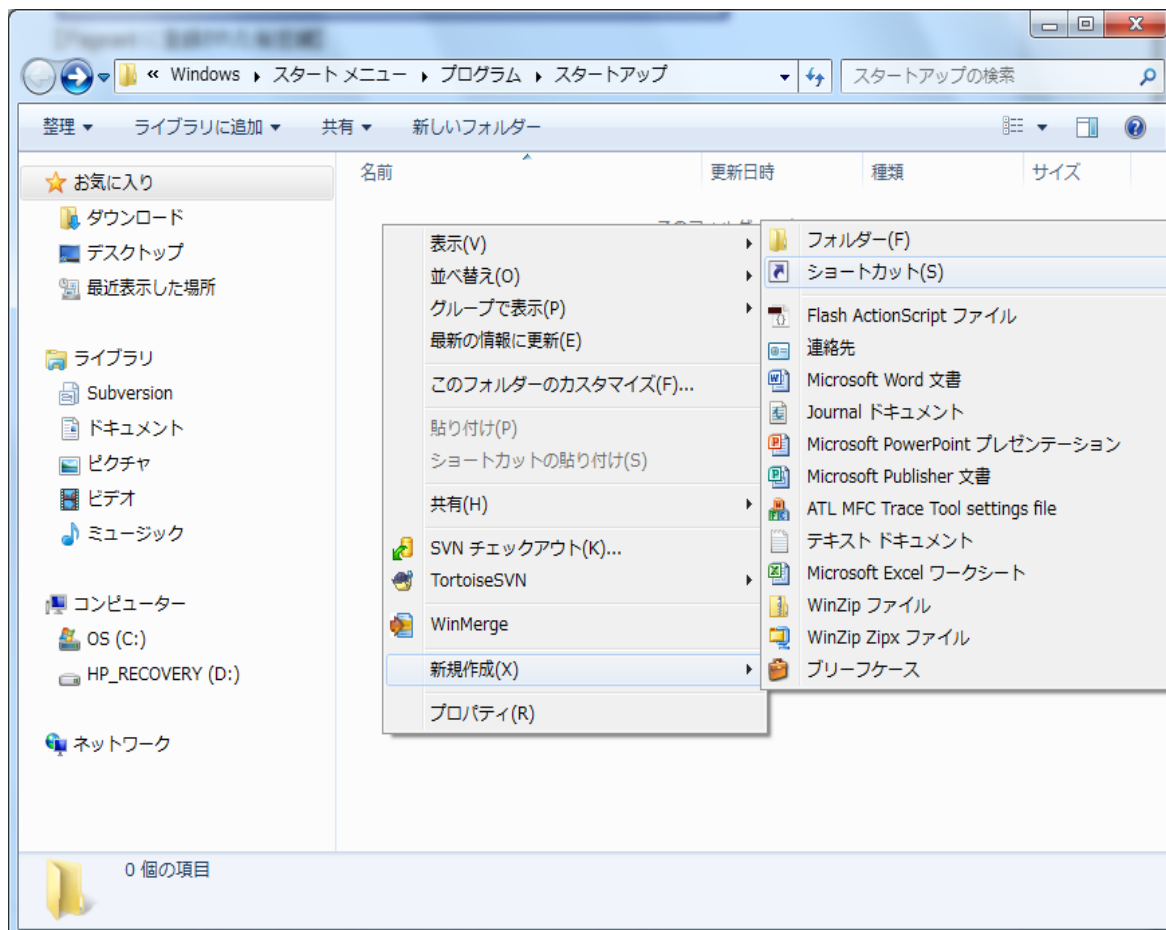
SVN リポジトリサーバへの公開鍵の登録は、規定の管理権限を保有する管理者のみが行える為、一般ユーザーは登録申請を行う必要がある。
詳細は、上長に確認。

iv. Pageant の常駐化

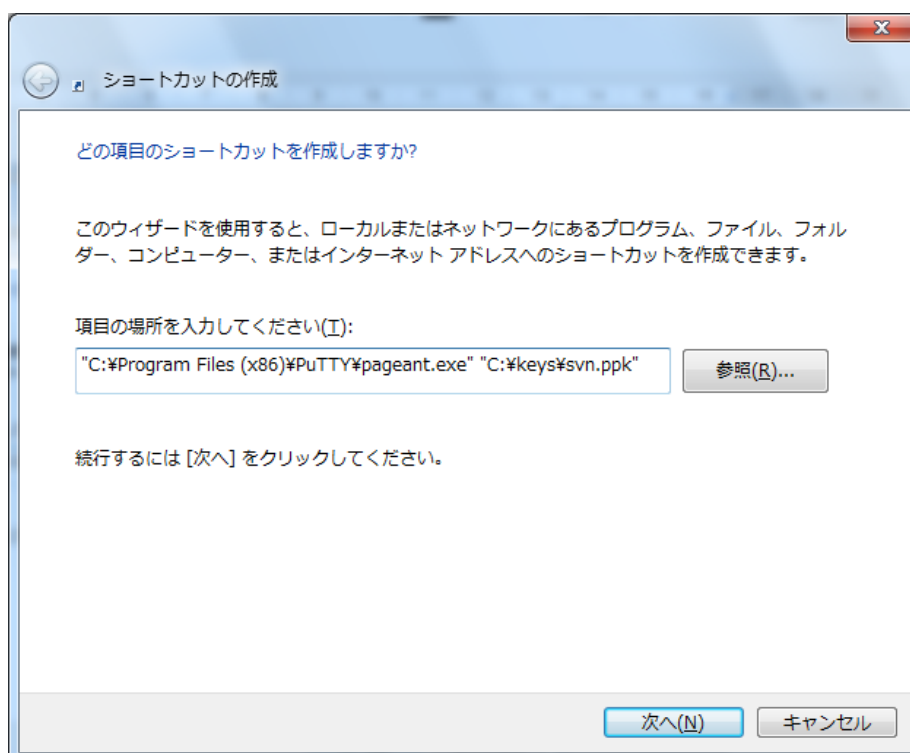


【ファイル名を指定して実行ダイアログ】

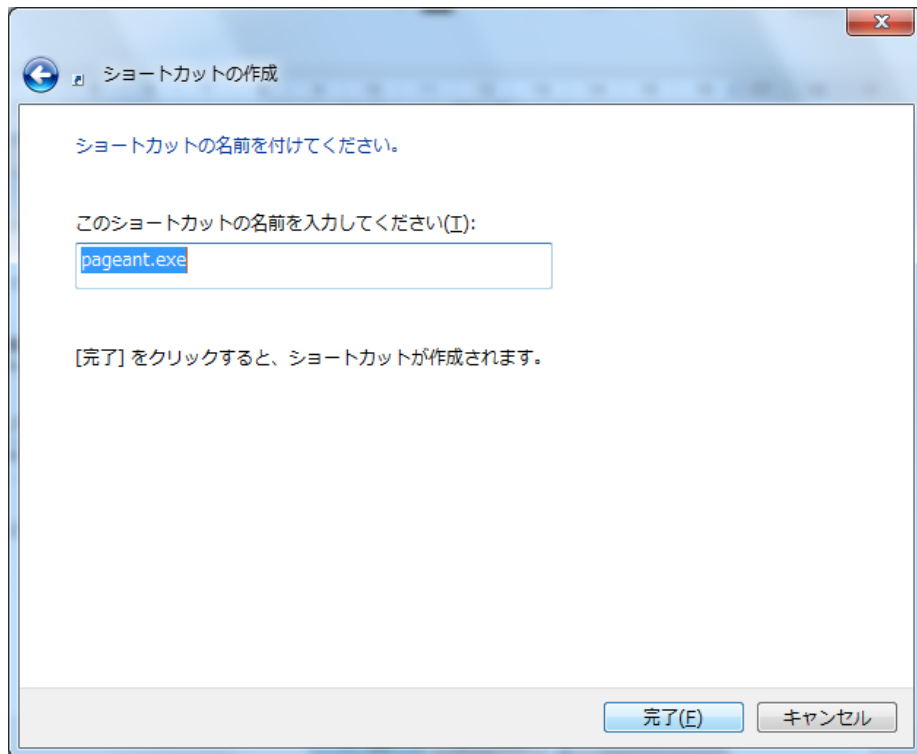
スタートアップ ディレクトリを開く。
エクスプローラーのアドレスバーに入力しても良い。



【ショートカットファイルの作成】
 スタートアップ ディレクトリで右クリックし、開かれたコンテキストメニューから [新規作成] > [ショートカット] をクリックする。

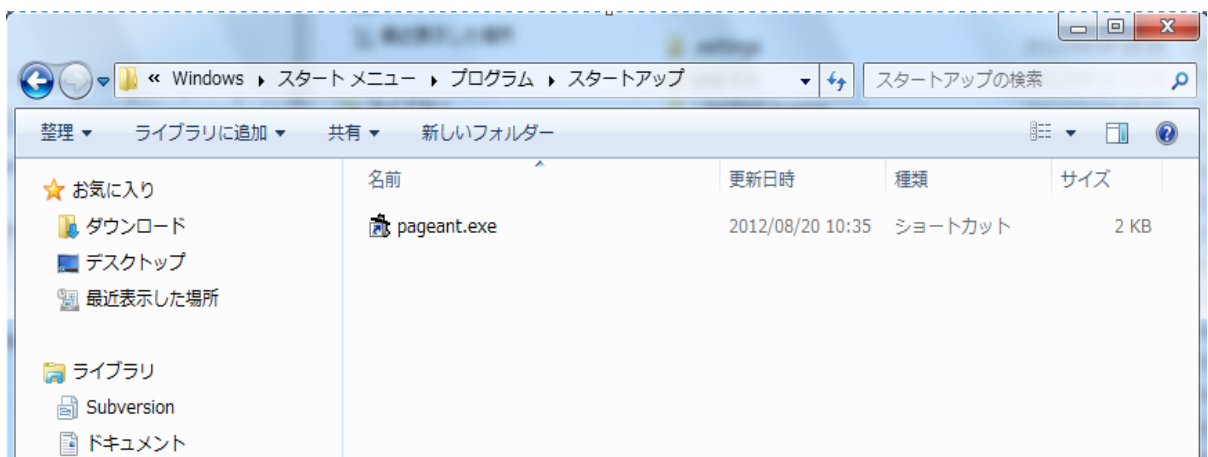


【ショートカットの作成ダイアログ】
 テキストボックスに **pageant** のファイルパスと、PuTTYgen で生成した秘密鍵ファイルのパスを設定する。



【ショートカットの作成ダイアログ(2)】

ショートカットの名前は特に変更する必要もなく、そのまま **[完了]** ボタンをクリックする。



【ショートカットファイルの確認】

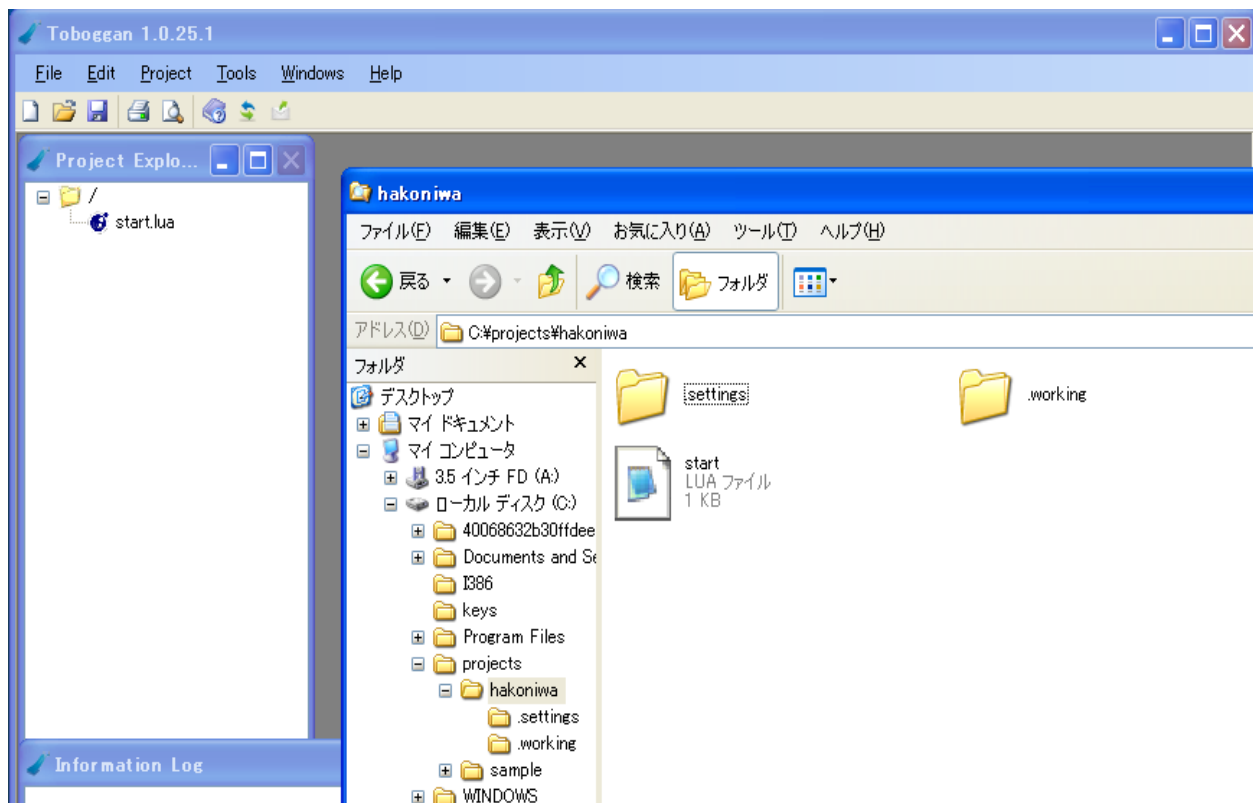
pageant.exe のショートカットファイルが作成されている事を確認する。

Point!

pageant の実行と秘密鍵の登録は Windows の起動の度に行わなければならないので、上記のようにスタートアップに登録しておくとう便利。
これを行うと、Windows 起動時にポップアップされるパスフレーズ入力ダイアログに、秘密鍵のパスフレーズを一度入力するのみになる。

b. プロジェクトをリポジトリからチェックアウトする

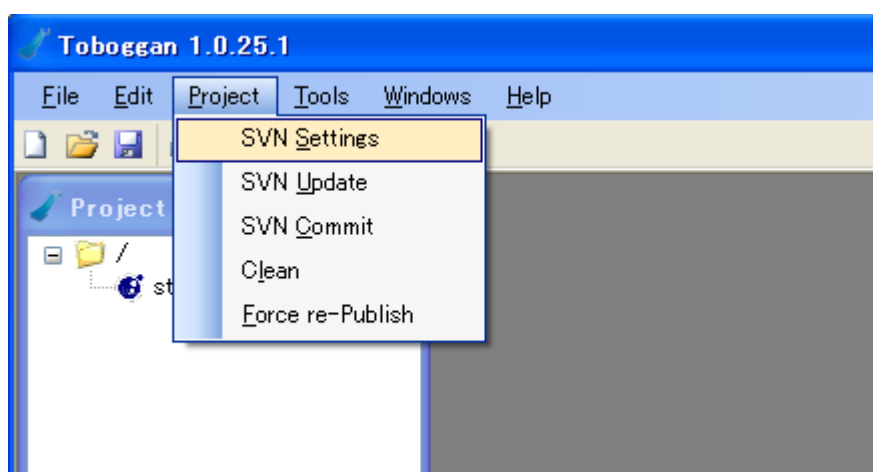
ここでは、hakoniwa を例に進める。



【専用のプロジェクトを作成】

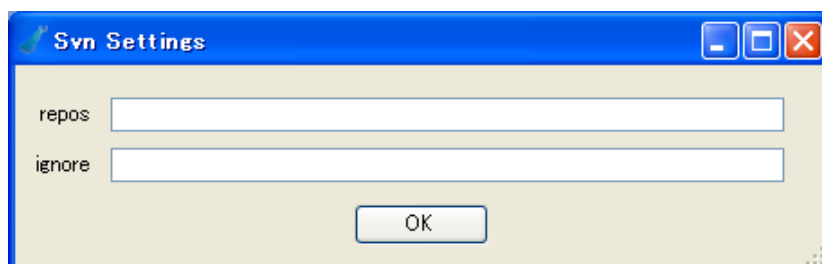
[新しくプロジェクトを作る](#) を参考に、専用のプロジェクトディレクトリに新規プロジェクトを作成する。

上図では **C:\projects\hakoniwa** に作成している。



【SVN クライアント機能の設定】

Toboggan メニューバーの [Project] > **[SVN Settings]** をクリックする。



【SVN リポジトリの接続情報設定ダイアログ】

[repos] の項目にプロジェクトリポジトリのパスを入力し、**[OK]** をクリックする。
hakoniwa では、下記のようなパスを入力する。

svn+ssh://[ユーザーID]@mysvnserver/hakoniwa/repo/project/trunk

ユーザーID は、リポジトリサーバに登録されているログイン用の ID で、[公開鍵をリポジトリサーバに登録申請](#) の際に作成されているはずなので、確認する。

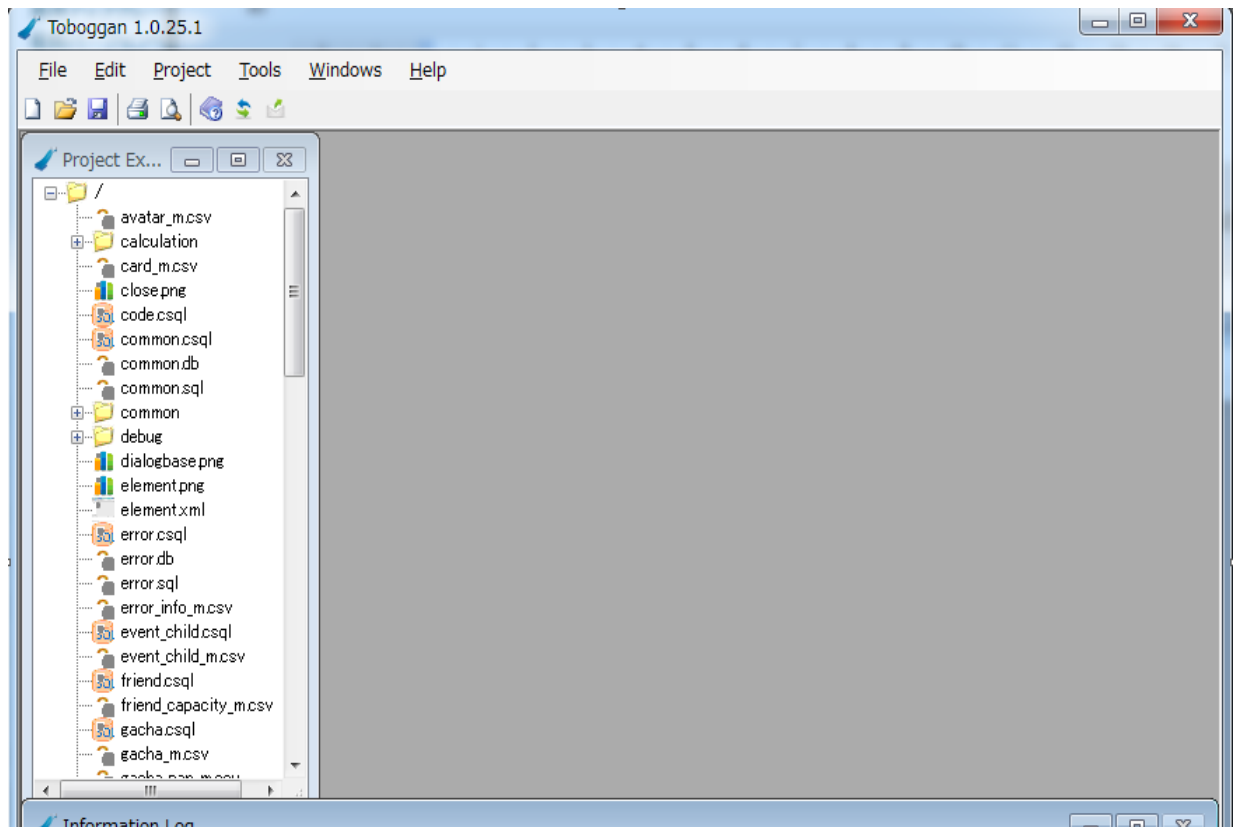
【チェックアウト中】

【OK】 をクリックすると、直ちに SVN リポジトリにアクセスされチェックアウトが実行される。

チェックアウトするプロジェクトの規模によっては数分程待つ必要があるので、気長に待つ。途中でアプリケーションを閉じてしまうと、中途半端な状態になりかねないので注意。

Notice!

鍵の認証に失敗していると、いつまでたっても処理が完了しない事がある。10分ほど待っても応答が無ければ、先ず認証の成否について確認する。

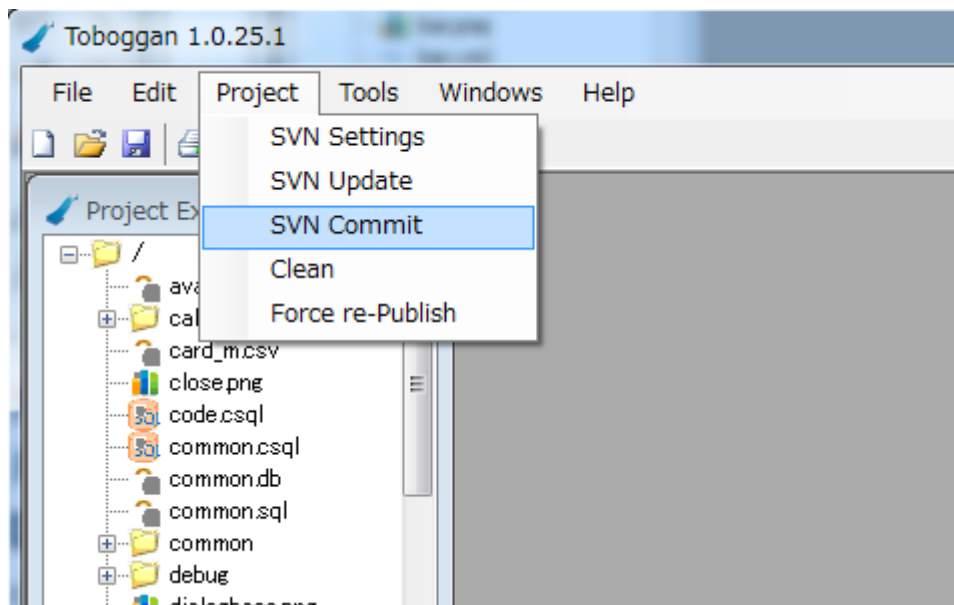


【チェックアウト後の Project Explorer】

チェックアウトに成功すると、Project Explorer にプロジェクトファイルの一覧が表示されるようになる。

c. プロジェクトをリポジトリにコミット⁴する

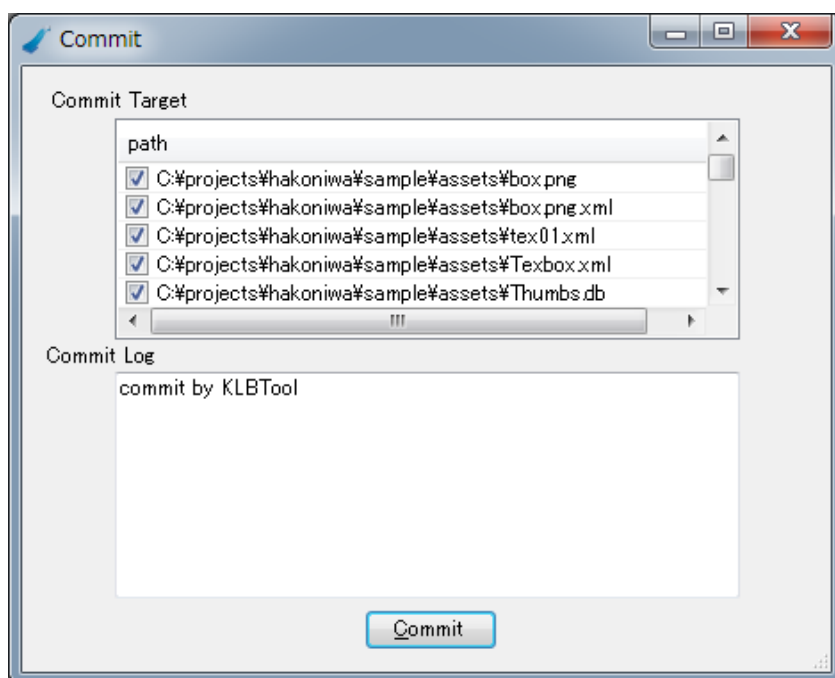
⁴コミット: チェックアウトしたプロジェクトはいわばコピーに過ぎず、これを編集しただけではチーム全体に共有されない。チェックアウト元のリポジトリに変更内容を送信する事で初めて共有・バージョン管理される。これをコミットという。



【プロジェクトのコミット】

チェックアウトしたプロジェクトに対してアセットファイルの追加や編集・削除を行ったら、その変更内容をリポジトリサーバへコミットする必要がある。

コミットは先ず、[Project] > [SVN Commit] をクリックする。

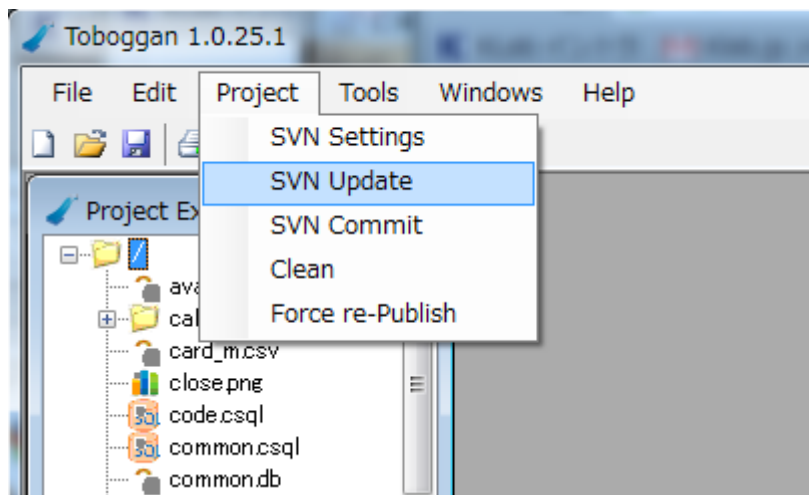


【コミット確認ダイアログ】

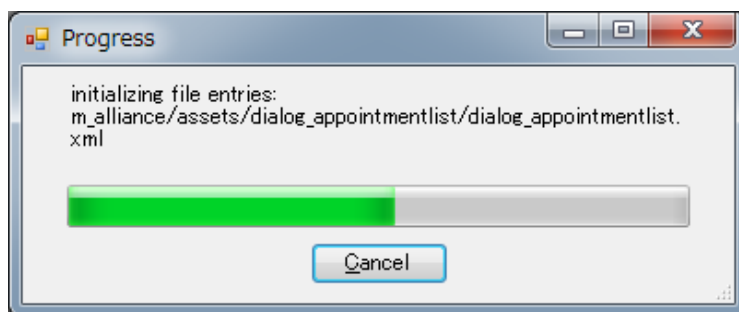
チェックアウトしたプロジェクトを変更していれば、変更したアセットファイルの一覧がダイアログの [Commit Target] に表示される。

コミットするアセットファイルを確認し問題なければ、[Commit] ボタンをクリックしてコミットする。

d. 最新のプロジェクトをリポジトリから取得する（アップデート）



【プロジェクトの更新⁷⁾】
 プロジェクトオープン時に自動的に更新されるのであまり気にする必要はない。
 明示的に更新するには、[Project] > [SVN Update] をクリックする。



【プロジェクトの再読み込み進捗ダイアログ】
 SVN リポジトリから最新のプロジェクトが取得されると、Toboggan へのプロジェクトの再読み込みが行われる。
 これが完了すれば、最新のプロジェクトを編集する準備が整う。

6. 付録

a. 用語集

| 名称 | 説明 |
|-----------|---|
| アセット | Toboggan で作成・管理される、静的なゲームデータファイルの名称。 イメージアセット、テクスチャアセット、UIアセット等がある。 |
| パブリッシュ | アセットをゲームエンジンがロード可能な形式へ変換する事。単にパブリッシュと云った場合には、プロジェクト内の全アセットの一括パブリッシュを指す。 |
| オーディオアセット | 音声ファイル。mp3 等がサポートされている。 |

⁵ プロジェクトの更新: プロジェクトをチェックアウトした後も、他の人によってプロジェクトは常に変更され続けている。

⁶ 他の人の変更内容と衝突しない為にも、プロジェクトはこまめに最新化（更新）する必要がある。

⁷

| | |
|--|--|
| | |
|--|--|

b. ショートカットキー表

| キー | 説明 |
|------------------|--|
| Ctrl + O | プロジェクトフォルダーの参照ダイアログを表示します。 |
| Ctrl + S | アクティブなエディタの編集内容をプロジェクトに保存します。 |
| Ctrl + Shift + S | アクティブなエディタの編集内容を保存し、直ちにパブリッシュします。 |
| Ctrl + Shift + P | プロジェクトをパブリッシュします。 |
| Ctrl + Shift + R | パブリッシュされたプロジェクトデータをゲームエンジンにロードし、実行します。 |
| Ctrl + Z | 最後に行った編集操作を取り消します。 |
| Ctrl + Y | 直前に取り消した編集操作をもう一度行います。 |