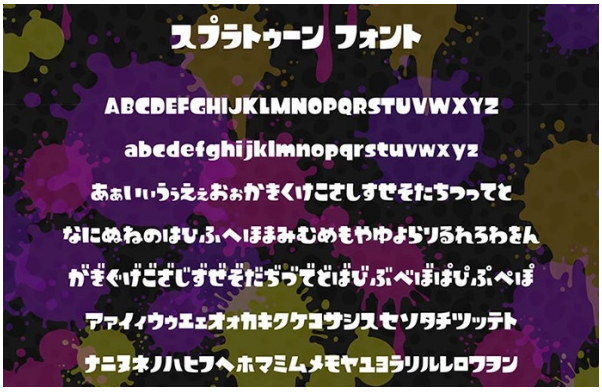
今回のレッスンで学ぶ概念

ビットマップフォント（書体ファイルではなく画像で表現する）

Unityでのゲーム制作で、デフォルトの「Arial書体」ばかりでは、なかなかゲームの世界観が出せませんし、各方面への応募作品や相手先に披露するのにも共感が得られません。ゲームの世界観に合わせた画像文字が必要です。

ここではUnityで気軽にビットマップフォントを実現する為の手法（生成ソフト）を2つ紹介します。

- 書体ファイルを変換する 【BMFont】
- 画像をフォントに変換する 【ShoeBox】



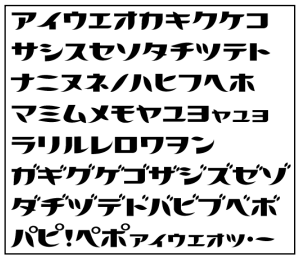
書体ファイルではなく、画像で作ってUnityで表示中。

もちろん、Unityで作られたコンテンツの中に通常のArial書体などを用いたまま運用されているケースも非常に多くあり、問題なく運用されています。しかし、ゲームとなるとデータの軽量化と高速化、効率化、そして世界観の訴求が必要です。これらのどの項目にも効き目があるのが、フォントファイルを使わずにビットマップフォントを扱う手法となります。

フォントファイルのまま

ゲーム中で使わない全ての文字が入っていて、データ量が多くてメモリ効率が悪い。なにより、スマート機器向けのゲームには、フォントの埋め込みが出来ない！

画像で準備



BMFont(Bitmap Font Generator) で作る

PCなどにインストールされている書体ライブラリの中から、指定したフォントをビットマップフォントに変換する手法です。基本的な動作は、フォントファイルから一括指定した文字群を1枚の画像に効率よく詰め込みます。全ての文字の位置（X,Y）と大きさ（W,H）を記述した、住所録に相当するテキストファイルもXML形式で同時に書き出します。

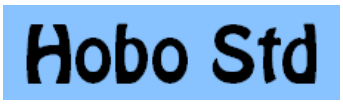
【BMFontの入手】

http://www.angelcode.com/products/bmfont/

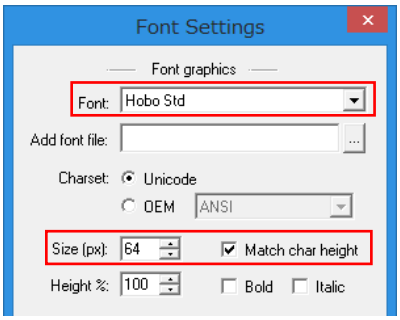
変換元となるフォントは、著作権上で変換が禁止されていないものを選びます。禁止されている物は、おそらく変換後に動作不正になることが多いです。フォント作家さんがネットで公開している場合もありますが、個人利用と商用利用について明記していますし、クレジットを正しく表示する利用規約もあります。ただ、名前を売りたい意図もあり、アプリ内で同梱する事が許可されている明記もあります。きちんと読んで、正しい利用をします。

【STEP1】 フォントファイルの変換

今回はPCに内蔵されていた書体から「Hobo Std」を選択しました。現在の作業環境を確認し、気に入った書体を探して下さい。（但し、変換に失敗する書体もあることを忘れずに。）



- BMFontを起動したら、メニューOptionsからFont Settingsを選択します。
  使いたいフォントと、出力する文字の大きさを指定します。
  ボタンOKを押下します。

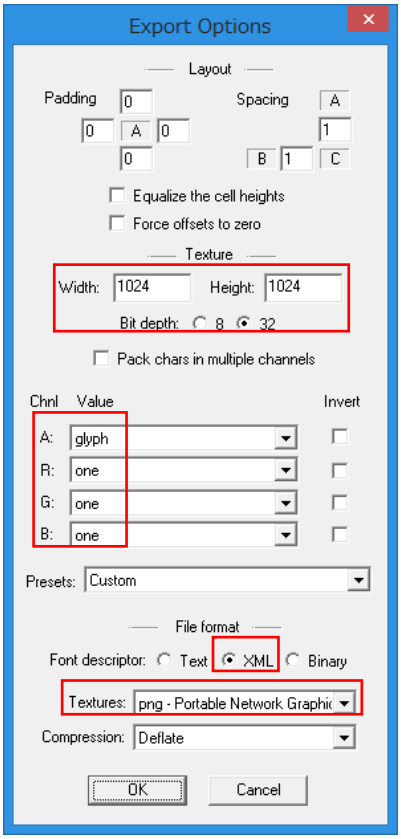


- メニューOptionsからExport Optionsを選択します。
  仕上がる画像に関する指定や、文字をどのように詰め合わせるのか？の指定を行います。（サイズは2の乗数にします。）

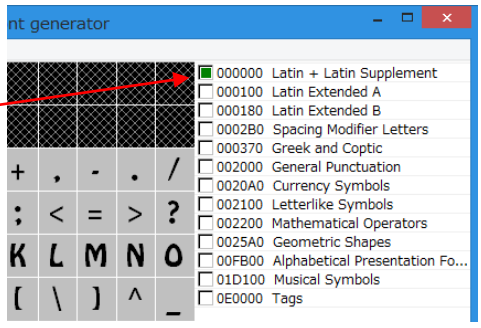
  アルファ部分も文字の面積として扱い、着色しやすいようにRGBは白にします。

  文字の住所を記したファイルはXML形式で、画像そのものはPNG形式を指定します。

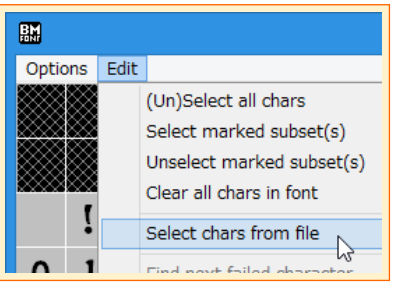
  最後にボタンOKを押下します。



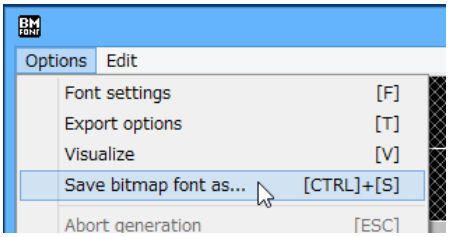
- 変換する文字は1字1字指定するのではなく、フォントセットで指定される文字群となります。
  1行目の Latin + Latin Supplement を選択します。



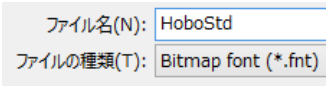
漢字の場合は、必要な文章から重複する文字を省いたテキストファイルをメモ帳で作成します。「王様姫大臣勇者城魔物村町あいう.....」みたいな感じ。
準備が出来たら、メニューEditから Select chars from file を選択し、このファイルを指定すれば、漢字の指定が完了します。



● メニューOptionsからSave bitmap font as…を選択します。

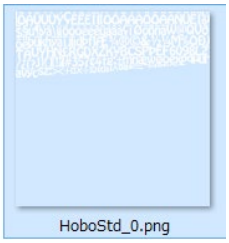


妥当なフォントファイル名を指定します。



書き出しが完了し、XMLファイル（拡張子は.fnt）と画像ファイルが作成されます。
画像ファイルは1～4枚くらいになることがあり、0から枝番が割り振られます。
1枚の場合は枝番が不要なので、削っておいていいでしょう。


HoboStd.fnt


HoboStd_0.png

Unityでビットマップフォント利用する

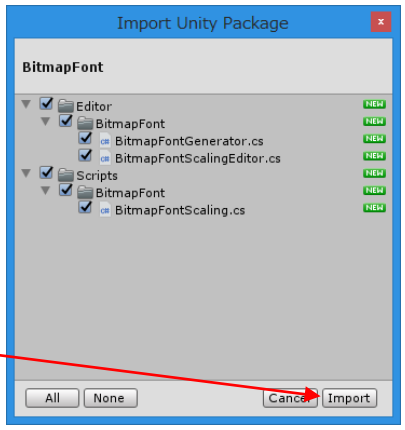
【STEP2】 プロジェクトの準備

● 新規にプロジェクトを準備し、Game画面をStandalone (1024x768)にします。

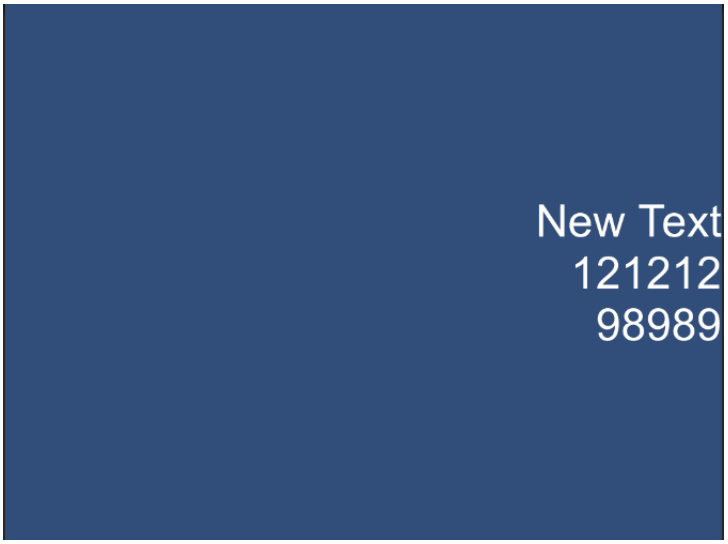
● ヒエラルキー欄のMain Cameraを選択し、インスペクタでパラメータを設定します。

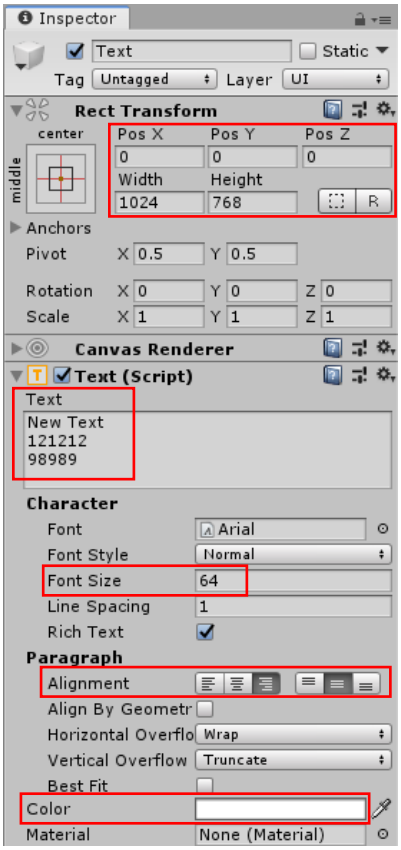
● メニューAssets から Import Package > Custom Package を選択し、配布してある BitmapFont.unitypackage を指定します。

ファイル一覧が表示されたら、Import を押下します。



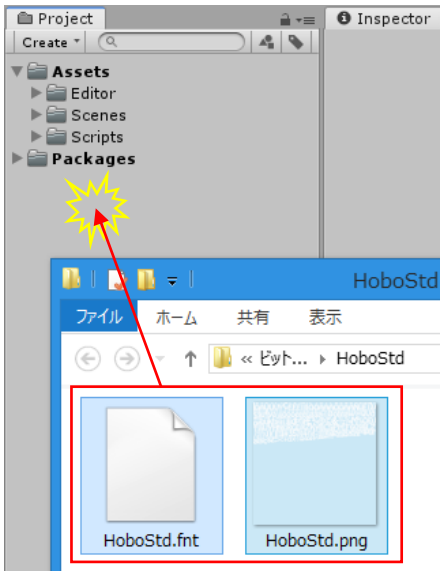
- ヒエラルキー欄の Create から UI > Text を選択します。インスペクタでパラメータを設定します。



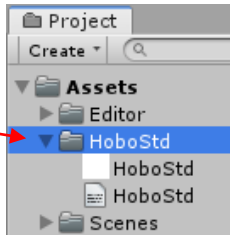


【STEP3】 ビットマップフォントの導入①

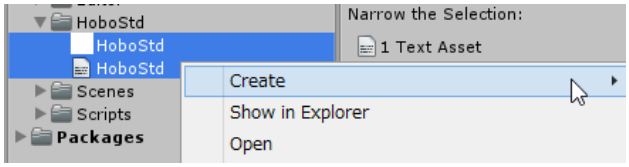
- 前工程で作成したXMLファイルとPNGファイルをプロジェクト欄にドラッグ＆ドロップします。



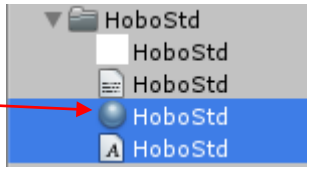
以降の作業でファイル数が増えるので、フォルダ管理をしておきます。



- この2つのファイルを両方とも選んだ時の反転文字を右クリックし、Create > Bitmap Font を選択します。

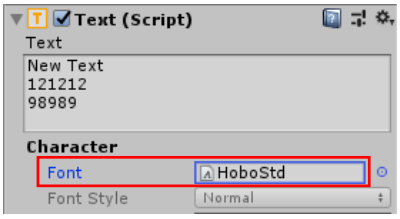


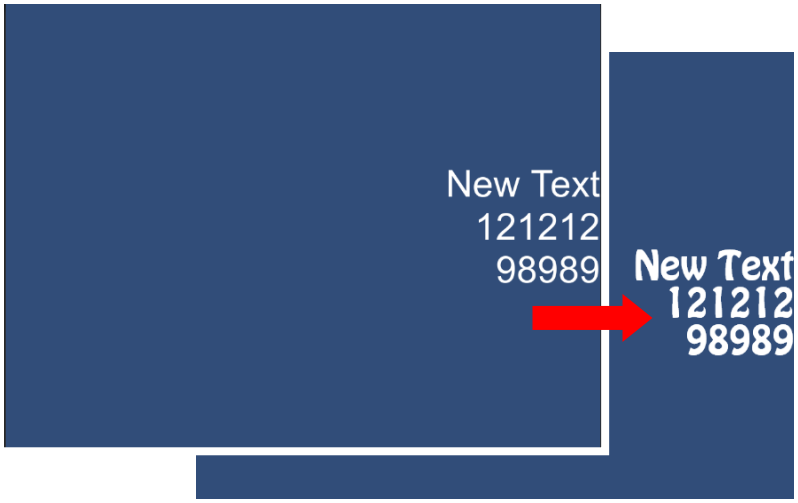
上手くビットマップフォントの導入が出来ると、マテリアルとフォントファイルの2つが生成されているのが判ります。



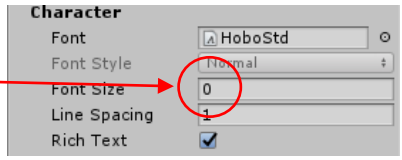
フォントそのものに問題があったり、変換拒否の設定があったりする場合は、この作業がエラーになります。今のところ回避策はありません。

● ヒエラルキー欄の Text を選択し、インスペクタで Font を指定します。

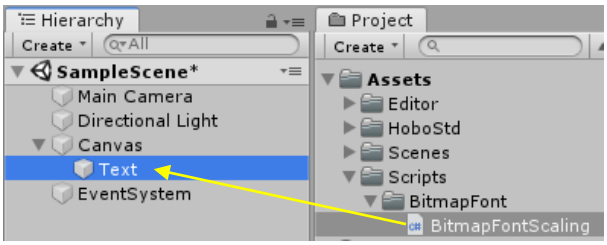




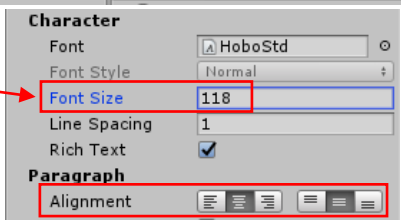
この時、奇妙な状況を見かけます。
Font Size が 64 だったはずなのに、ゼロになっています。
この数値を変更しても、文字の大きさに変化がありません。
それもそのはず。1文字の最大の大きさを 64 ピクセルで作ったので、ピクセルの値がそのまま画面内での大きさとなっているからです。



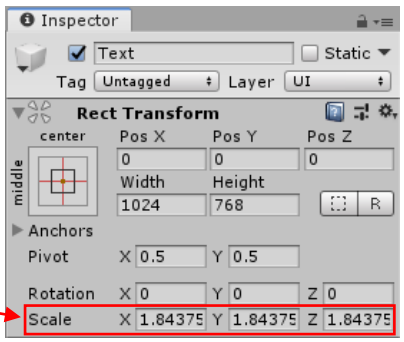
● 文字の大きさは設計通りであっても、制作経過の中で微調整も発生するので、対策方法があります。
プロジェクト欄のスクリプト BitmapFontScaling をヒエラルキー欄の Text にドラッグ＆ドロップします。



フォントサイズを変更できるようになります。



仕掛けは単純で、フォントサイズで変更した値を、元のサイズとの比率に変換し、オブジェクトの拡大縮小に自動で代入してくれるスクリプトです。
例） 118 ÷ 64 = 1.84375



元々の値である 64 ピクセルが増えたりはしないので、あまり大きな値を設定すると、ピクセルのジャギが目立ってしまいます。ほんの少しの微調整と考えておき、作り直した方が綺麗に見えることを覚えておきます。

ShoeBoxで作る

シェアウェア ShoeBox（シューボックス）は、2Dゲーム開発で必要になる画像処理ツールが数多く同梱されており、使っていて当然の存在としてゲーム開発の現場で使用されています。（なぜ名称が靴箱なのか？は不明です。）

その中でも今回は、画像編集ソフト（Photoshop など）で精緻に制作された画像からビットマップフォントを作る機能を解説しています。有償ソフトですが、価格は各人が寄付の形式で決める方式となっています。

【ShoeBoxの入手】

http://renderhjs.net/shoebox/

このソフトはAdobe（アドビ）のAIR（エア：Adobe Integrated Runtime）という仕組みで動くので、PCにAIRのインストールが先になります。AIRはアプリ動作させる為の仮想ステージをソフトとして動かす形になり、そのソフトの上で様々なAIRアプリが動作する感覚となります。

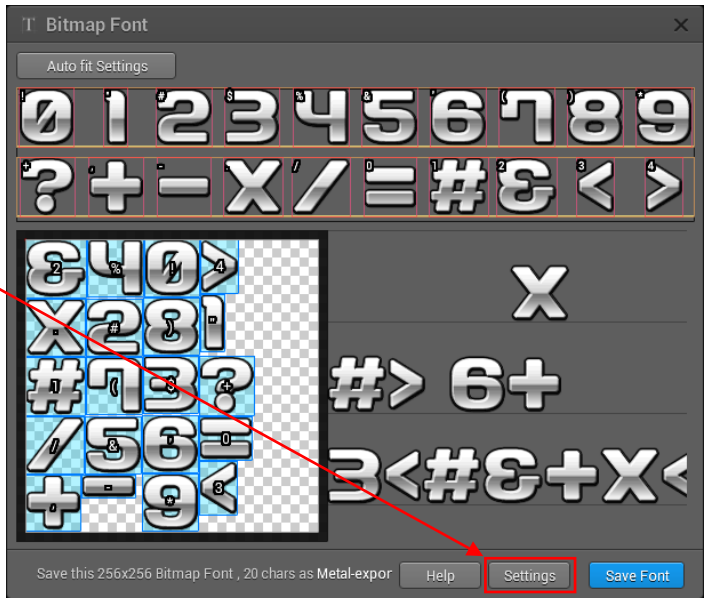
【AIRの入手】

https://get.adobe.com/jp/air/

【STEP4】 画像ファイルの変換

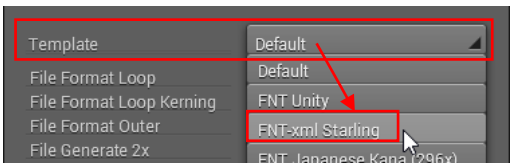
- ShoeBox を起動します。
  配布してある画像ファイル Metal.png を GUI タブの Bitmap Font アイコンにドラッグ＆ドロップします。

- 文字の隙間や大きさなど、画像を解析し設定する画面が表示されます。
  Settings を押下します。

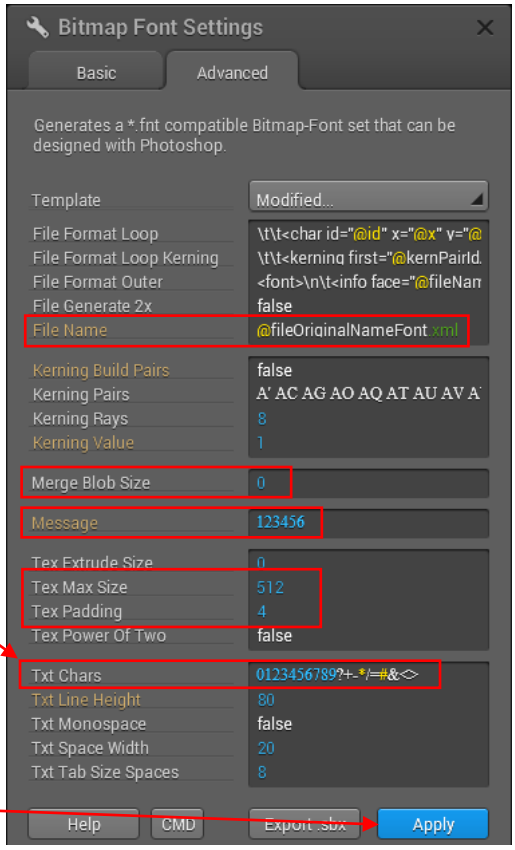


- 設定のテンプレートから FNT-xml Starling を選択します。

  隣に FNT Unity があって紛らわしいですが XML 形式を選択します。



● 各種項目を設定します。



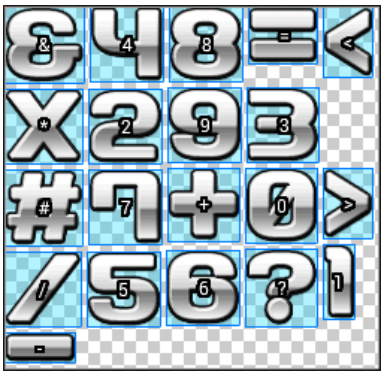
フォントファイル名は@fileOriginalNameFont.xml とします。

項目 Txt Chars の内容は、以下の文字列で置き換えます。

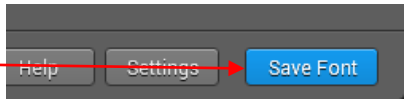
0123456789?+-*/=#&<>

全ての設定が出来たら Apply を押下します。

● 文字画像が分割され、割り当てたい上記テキストと一致していることを確認します。



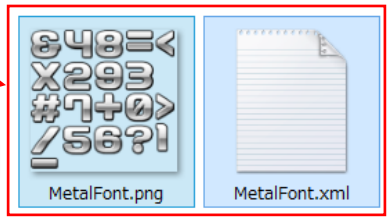
● Save Font を押下します。



画像ファイルの設置場所に2つのファイルが作成されます。

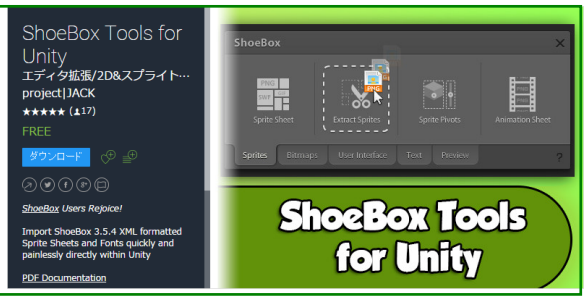
一つは先ほど文字画像を配置し直した画像ファイルで、もう一つはバラバラになった文字画像の位置座標を記録した XML 形式ファイルです。


Metal.png



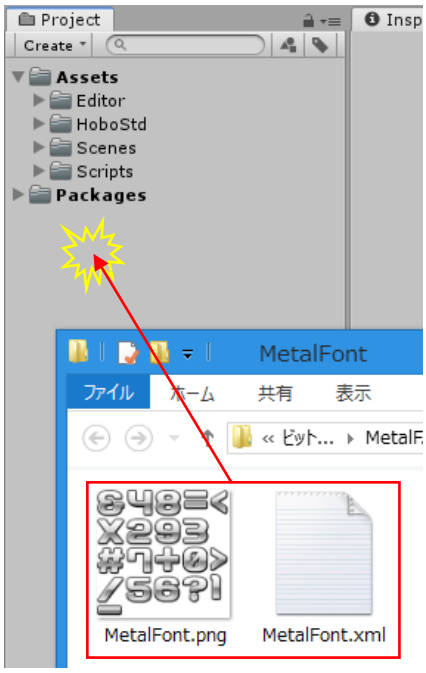
この2ファイルを Unity に読み込む際に Unity のアセットストアから無料ダウンロード可能なユーティリティが提供されています。

ですが、このユーティリティ。思うように動作してくれず、先と同様の方法で導入します。

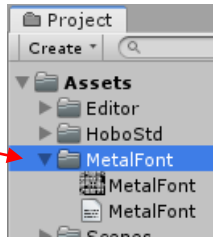


【STEP5】ビットマップフォントの導入②

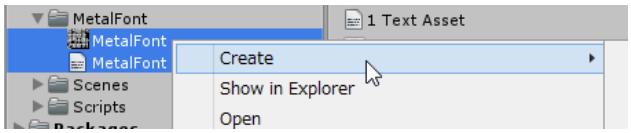
- 前工程で作成したXMLファイルとPNGファイルをプロジェクト欄にドラッグ＆ドロップします。



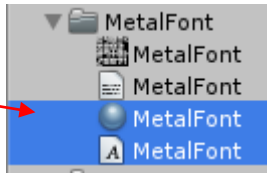
以降の作業でファイル数が増えるので、フォルダ管理をしておきます。



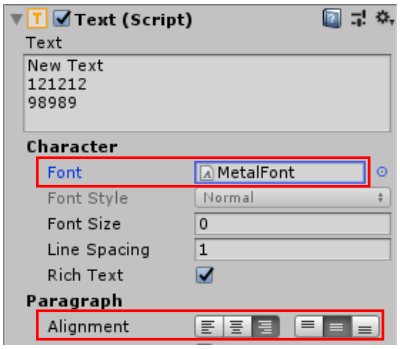
- この2ファイルを両方とも選択し、その反転文字の上で右クリックし、Create > Bitmap Font を選択します。



マテリアルとフォントの2ファイルが作成されます。



- ヒエラルキー欄の Text を選択し、インスペクタでパラメータを設定します。



ここで奇妙なことが起こっています。文字 New Text は画像に無いので、表示されないのは奇妙ではないのですが、12万の数値と9万の数値を上下に並べた時に、9万の方が大きい値に見えてしまうのです。
もちろん理由は明らかで、画像の文字を切り出す際に1の幅が9や8などに比べて極端に小さく、等幅のフォントとして生成されていないことに起因しています。

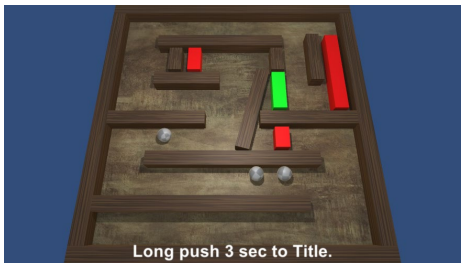
様々な解決策がありますが、原始的な方法として、幅が足りない文字の背景に濃さ 1%の板を敷いて等幅に認識させる方法があります。

変換の設定で回避できる場合もあります。

既存のゲームへの導入例

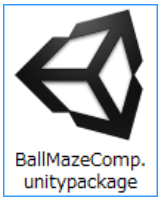
今回は、ゲーム構築の手間にスポットライトを当てないので、既出のゲーム「BallMaze」を流用します。

このゲームの完成状態を配布しますので、5位までのハイスコア管理画面に用いている数値に、ビットマップフォントを導入する手順を中心に解説を行います。

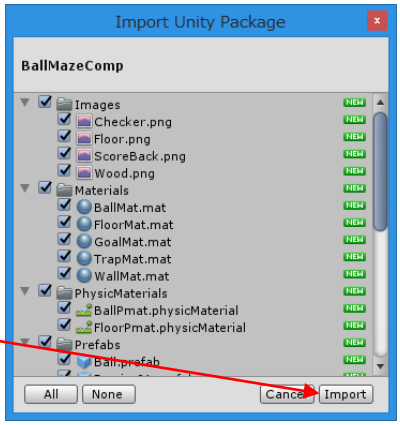


【STEP6】 完成状態の復帰

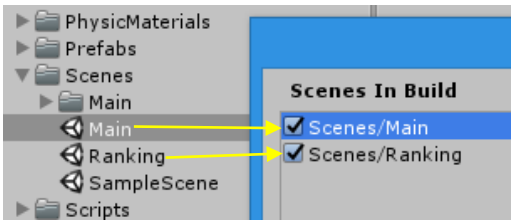
- 配布された素材を読み込みます。
  メニューAssets（アセッツ）から Import Package（インポートパッケージ）> Custom Package（カスタムパッケージ）と進み、今回のフォルダ内にある BallMazeComp.unitypackage を指定します。



- 内容物が表示されたら、ボタン Import（インポート）を押下します。



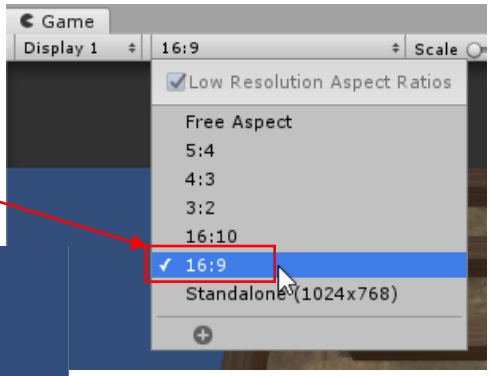
- メニューFile から Build Settings...を選択し、プロジェクト欄のシーン Main と Ranking を Scenes In Build にドラッグ＆ドロップします。
  1行目に Main が来るようにします。

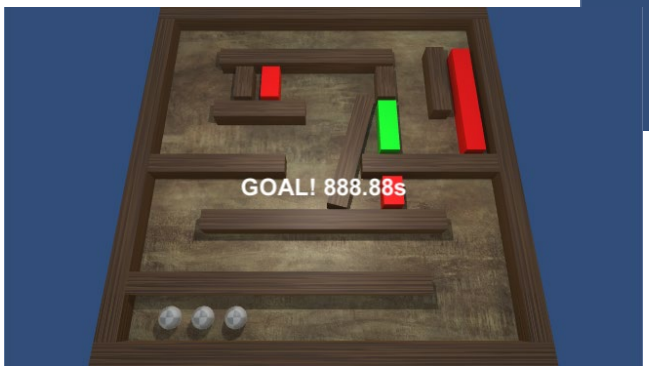


- プロジェクト欄のシーン Main をダブルクリックして表示します。

- 各人の制作プラットフォームの比率に応じ、ゲーム画面の比率を調整します。

  ここでは、PCの全画面でプレイすることを想定して、16:9 の比率としています。iPad なら 4:3 となります。



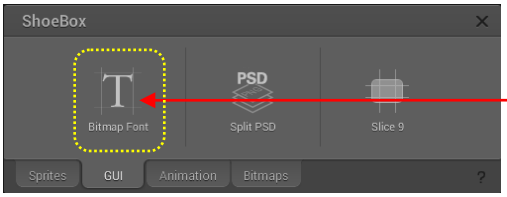


● プレイボタンを押下します。

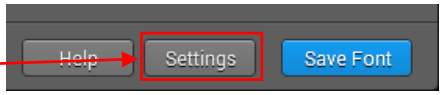
このシーンMainは完成状態なので、通常にプレイ操作が可能です。ボール3個を同時にゴールに突入させるまでの時間を計測し、ランキング画面で数件の記録が登録されるまでプレイを行います。

【STEP7】 時間表示用のフォント作成

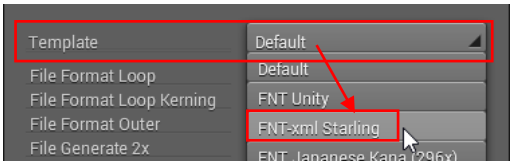
● 配布してある画像ファイル Time.png を ShoeBox の GUI タブの Bitmap Font アイコンにドラッグ&ドロップします。



● 文字の隙間や大きさなど、画像を解析し設定する画面が表示されます。Settings を押下します。



● 設定のテンプレートから FNT-xml Starling を選択します。
隣に FNT Unity があって紛らわしいですが XML 形式を選択します。



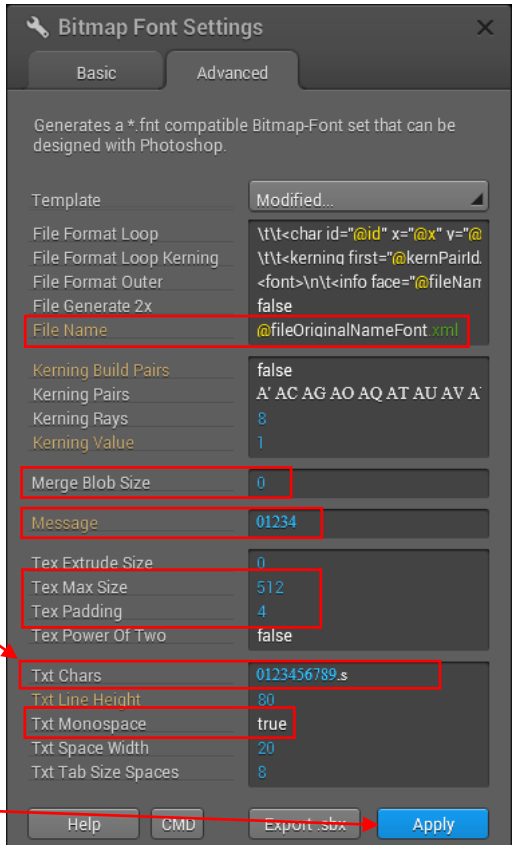
● 各種項目を設定します。

フォントファイル名は@fileOriginalNameFont.xml とします。

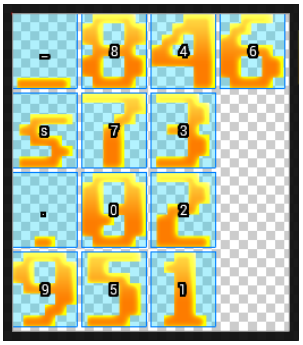
項目 Txt Chars の内容は、以下の文字列で置き換えます。

0123456789.s_

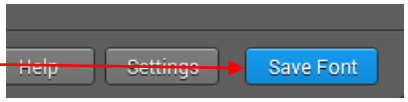
全ての設定が出来たら Apply を押下します。



● 文字画像が分割され、割り当てたい上記テキストと一致していることを確認します。

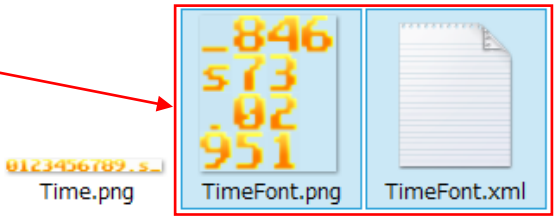


● Save Font を押下します。



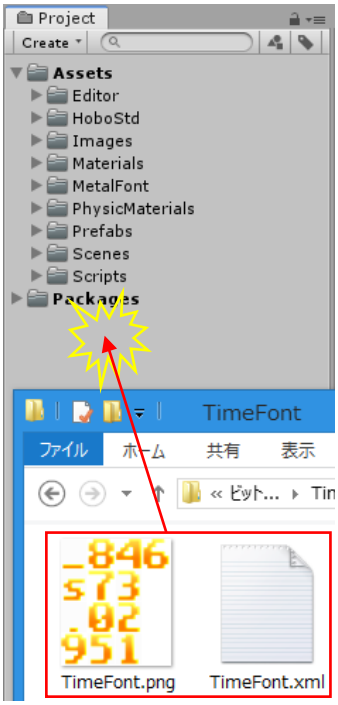
画像ファイルの設置場所に2つのファイルが作成されます。

一つは先ほど文字画像を配置し直した画像ファイルで、もう一つはバラバラになった文字画像の位置座標を記録したXML 形式ファイルです。

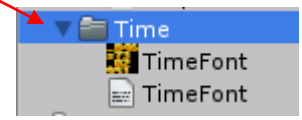


【STEP8】 時間表示用のフォントを用いる

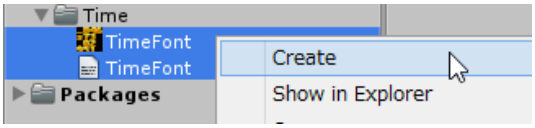
● 前工程で作成したXMLファイルとPNGファイルをプロジェクト欄にドラッグ &ドロップします



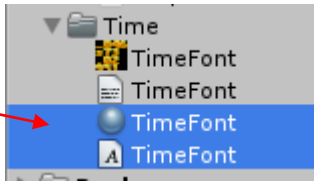
● 以降の作業でファイル数が増えるので、フォルダ管理をしておきます。



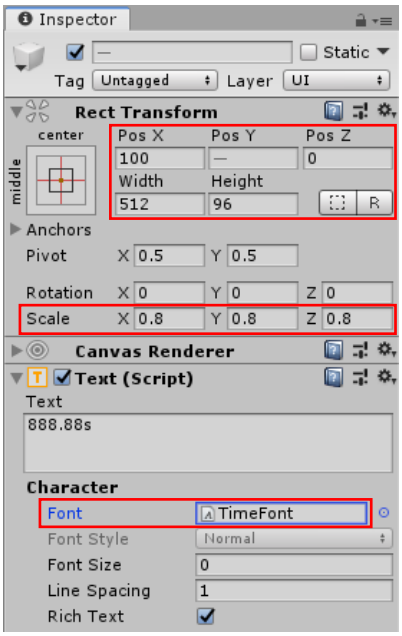
● この2ファイルを両方とも選択し、その反転文字の上で右クリックし、Create > Bitmap Font を選択します。



マテリアルとフォントの2ファイルが作成されます。



- プロジェクト欄のシーン Ranking をダブルクリックして表示します。
- ヒエラルキー欄の txtRank1～5 までについて、ポジションY座標値を変更します。
  - txtRank1　　Position-Y：240
  - txtRank2　　Position-Y：140
  - txtRank3　　Position-Y：40
  - txtRank4　　Position-Y：-60
  - txtRank5　　Position-Y：-160
- ヒエラルキー欄の txtRank1～5 の全てを選択した状態で、インスペクタでパラメータを一気に設定します。



- プレイボタンを押下します。 

標準のArialフォントだった表現力が、ビットマップフォントに置き換えられています。

付録

【オンライン変換：Font Meme】

今回は変換元となる画像文字を配布しました。どれも変換が上手く行くように確認済みのファイルです。

Time.png

Orange.png

Metal.png

Techno.png

自身が想定するゲームの世界観にマッチした上記のような文字画像は、ネットで探してもなかなか得られないことが多く、検索に時間を費やすばかりです。

フォントを見つけて来てPCに実装し、ワープロソフトで必要な文字を入力し、それを画像にして出力する試みも見かけましたが、それをネットで実現してしまうサービスがあります。

Font Meme

【Font Meme：日本語対応有り】

https://fontmeme.com/jfont/

ただ、ここで得られるのは、単色の輪郭画像だけです。彩色やテクスチャ、装飾を施したり、立体的に見せたりする作業には、やはり意図や用途を理解できるデザイナーが、画像処理ソフトで加筆修正することになるでしょう。

注意したいのは、本当に世界観にマッチしているかどうか？であり、ビットマップフォントを用いたが故に、違和感が生じてしまうのでは本末転倒となります。

「使う」という目標よりも、「世界観の調和」を目指して用いたいものです。

以上