# 自己紹介

Voyage Voyage

VRCHAT民 since 2020/10
プレイ時間 : 3000時間以上

# VRCHAT向けの自作

- Quick Avatar Setup (Godot Engine + Unity Editor - 自分用)
  - Blendshapes Animationsで表情をアバターに付けるツール
- World Lock Autosetup (Unity Editor - リリース)
  - アバターにワールドで固定できるアイテムを装備するツール

そして色々・・・

今回の話題は新しいの自作:

- Myy bones merger (Blender - リリース)
  複数のボーンとそのウエイトペイントを結合ためのBlenderアドオン

# 作る理由

## 問題

普段、使ているVRCHAT用モデルを最適化します。
アバターをアップロードするほど、最適化のやり方を覚えましたね。

3Dモデルを最適化したいなら、減らす所は三つがあります：

- マテリアル数
- ポリゴン数
- ボーン数

マテリアルとポリゴン数を減るのは大分簡単です。

マテリアルはAtlasを作れば、すべてのUVをAtlasに移動したら、ほとんど完成です。

ポリゴンは面倒です。
Decimate・Modifierを使って、ポリゴンを減ることが出来ますが：

- Blendshapesのメッシュを外さなきゃなりませんね。しないと、Modifierを適用できません。
- そして時々、UVを直すところがあります。

にしても、時間が掛かっても、やることは把握しています。

問題はボーンを減らすこと

まずは、ボーンを85個以上があれば、
VRCHATのExcellentのランキングをたどり着くことが不可能です。

でもね、ボーンは簡単に消すことが出来ません。
そうすると、モデルがグニャグニャになりますね。

理由は消したボーンの関連する頂点がもう動きません。

# ボーンの仕組み

どうして動かないですか

モデルの頂点にはウェイトの情報が含んでいます。
大体その情報は 0.0~1.0　（つまり0% ~ 100%）のFloatです

そして、UnityとBlenderはそれぞれの方法でそのウェイト情報をあるボーンに繋がります。
で、そのボーンを動くと、繋がった頂点がウェイトによると動きます。

Blenderでは、ボーン・ウェイト・頂点の関係はこんな感じです：

（Armature) ボーン <- (Mesh) Armature Modifier, バインド先：頂点グループ -> 同名頂点グループ <- ウェイト

つまり、BodyのArmature Modifierが、Armatureの各ボーンをMeshの同名頂点グループに繋がります。

例えば、Headのボーンは
Headの頂点グループに繋がって、
そのグループのの頂点に割り振れたウェイトを使います。

# 問題を解決方法

じゃぁ・・・

AとBのボーンがあれば、
Aのボーンを消したら、
その後、Aの同名頂点グループの頂点情報を
残っているBのボーンの同名頂点グループに追加したら、
Bを動かすと、
Bの頂点と共に、Aの頂点が再び動くはずですね。

これで、グニャグニャの問題がなくなります。

と思い始めました。

そして、この推理を確認できるようにModifierが存在します。

「頂点ウェイト合成」Modifier

使い方は：

- MeshにModifierを付けて、
- Aの頂点グループを決めて
- Bの頂点グループを決めて
- 混ざり方法を決めて、
- 適用します。

はい、出来上がり！

その方法を使うと、確かに、動かなった頂点はもう華麗に動きますね。

じゃぁ、120ボーンのモデルを85以下にしたいなら：

- あるボーンAを消して
- Meshに「頂点ウェイト合成」を付けて
- Aの頂点グループを決めて
- 残っているボーンBの同名頂点グループを決めて
- OperationをAddにして
- 適用して
- そしてそれを35回をくりかえす

・・・

ざっけなよ！

2022年ですよ！

こんな時期で、ボーン＋ウェイトを簡単に合成できるシステムがあるはずなんですよ！

・・・

Dissolveかな？

いや、Dissolveはボーンを合成しますが、頂点グループを完全に無視しますね。

Dissolveでウェイトペイントを結合する方法を検索しましょう・・・
「頂点ウェイト合成」を使えばいいの答えしか出ない・・・はい・・・

もう方法が解っているんですから。
でも一回でやると、二度とやりたくないです。

・・・　スクリプトで自動化できないかな　・・・

# 難点１： BlenderのAPI

BlenderのAPIは変です。

ある難しいことを簡単にできる。
ある簡単なことがとんでもなく面倒くさくにする。

結局、スクリプトで自動化したいなら、自分の目的はこうになります：

- 消すボーンを決めて
- ウェイトペイントを受けるターゲットボーンを決めて
- それぞれの頂点グループを見つけて
- 消したボーンの頂点グループの情報をターゲット・ボーンの頂点グループに追加する

目的を達成したら、完成になるはずです。

Blenderでは、選択とアクティブ選択と言う違いがあります。
アクティブ選択は最後に選択された物です。

初期設定では、アクティブ選択のOutlineは黄色
他の選択のOutlineはオレンジ色

じゃぁ、アーマチュアの編集時に、ボーンを選択したら：

- 選択を消すボーンに決めて
- アクティブ選択をターゲットボーンに決めたら、
  最初の目的を達成出来ます。

次は、頂点グループを見つける

BlenderのScriptingの画面でREPLがあります！
しかも、TABを使うとオートコンプリートがでます！
すごいね！
そして、そのオートコンプリートで、メソッドとプロパティの説明書が付けています！

`bpy.context.object.vertex_groups[` <TAB> <TAB>

Dictionaryです！選択されているボーンの名前を使えば、関連の頂点グループを簡単に見つけるはずです！

```
>>> bpy.context.object.vertex_groups['HairFront1'].add(
add()
VertexGroup.add(index, weight, type)
Add vertices to the group
```

なるほど、これで追加方法が解った、後は・・・

```
>>> bpy.context.object.vertex_groups['HairFront1'].items(
items()
.. method:: items()
Returns the items of this objects custom properties (matches Python's
dictionary function of the same name).
:return: custom property key, value pairs.
:rtype: :class:`idprop.type.IDPropertyGroupViewItems`
.. note::
   Only the :class:`bpy.types.ID`, :class:`bpy.types.Bone` and
   :class:`bpy.types.PoseBone` classes support custom properties.
```

はい、もうできたね！

```
>>> bpy.context.object.vertex_groups['HairFront1'].items()
Traceback (most recent call last):
  File "<blender_console>", line 1, in <module>
TypeError: bpy_struct.items(): this type doesn't support IDProperties
```

・・・？

・・・・・・

はい・・・なるほど・・・

あ！Valuesかもしれません

```
>>> bpy.context.object.vertex_groups['HairFront1'].values()
Traceback (most recent call last):
  File "<blender_console>", line 1, in <module>
TypeError: bpy_struct.values(): this type doesn't support IDProperties
```

💢・・・糞APIの予言・・・

まぁ、まぁ、まぁ、オートコンプリートで説明書が付いていますが

当たるまでメソッドをランダムに試したら霧がないね。
本物の文章を読む方が速いですね。

VertexGroup

## メソッド

- add(index, weight, type) : Add vertices to the group
- remove(index) : Remove vertices from the group
- weight(index) : Get a vertex weight from the group

おわかりいただけただろうか？

## メソッド

- add(index, weight, type) : グループに頂点を追加できる。
- remove(index) : グループから頂点を消すことが出来る。
- weight(index) : グループに含んでいる頂点のウェイト情報を確認できる。

頂点のリストを取ることが出来ないよ！

狂うタイム

CREATE・READ・UPDATE・DELETEじゃなくて、
CREATE・UPDATE・DELETEだね！
READはできない！

グループのObjectを作ると、そのグループのメンバー・リストを返すメソッドはザ・基本なんですけど！

そのAPIを作った人は使ったことがある！？
あったら、使い方が教えて貰ってほしいね！

って、本気かよあいつら！？

ふぅ・・・

落ち着きましょう。

・・・

Blenderでは頂点グループのパネルの下に、「選択」というボタンが置いてありますね。
じゃぁ、

# Q. 頂点グループの「選択」ボタンをクリックする時、Blenderはどうやって正しい頂点を選択出来ますか？

ボタンをクリックして、ScriptingのConsoleを見ましょうか

```
bpy.data.objects["Body"].(null) = 3
bpy.ops.object.vertex_group_select()
```

まじかよ

`bpy.data.objects["Body"].(null) = 3`ってなんだろう・・・

はい、ネットで答えを探しましょう。

https://blender.stackexchange.com/questions/75223/finding-vertices-in-a-vertex-group-using-blenders-python-api

(画像)

https://blender.stackexchange.com/questions/75223/finding-vertices-in-a-vertex-group-using-blenders-python-api

(画像)

# A. 各頂点で、関連の頂点グループの情報が付いています

・・・

本気かいこいつ

```
>>> bpy.context.object.data.vertices[0].groups[0].group
1

>>> bpy.context.object.data.vertices[1].groups[0].group
1

>>> bpy.context.object.data.vertices[12451].groups[0].group
2
```

まじかよ・・・しかも、グループは添え字で・・・

じゃぁ、判ったんですか？

各頂点グループの情報を回収したいなら、Meshの各頂点を確認しないとだめです！

素晴らしいAPIでしょう！

VRCHAT用のモデルは普段10万頂点を超えてないのはありがたいです。
数千万頂点のモデルでは狂うと思います。

# 難点2 : Python

さぁ、繰り返しましょう

目的は :

- [x] 消すボーンを決める
- [x] ウェイトペイントを受けるターゲットボーンを決める
- [x] それぞれの頂点グループを見つける
- [ ] 消しているボーンの頂点グループの頂点をターゲット・ボーンの頂点グループに追加する

最後のステップをやり遂げるために、複数の方法がありますね。

でも、スクリプトを制作していた時には、
「後はステップが増やすかもしれませんから、頂点グループの頂点リストを簡単に取れるようにしましょうか」と考えました。

つまり、目的はこうになります：

- [x] 消すボーンを決める
- [x] ウェイトペイントを受けるターゲットボーンを決める
- [ ] 頂点グループの頂点リストを簡単に取れるようにする
- [x] それぞれの頂点グループを見つける
- [ ] 消しているボーンの頂点グループの頂点をターゲット・ボーンの頂点グループに追加する

「頂点グループの頂点リストを簡単に取れるようにする」のために、次のやり方を決めましたね：

- リストを作って
- そのリストのサイズを頂点グループの数と決めて
- 各添え字で、Setを用意して
- 各頂点をスキャンして、見つかったグループの添え字づつ、その頂点を関連するSetに追加します。

(???)
そしたら、後は、頂点グループを頂点リストを簡単に取ることが出来ます。

そして、グループを結合したいなら、「VertexGroup.add」のメソッドを使って、
グループAの頂点Setを頂点グループBに追加して、完成です。

じゃぁ、固定サイズのリストの作り方・・・

まずはどうしてサイズを固定したいのは、Python語では
配列の添え字はサイズを超えたら、Exceptionが出てます。

C語, C#語, Java語, そういう仕組みが割りますが、
スクリプトの言語では本当に面倒くさいです。

じゃぁ、Python語で、どうやって配列のサイズを決めることが出来ますか？

画像

原始的な方法と思いますが、でもそれで行けるなら使いましょう。

どうして、list(サイズ)みたいなConstructorがないんですか・・・

えと、何をしたかったっけ：

- リストを作って
- そのリストのサイズを頂点グループの数と決めて
- 各添え字で、Setを用意して
- 各頂点をスキャンして、見つかったグループの添え字づつ、その頂点を関連するSetに追加します。

では：

- リストを作って
- そのリストのサイズを頂点グループの数と決めて
- 各添え字で、Setを用意して

先の方法を使えば、こうなりますね：

```
cached_groups = [set()] * max_vertex_groups
```

残りは

- 各頂点をスキャンして、見つかったグループの添え字づつ、その頂点を関連するSetに追加します。

```
for vertex in mesh.data.vertices:
    for group_info in vertex.groups:
        group_index = group_info.group
        cached_groups[group_index].add((vertex.index, group_info.weight))
```

すべてが揃っている！

じゃぁ、本番に戻って：

- [x] 消すボーンを決める
- [x] 頂点グループの頂点リストを簡単に取れるようにする
- [x] ウェイトペイントを受けるターゲットボーンを決める
- [x] それぞれの頂点グループを見つける
- [ ] 消しているボーンの頂点グループの頂点をターゲット・ボーンの頂点グループに追加する

頂点グループの頂点の情報は既に保存しましたから、さぁ、本番だ！

```
for selected_bone in selected_bones:
  if selected_bone == active_bone:
    continue

  # Some bones have no associated vertex group.
  # Skip it if that's the case
  if selected_bone.name not in mesh.vertex_groups:
    continue

  # For each cached vertex data, add it to the target VertexGroup
  vertex_group = mesh.vertex_groups[selected_bone.name]
  for vertex_data in cached_groups[vertex_group.index]:
    target_vertex_group.add([vertex_data[0]], vertex_data[1], 'ADD')
```

できたぜ！じゃぁ、結果をみましょう・・・

（画像）

！！！？　Headのウェイトペイントが体全体を巻き込んでしまったけど！？

はぁ？

ん、問題は頂点は多いんですね。
Setの中身を確認しづらいです。

一応、キャッシュのSetのサイズを確認しましょう

・・・

おおいみたいですね。

っていうか、全部のSetのサイズは同じ見たいです

・・・

・・・

まさか

一時間後

解ってしまった。

あの原始的方法は罠だった！！

```
cached_groups = [set()] * max_vertex_groups
```

って

```
cache_set = set()
cached_groups = []
for _ in range(0, max_vertex_groups):
  cached_groups.add(cache_set)
```

と同じです！

つまり、こうすると：

```
cached_groups = [set()] * max_vertex_groups
```

かくインデックスのsetのリファレンスは同じとなります！

```
>>> cached_groups[0] == cached_groups[n]
True
>>> cached_groups[0].add(1)
>>> cached_groups[n]
{1}
```

だから、StackOverflowを信用できませんよ！

じゃぁ、前の質問に戻りましょう：

Python語で、どうやって配列のサイズを決めることが出来ますか！！？

高技術の質問を訊いている訳じゃないです。

簡単な方法があるはずです！

そして、答えは：

```
cached_groups = [set() for _ in range(max_vertex_groups)]
```

```
cached_groups = [set() for _ in range(max_vertex_groups)]
```

ねぇ、Python語を触ってない人:
このシンタックスは読めますか？
結果を想像できますか？

わたくしは想像できませんね！

なんなんだこのシンタックス！

まぁ、とりあえず、この方法を使って、先の問題を解決して、もう一度試したら：

出来た！

もう一度試して・・・

よっしゃー！！！

（喜びの舞）

はい！じゃぁ、後は、右クリックメニューでどうやってそのスクリプトを呼ぶことが出来ることを探して・・・

(Code)

出来た！

はい、じゃぁ、後はスクリプトをGithubでアップロードして、必要な時使いましょう。
別に、毎日ボーンを減らしている訳じゃないんですから、これでいいんだ。

# ちゃんとしたアドオンへ

この状態では、

「じゃぁ、もうスクリプトは出来ていますから、
他のプロジェクトに集中しましょう。」

という流れになりす、普段。

## ちゃんと製品にしませんか？

だるいです。

コアー機能がちゃんと動いても、公開できる製品になる道は

ながい！

コアー機能が動けるまで、見た通り、色んな問題に逢いましたな！

やることは少なくても、あのAPIとプログラミング言語のため、想像出来なかった問題がに遭って、時間を失いましたね。

ですから、「皆さんが円滑に使えるように、スクリプトを改善する」の時間を

計れない！

時間を計れないんですから、不安です！

また妙な問題に合ったら、どれだけの時間を失いますか？

そして、結局出来ても、誰が使いますか、あれは？

紹介したスクリプトは、そのままで使いたいなら、
Blenderの「Scripting」タブでコピペして、「実行」ボタンをクリックするしかないです。

ちょっと手間がかかりますが、
自分はモデルの最適化を、一か月ほど、二回ぐらいやりますね。

ですから、その手動ステップを抜いても、メリットはほとんどないですね。

じゃぁ、まずは、どうしてスクリプトを開発したんですか？と言われたら、
Blenderでどうやってスクリプトを開発できることを学びたかったね。

自分で使うツールをどうやってカスタマイズ方法を判るのは、いつでも役に立ちますね。

## それでも、公開品にしましたね

ある友達がモデルを同じく最適化したかったんですね。

その友達にスクリプトを渡したが、使うために友達がめっちゃ苦労していることを見て、
「だめこりゃ。
これは渡すものではない。
ちゃんとしないとだめだ。」
ということ結論にたどって、ちゃんとした製品になるように、そのスクリプトを改善して始めました。

# 動いている製品を宣伝できる公開品まで

そのスクリプトを公開できるまで、やることは：

- 簡単なインストールできるようにする
- 面倒くさい所を抜いて、円滑にする
- UIを考えずに使えるにする
- テストして、頼れるようにする (毎日で使う)
- 文章を作る
- 言語に対応する
- 紹介動画を取って、編集して、字幕を追加する
- 宣伝する

## 面倒くさい所を抜いて、円滑にする

（だいふくさんのMDDを紹介して）

## 簡単なインストールできるようにする

Blenderなら、そのスクリプトをAddonにしないとだめです。

意外と、Scriptingタブのエディターは「Addon」のテンプレートを持っています。

にしても、文章をちゃんと読まないとだめですね。
`"category"` をなにすればいいんですか？自由ですか？
あ、そうじゃなそうです。

（画像）

で、 `"support"` でなにを書けばいいんですか？

・・・

後は、インストールして、抜いて、もう一度インストールして・・・

「右クリックメニューのアイテムが複製された・・・なんだこりゃ・・・」

## UIを改善

今回はUIはほとんど出来ていましたね。
既に右クリックメニューで使えますから、改善する所もなさそうです。

問題はエラーメッセージがちゃんと表示されていないかったね。
その問題を解決したら、出来ていましたね。

## テストして、頼れるようにする

### 毎日で使う

これはね、自分のツールをテストしても、
毎日で使わないと、色んな問題を見逃しますね。

ですから、自分のツールを使って、すべての使っている
アバターのモデルでためしていましたね。

そして、後は、Addonをアンインストールして、
もう一度インストールして、それから作業をして、
結果が出ることを確認するべきですね。

## 文章を作る

## スクリーンショットを追加する

「画像を見てわかるでしょう！」と思っても、手順ぐらいを書く方がいいんです。
その手順でスクリーンショットを追加したら、大体皆さんが解りますね。

後は、遭える問題を説明しないとだめです。
遭える問題は：

- エラーメッセージを用意ところ
- 解っている不具合（Known bugs）

そのエラーメッセージが普段見えない場所で現れるなら、まずはその場所を説明しないとだめです。

例えば、UnityのAddonでは、「Consoleをご覧ください」を説明しないと、色んな人はそこで表示されるメッセージに気を付けません。

## 紹介動画を取って、編集して、字幕を追加する

苦労して、皆さんに渡せるような製品を作ると、
だいたいちゃんと宣伝したいんですね。

製品を紹介するために、動画は一番です。

動画を見て、興味のあるユーザーがそのツールを
簡単に使えるのかどうか確認できますね。

編集している時は、カットは最低限にすることをお勧めします。

動画編集で、めっちゃ長いプロセスを一分で出来るように見せることが出来ますが、ユーザーがその紹介されている製品を使うと、動画で騙されたの感じになりますと：

- その背品をもう使いません。
- 勝ったなら、返金を要求する可能性は高いです。
- 製作者をもう信用しません。

ですから、動画編集はほどほどに。

TV広告じゃなく、製品紹介です。また解説動画です。

ソフトをインストールした後で、動画を見ながら、自分の作業を進めるなら、最適です。

にしても、製品をその動画がなくても簡単に使えるようにしないとだめです。

## 宣伝する

一番やっていない所です。

Twitterで宣伝することは・・・できますが、改善するほど、その改善をTwitterで報告したら、ユーザーが増えますね。

でも、Tweetは一日後忘れます、だいたい。

# 結果

結果はAddonを出来ました！
そして、Boothでリリースしました。

https://voyage-vrsns.booth.pm/items/4077659

そしてGithubでもね：

https://github.com/vr-voyage

伝えたかった感想は：

- 自分のツールを作る時には、やりたいことを解っても、色んな予想できない問題に遭えますね。
- ツールがちゃんと動いても、Booth・Gumroadでリリースできるまで、道がまだ長いです。
  これから本番って感じです。

にしても、出来ていたら、その円滑で使えるようなソフトを毎日使いたい感じになりますね。
そして、ユーザーが自分の製品を自分で宣伝しているときに、モチベーションがかなり上がりますね。

Fin