MARUI-PlugIn

サポートコミュニティ

スクリプティング・カスタマイゼーション

MARUI には、独自の VR ユーティリティを構築したり、必要に応じて UI をカスタマイズしたりするための一連の Maya コマンドが用意されています。

VR デバイスデータの読み込み

下記コマンドを使用して、VR デバイス（ヘッドセットまたはコントローラ）の現在位置を読み取ることができます。
MARUI_UI -dt <device>;
MARUI_UI -dr <device> <rotationOrder>;
<device> は "rightController" "leftController" "hmd”のいずれかになります。
そして <rotationOrder> は "xyz" "xzy" "yxz" "yzx" "zxy" "zyx”のいずれかです。
以下が MEL でコマンドを使用する方法例です。

```
// コントローラー位置の照会
float $t[] = `MARUI_UI -dt "rightController"`;
// コントローラの回転
float $r[] = `MARUI_UI -dr "rightController" "zxy"`;
// 「zxy」は希望する回転順序
```

ウィジット

MARUI のユーザーインターフェイス要素は「ウィジット」と呼ばれます。
利用可能なウィジェットにはいくつかの種類があります。
そしてすべて MARUI_Widget コマンドで作成および編集されます。

MEL:
MARUI_Widget -create <widgetType> <additionalParameters> "myWidgetName";

Python:
MARUI_Widget("myWidgetName", create=<widgetType>, <additionalParameters>)

タイムウィジット

タイマーウィジットは、目覚まし時計や定期的なコールバックに似ています。

設定された時間が経過すると、設定した MEL または Python コマンドが呼び出されます。
これを活用して、スクリプトの定期的な更新を行ったり、一定の時間を待機させたりことができます。
以下のパラメータが利用可能です。

command <string>
タイムアップになると呼び出されるコマンド。
独自の MEL または Python 関数を定義したり、ここに任意の Maya コマンドを挿入したりすることができます。
操作が終了するまで MARUI は操作を続行できないため、コマンドが遅いと VR が使用できなくなる可能性があります。

python <boolean>
コマンドが Python(true)コマンドか MEL(false)コマンドのどちらであるかのパラメーターを設定しないと、MARUI は MEL コマンドを予想します。

time <integer>
タイマがミリ秒単位でコマンドをトリガーするまでの時間。
このパラメーターを省略すると、デフォルト値の 1000ms（1 秒）が使用されます。
時間を 0 に設定すると、コマンドはすべてのフレームで実行されます（毎秒 60 回まで）。

number <integer>
タイマー自体を繰り返す回数。デフォルトは 1
設定した回数が完了すると、Timer ウィジェットが破棄され、値を変更できなくなります。
ただし、同じ名前の新しいタイマーを作成することはできます。
数値を 0 に設定すると、タイマーは編集または MARUI を終了するまで無期限に続きます。

活用事例

Python:
import maya.cmds as cmds

```
# 10 回、200ms ごとに 1 回+ のタイマーを「myTimer」という名前で設定
cmds.MARUI_Widget("myTimer", create="timer", command="print(\"timer\")", python=True, number=10, time=200)
# 1 /秒の周波数で無期限に続行するようにタイマーを変更する
cmds.MARUI_Widget("myTimer", number=0)
cmds.MARUI_Widget("myTimer", time=1000)
# タイマーの設定を解除（削除）
cmds.MARUI_Widget("myTimer", command="")
```

MEL:

```
// 200 回ごとに 10 回、10 回呼び出されるタイマーを「myTimer」という名前で設定
MARUI_Widget -create "timer" -command "print(\"timer\");" -python false -number 10 -time 200 "myTimer";
// 1 /秒の周波数で無期限に続行するようにタイマーを変更する
MARUI_Widget -number 0 "myTimer";
MARUI_Widget -time 1000 "myTimer";
// タイマーの設定を解除（削除）
MARUI_Widget -command "" "myTimer";
```

コマンドウィジット

コマンドウィジットは、MEL または Python コマンドを HTC Vive または Oculus Touch コントローラのボタンにバインドするために使用されます。
コマンドの使用方法に応じて、ボタンをクリックまたはドラッグすると、コマンドが呼び出されます。
コマンドウィジットをボタンにバインドするには、MARUI_UI -map コマンドを使用します。
下記のパラメーターが活用可能です。

icon <string>
コマンドをシンボル化するためのディスク上のアイコンイメージファイル。
このアイコンは、コントローラーをバインドするボトム部分に表示されます。
サポートされている画像タイプは jpg、png、bmp、tga です。
アイコンを指定しない場合は、代わりにコマンドの名前が表示されます。

command <string>
<event>に対して実行する MEL または Python コマンド。
これは、独自の自己定義関数または Maya 関数にすることができます。

event <string>
コマンドを呼び出すイベント。
使用可能なイベントは、「click」「dragStart」「dragContd」「dragStop」です。

python <boolean>
呼び出すコマンドが Python (true) であるか MEL (false) であるか。
このパラメータを省略すると、MEL が仮定されます。

例

MEL:

```
// 「myCommand」という名前の新しいコマンドメニューを作成
MARUI_Widget -create "command" -icon "C:/Temp/icon_command.bmp" "myCommand";
//使用可能な全ての event ごとに異なるコマンドを設定
MARUI_Widget -event "click"     -command "print(\"click\");"      -python true "myCommand";
MARUI_Widget -event "dragStart" -command "print(\"drag start\");" -python true "myCommand";
MARUI_Widget -event "dragContd" -command "print(\"drag contd\");" -python true "myCommand";
MARUI_Widget -event "dragStop"  -command "print(\"drag stop\");"  -python true "myCommand";
// 左コントローラー（Oculus Touch）の"A" もしくは "X"ボタンへコマンドウィジットをマップ
MARUI_UI -map "left" "AX" "myCommand";
// click-event へ設定されたコマンドのクエリ
MARUI_Widget -q -event "click" -command "myCommand";
// Python コマンドとして扱われるコマンドのクエリ
MARUI_Widget -q -event "click" -python "myCommand";
```

Python:

```
import maya.cmds as cmds

# "myCommand"という新しいコマンドウィジットを作成する
cmds.MARUI_Widget("myCommand", create="command" , icon="C:/Temp/icon_command.bmp")
# 使用可能なすべてのコマンドのコマンドを設定する
cmds.MARUI_Widget("myCommand", event="click"    , command="print(\"click\")"     , python=True)
cmds.MARUI_Widget("myCommand", event="dragStart", command="print(\"drag start\")", python=True)
cmds.MARUI_Widget("myCommand", event="dragContd", command="print(\"drag contd\");",
python=False)
cmds.MARUI_Widget("myCommand", event="dragStop" , command="print(\"drag stop\")" , python=True)
# このコマンドウィジットを左のコントローラー "A"（または "X"）ボタンに接続します。
cmds.MARUI_UI(map=["left", "AX", "myCommand"])
```

マーキングメニューウィジット

マーキングメニューは、MARUI が Maya のツールや機能にアクセスするために使用する円形のメニューです。
マーキングメニューウィジットを使用すると、独自のマーキングメニューを作成し、それらをコントローラ上の任意のボタンに割り当てることができます。
コマンドウィジットをボタンにバインドするには、MARUI_UI -map コマンドを使用します。

MEL:
MARUI_Widget -create "markingmenu" <additionalParameters> <menuName>;
MARUI_Widget -addElement <additionalParameters> <menuName>;
MARUI_UI -map <side> <button> <menuName>;

Python:
MARUI_Widget(<menuName>, create="markingmenu", icon="<pathToIconFile")
MARUI_Widget(<menuName>, addElement=True, <additionalParameters>)
MARUI_UI(map=[<side>, <button>, <menuName>])

以下のパラメーターが活用可能です。

icon <string>
コマンドをシンボル化するためのディスク上のアイコンイメージファイル。
このアイコンは、コントローラーをバインドするボトム部分に表示されます。
サポートされている画像タイプは jpg、png、bmp、tga です。
アイコンを指定しないと、代わりにコマンドの名前（または要素のタイトル）が表示されます。

title <string>
要素の名前または説明。
要素の上にマウスを置くと、これが表示されます。

command <string>
<event>に対して実行する MEL または Python コマンド。
これは、独自の自己定義関数または Maya 関数にすることができます。

python <boolean>
呼び出すコマンドが Python (true) であるか MEL (false) であるか。
このパラメータを省略すると、MEL が仮定されます。

例

MEL:

```
// "myMenu"というマーキングメニューウィジットを作成
MARUI_Widget -create "markingmenu" -icon "C:/Temp/icon_menu.bmp" "myMenu";
// いくつかの要素を追加
MARUI_Widget -addElement -title "command one" -icon "C:/Temp/icon1.jpg" -command "print(\"one\");" -
python true "myMenu";
MARUI_Widget -addElement -title "command two" -icon "C:/Temp/icon2.png" -command "print(\"two\");" -
python true "myMenu";
MARUI_Widget -addElement -title "command three" -icon "C:/Temp/icon3.tga" -command
"print(\"three\");" -python true "myMenu";
// メニューを左コントローラーの「親指スティックを左に押す」動作にバインド
MARUI_UI -map "left" "StickLeft" "myMenu";
```

Python:
import maya.cmds as cmds

```
# "myMenu"というカスタムマーキングメニューを作成
cmds.MARUI_Widget("myMenu", create="markingmenu", icon="C:/Temp/icon_menu.bmp")
# いくつかのメニュー項目を追加
cmds.MARUI_Widget("myMenu", addElement=True, title="command one",  icon="C:/Temp/icon1.jpg",
command="print(\"one\");",  python=True)
cmds.MARUI_Widget("myMenu", addElement=True, title="command two",  icon="C:/Temp/icon2.png",
command="print(\"two\");",  python=True)
cmds.MARUI_Widget("myMenu", addElement=True, title="command three", icon="C:/Temp/icon3.tga",
command="print(\"three\");", python=True)
# メニューを左コントローラーの「親指スティックを左に押す」動作にバインド
cmds.MARUI_UI(map=["left", "StickLeft", "myMenu"])
```

マッピングウィジットボタン

MARUI_UI コマンドの setmapping または map パラメーターを使用して、ウィジットをコントローラー上のボタンにバインドすることができます。

MEL:
MARUI_UI -setmapping <side> <button> <widget>;

Python:
MARUI_UI(map=[<side>, <button>, <widget>])
<function>を<side>コントローラの<button>にバインドします。
例として、左のコントローラートリガーボタンを MARUI Omni-Tool にバインドするには、次のコマンドを使用します。
MARUI_UI -setmapping "left" "Trigger" "Omni";
<side> は”*left*" もしくは "*right*” になります。
使用している VR デバイスごとに、以下のボタンを使用できます。

HTC Vive

HTC Vive コントローラーでは、以下のボタンを使用できます。

- *SystemButton* : システムボタン（タッチパッドの上領域）
- *MenuButton* : メニューボタン（タッチパッドの下領域）
- *Grip* : グリップまたはショルダーボタン（ハンドルを握る）
- *DPadLeft* : タッチパッドの左領域
- *DPadUp* : タッチパッドの上領域
- *DPadRight* : タッチパッドの右領域
- *DPadDown*" : タッチパッドの下領域
- *Trigger* : トリガーボタン

Oculus Rift

Oculus Touch コントローラーでは、以下のボタンを使用できます。

- *StickRight* : 右に親指スティックを押す
- *StickLeft* : 左に親指スティックを押す
- *StickUp* : 前に親指スティックを押す
- *StickDown* : 後ろに親指スティックを押す

- *AX* : A (右コントローラー) と X (左コントローラー) ボタン
- *BY* : B (右コントローラー) と Y (左コントローラー) ボタン
- *Thumb* : 親指スティックを押し込む
- *Shoulder* : グリップまたはショルダーボタン（ハンドルを握るように押す）
- *Grip* : グリップまたはショルダーボタン（ハンドルを握るように押す）
- *Trigger* : トリガーボタン

LeapMotion

LeapMotion デバイスでは、以下のボタンを使用できます。

- ピンチ : 親指と人差し指でつまむ
- 握る : 拳を握る

Build-in の MARUI ウィジット

独自のウィジェットとは別に、MARUI_UI -map コマンドを使用して、組み込み MARUI 関数を任意のボタンにバインドすることができます。
以下のウィジェットが利用できます。

- *Trigger* : 選択されたツールのデフォルトのトリガーアクション。
- *Select* : 選択（レイキャスティングまたは "ポインティング"）
- *Select_Proximity* : コントローラまたは容積選択の近くで選択する
- *Move* : 3D 移動ツール (translation).
- *Rotate* : 3D 回転ツール
- *Scale* : 3D スケールツール
- *Omni* : 6DOF "Omni" ツール
- *Menu_Tool* : ツールを選択するツールメニュー
- *Menu_Mode* : コンポーネントモード（Vertex、Edge、Object、...）を選択するモードメニュー
- *Menu_Action* : アクションメニュー（「削除」、「元に戻す」、「やり直し」など）
- *Menu_Anim* : アニメーションメニュー（キーフレームの設定など）
- *Menu_MARUI* : MARUI メニュー（「look-through-selected」「シェーディング」など）
- *Menu_PolyTools* : ポリゴンモデリングツール
- *Shelf* : Maya シェルフ
- *Navi* : 設定で選択されたナビゲーション
- *Navi_GrabAir* : Grabbing-the-air ナビゲーション
- *Navi_Tumble* : Maya-mouse-style tumbling ナビゲーション
- *Navi_Joystick* : Joystick-style ナビゲーション
- *TimeSlider* : MARUI タイムスライダ
- *Shift* : Shift ボタン

タスクモード

モデリング・スカルプティング・アニメーション・ライティング/レンダリングにはすべて、特化下ツールや機能が必要です。
これらのタスクのそれぞれに有用な UI を提供するために、プログラム可能なタスクモードを追加しました。
コントローラーメニュー（モデリング、アニメーション、ライティング/レンダリング）の基本的なタスクモード（右手の Vive コントローラーでは D パッド "上部分”、右手の Oculus Touch コントローラーではスティックを "上に"）を選択できます。
それぞれのモードには異なるメニューと機能があります。
次のコマンドを使用して、独自のタスクモードを定義することもできます。

MARUI_UI -listtaskmodes;
現在利用可能なタスクモードのリストを返します。デフォルトのタスクモードは、【モデリング】【アニメーション】【ライティング/レンダリング】です。

MARUI_UI -taskmode "name_of_taskmode";
指定された名前のタスクモードを選択します。
タスクモードがまだ存在しない場合は、作成されます。