コンポーネント事典

この事典では、App Inventor で使えるすべてのコンポーネントを解説します。App Inventor でアプリをつくる際に手元に置いておくと役に立つでしょう。

※この事典の内容は、2011 年 5 月時点のものです。今後、App Inventor のアップデートによって、コンポーネントの追加や変更が行われる可能性があります。

基本コンポーネント

Button 

Button コンポーネントは、ユーザーがクリックできるボタンを表示するコンポーネントです。これまでの説明でも何度も登場した、いちばんよく使うコンポーネントです。

プロパティ

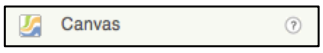
BackgroundColor	ボタンの背景色です。
Enabled	チェックされている場合、ボタンをクリックできます。
FontBold	チェックされている場合、ボタンのテキストが太字になります。
FontItalic	チェックされている場合、ボタンのテキストが斜体になります。
FontSize	ボタンのテキストの大きさです。
FontTypeface	ボタンのテキストのフォントです。
Image	ボタンに表示する画像です（*1）。
Text	ボタンのテキストです。
TextAlignment	ボタンのテキストの表示位置です。
TextColor	ボタンのテキストの色です。
Visible	チェックされている場合、ボタンが表示されます。
Width	ボタンの幅です。
Height	ボタンの高さです。

***1**　このプロパティに画像を設定した場合は、その画像サイズと同じ大きさでボタンが表示されます。クリック（タップ）できる画像を表示したい時は、あとで登場する Image コンポーネントではなく、Button コンポーネントを使い、このプロパティに画像をセットします。

イベント

Click()	ユーザがボタンをクリックした時に呼び出されます。
GotFocus()	ボタンがフォーカスされた時に呼び出されます。
LostFocus()	ボタンがフォーカスされなくなった時に呼び出されます（*2）。

***2**　フォーカス関係のイベントは、トラックボールがある端末などでしか使えないので、あまり使いどころがありません。

Canvas 

Canvas コンポーネントは、点や線や円を描くことができるキャンバスを表示するコンポーネントです。

タッチしたり、ドラッグした時のイベントがあるので、指でなぞって絵を描いたりできます。背景に画像をセットできるので、写真に落書きする Android アプリも作れそうです。

キャンバスの状態（点や線を描いた状態）を画像ファイルに保存することもできます。

プロパティ

BackgroundColor	キャンバスの背景色です。
BackgroundImage	キャンバスの背景画像です。
FontSize	キャンバスにテキストを描く時のテキストの大きさです。
LineWidth	キャンバスに点や線や円を描く時の線の太さです。
PaintColor	キャンバスに点や線や円を描く時の線の色です。
TextAlignment	テキストの位置という意味なのですが、どれを選んでも変化がなかったので謎です（*3）。
Visible	チェックされている場合、キャンバスが表示されます。
Width	キャンバスの幅です（*4）。
Height	キャンバスの高さです（*5）。

***3**　App Inventor はベータ版ということもあって、このような謎のプロパティがあったりしますが気にしないでください。

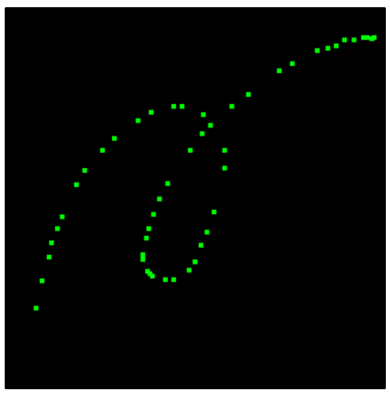
***4**　キャンバスを画面全体に表示したい場合は、Fill parent を選択します。

***5**　キャンバスの高さは、Fill parent を選択しても画面全体の高さにならないので、pixels で高さを指定します。もし、正方形にしたい場合は、320pixels に設定してください。

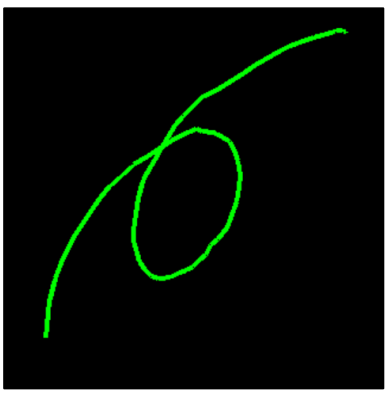
イベント

Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY, boolean draggedSprite)	キャンバス上でドラッグ（***6**）した時（画面をタッチして指でなぞった時）に呼び出されます（***7**）。
currentX	ドラッグ中の現在の X 座標です。
currentY	ドラッグ中の現在の Y 座標です。
draggedSprite	スプライト（***8**）をドラッグしている時は true（***9**）、そうでない時は false です。
Touched	キャンバス上をタッチした時に呼び出されます。
x	タッチした X 座標です。
y	タッチした Y 座標です。
touchedSprite	スプライトをタッチした時は true、そうでない時は false です。

***6**　Dragged イベントはドラッグしている間に何回か呼び出されます。例えば、Dragged イベントが呼び出されたら、現在の X,Y 座標に点を描画するようにすると、画面 1 のようになります。これを前回の X,Y 座標から現在の X,Y 座標に線を描画するようにすると画面 2 のようになります。つまり、Dragged イベントはドラッグ中に「ちょこちょこ」と呼び出されるイベントなので、そのまま点を描画しただけでは線にならず、なぞった部分に線を描きたい時などに、この prevX,prevY プロパティの前回の X,Y 座標から現在の X,Y 座標に線を描画したい時などに使います。



▲画面 1



▲画面 2

***7** イベントのブロックで右側にピースがあるものは初めて登場しました。この右側のピースには、このイベントが起きた時の様々な値が入っています。それぞれの意味は以下の通りです。

startX	ドラッグを開始した X 座標です。
startY	ドラッグを開始した Y 座標です。
prevX	前回の Dragged イベントの X 座標です。
prevY	前回の Dragged イベントの Y 座標です。

***8** スプライトとは、あとで登場する Ball コンポーネントや ImageSprite コンポーネントのことです。

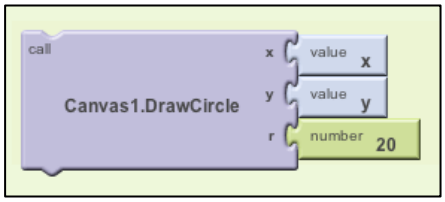
***9** true/false とは、真偽値と呼ばれるもので、true は yes や on、false は no や off というイメージで覚えてください。ある条件によって何かしたい時や on/off を切り替えたい時によく使う値です。例えば、ボタン（Button コンポーネント）をクリック時に、画像（Image コンポーネント）を表示させたい時には、始めに Image コンポーネントの Visible プロパティのチェックを外しておいて非表示にしておいて、Button コンポーネントの Click イベントで Image コンポーネントの Visible プロパティに true をセットして表示に切り替えるということを、よくやります。つまり、今までに説明していた Visible などのプロパティのチェックのありなしは、あり（true）なし（false）を設定していたのです。

メソッド

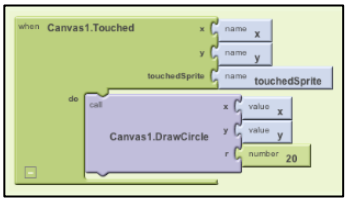
Clear()	キャンバスをクリアします。
DrawCircle(number x, number y, number r)	キャンバスの座標 x,y に半径 r の円を描きます（*10）。
x	描きたい円の X 座標
y	描きたい円の Y 座標
r	描きたい円の半径
DrawLine(number x1, number y1, number x2, number y2)	キャンバスの座標 x1,y1 から座標 x2,y2 まで線を描きます。
x1	描きたい線の始点の X 座標
y1	描きたい線の始点の Y 座標
x2	描きたい線の終点の X 座標
y2	描きたい線の終点の Y 座標
DrawPoint(number x, number y)	キャンバスの座標 x,y に点を描きます。
x	描きたい点の X 座標
y	描きたい点の Y 座標
DrawText(text text, number x, number y)	キャンバスの座標 x,y にテキストを描きます。

text	描きたいテキスト
x	描きたい点の X 座標
y	描きたい点の Y 座標
DrawTextAtAngle(text text, number x, number y, number angle)	キャンバスの座標 x,y にセットされた角度でテキストを描きます。傾いたテキストを描くことができます。
text	描きたいテキスト
x	描きたい点の X 座標
y	描きたい点の Y 座標
angle	描きたいテキストの角度
Save()	キャンバスを画像ファイルとして SD カードに保存します（*11）（*12）。
SaveAs(text fileName)	キャンバスを画像ファイルとして SD カードにセットされたファイル名で保存します（*13）。
fileName	保存するファイル名

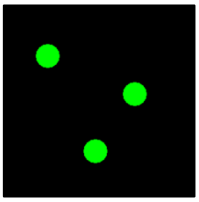
*10　メソッドのブロックで右側に凹みがあるものは初めて登場しました。この凹みにブロックを差し込むことでメソッドに指示を行います。例えば、タッチされた X,Y 座標に半径 20px の円を描きたい場合は、画面 3 のようにブロックを差し込みます。Touched イベントのブロックと組み合わせると画面 4 のようになります。これで画面 5 のようにタッチした X,Y 座標に半径 20px の円を描くことができます。



▲画面 3



▲画面 4


▲画面 5

*11　画像ファイルは、/mnt/sdcard/My Documents/Pictures/ というディレクトリに、app_inventor_1303324313210.png というファイル名で保存されます。

*12　執筆時点(2011/04/21)では、キャンバスのサイズが大きいとエラーになり保存できないバグがあります。Save メソッドを使いたい場合は、エラーがでないように Width プロパティと Height プロパティを 270px 以下に設定してください。機種によっては、さらにサイズを小さくする必要があるかもしれません、少しずつサイズを小さくして試してみてください。

*13　例えば、fileName に gabu という文字をセットした場合は、自動的に拡張子が追加されて

/mnt/sdcard/というディレクトリに、gabu.png というファイル名で保存されます。

CheckBox 

CheckBox コンポーネントは、ユーザがクリックできるチェックボックスを表示するコンポーネントです。アンケートなど何かをチェックする形式で入力したい時に使います。

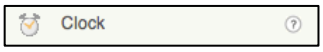
プロパティ

BackgroundColor	チェックボックスの背景色です。
Checked	チェックボックスのチェック状態です。あらかじめチェックされた状態で表示することができます。
Enabled	チェックされている場合、チェックボックスをクリックできます。
FontBold	チェックされている場合、チェックボックスのテキストが太字になります。
FontItalic	チェックされている場合、チェックボックスのテキストが斜体になります。
FontSize	チェックボックスのテキストの大きさです。
FontTypeface	チェックボックスのテキストのフォントです。
Text	チェックボックスのテキストです。
TextColor	チェックボックスのテキストの色です。
Visible	チェックされている場合、チェックボックスが表示されます。
Width	チェックボックスの幅です。
Height	チェックボックスの高さです。

イベント

Click()	ユーザがチェックボックスをクリックした時に呼び出されます。
GotFocus()	チェックボックスがフォーカスされた時に呼び出されます。
LostFocus()	チェックボックスがフォーカスされなくなった時に呼び出されます（*14）。

***14**　フォーカス関係のイベントは、トラックボールがある端末などでしか使えないので、あまり使いどころがありません。

Clock 

Clock コンポーネントは、時間を扱うコンポーネントです。画面には表示されないコンポーネントです。タイマーという一定間隔で呼び出されるイベントを使って、一定間隔で何か動作する機能が作れます。現在日時を取得したり、日付や時間の計算ができます。

プロパティ

TimerInterval	Timer イベントの間隔です。ミリ秒で設定します。
TimerEnabled	チェックされている場合、Timer イベントが呼び出されます。
TimerAlwaysFires	チェックされている場合、アプリをホームボタンなどで閉じても Timer イベントが呼び出されます。

イベント

Timer()	TimerInterval プロパティで設定した間隔で呼び出されます。

メソッド

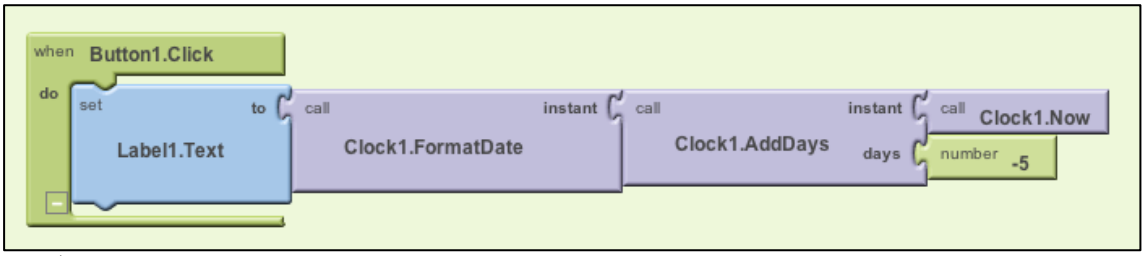
AddDays(Calendar instant, Number days)	ある特定の日時に日数を足します。これを使うと何日後の日時か、または、何日前の日時かが計算できます。
instant	特定の日時
days	日数（*15）
AddHours(Calendar instant, Number hours)	ある特定の日時に時間を足します。これを使うと何時間後の日時か、または、何時間前の日時かが計算できます。
instant	特定の日時
hours	時間
AddMinutes(Calendar instant, Number minutes)	ある特定の日時に分を足します。これを使うと何分後の日時か、または、何分前の日時かが計算できます。
instant	特定の日時
minutes	分
AddMonths(Calendar instant, Number months)	ある特定の日時に月を足します。これを使うと何ヶ月後の日時か、または、何ヶ月前の日時かが計算できます。
instant	特定の日時
months	月

AddSeconds(Calendar instant, Number seconds)	ある特定の日時に秒を足します。これを使うと何秒後の日時か、または、何秒前の日時かが計算できます。
instant	特定の日時
seconds	秒
AddWeeks(Calendar instant, Number weeks)	ある特定の日時に週を足します。これを使うと何週間後の日時か、または、何週間前の日時かが計算できます。
instant	特定の日時
weeks	週
AddYears(Calendar instant, Number years)	ある特定の日時に年を足します。これを使うと何年後の日時か、または、何年前の日時かが計算できます。
instant	特定の日時
years	年
DayOfMonth(Calendar instant)	特定の日時の日にちだけ取得します。例えば、instant が 2011/04/17 であれば、17 が取得できます。
instant	特定の日時
Duration(Calendar start, Calendar end)	二つの特定の日時の差を計算します。差はミリ秒で取得できます。
start	差を計算する元の特定の日時
end	差を計算する先の特定の日時
FormatDate(Calendar instant)	特定の日時の年月日を 2011/04/17 のようなテキストに変換します（*16）。
instant	特定の日時
FormatDateTime(Calendar instant)	特定の日時の年月日と時刻を 2011/04/17 12:30:15 のようなテキストに変換します。
instant	特定の日時
FormatTime(Calendar instant)	特定の日時の時刻を 12:30:15 のようなテキストに変換します。
instant	特定の日時
GetMillis(Calendar instant)	1970 年からの特定の日時までに経過したミリ秒を取得します。これはいかにもプログラミング的なので、あまり使わないかもしれません。

instant	特定の日時
Hour(Calendar instant)	特定の日時の時間だけ取得します。例えば、instant が 2011/04/17 12:30:15 であれば、12 が取得できます。
instant	特定の日時
MakeInstant(Text from)	テキストから日時を作ります（*17）（*18）。
from	日時に変換したいテキスト
MakeInstantFromMillis(Number millis)	ミリ秒の時間から日時を作ります。GetMillis メソッドと同様に、これはいかにもプログラミング的なので、あまり使わないかもしれません。
millis	ミリ秒
Minute(Calendar instant)	特定の日時の分だけ取得します。例えば、instant が 2011/04/17 12:30:15 であれば、30 が取得できます。
instant	特定の日時
Month(Calendar instant)	特定の日時の月だけ取得します。例えば、instant が 2011/04/17 12:30:15 であれば、4 が取得できます。
instant	特定の日時
MonthName(Calendar instant)	特定の日時の月の名前を取得します。例えば、instant が 2011/04/17 12:30:15 であれば、4 月というテキストが取得できます。
instant	特定の日時
Now()	現在日時を取得します。
Second(Calendar instant)	特定の日時の秒だけ取得します。例えば、instant が 2011/04/17 12:30:15 であれば、15 が取得できます。
instant	特定の日時
SystemTime()	現在日時をミリ秒で取得します。
Weekday(Calendar instant)	特定の日時の曜日を数値で取得します。1 が日曜日で、7 が土曜日です。
instant	特定の日時
WeekdayName(Calendar instant)	特定の日時の曜日をテキストで取得します。例えば、instant が 2011/04/17 であれば、日曜日というテキストが取得できます。

instant	特定の日時
Year(Calendar instant)	特定の日時の年だけ取得します。例えば、instant が 2011/04/17 12:30:15 であれば、2011 が取得できます。
instant	特定の日時

***15**　現在の 5 日前の日時をラベルに表示する例



▲ブロックの例

***16**　Clock コンポーネントのメソッドで取得できる値は少し特殊なので、そのまま Label コンポーネントで画面に表示することができません。なので、FormatDate メソッドなどで表示したい形式（年月日表記や時間表記）に変換して、Label コンポーネントで表示します。

***17**　指定できるテキストの形式は以下のパターンになります。

パターン	例	作られる日時
MM/DD/YYYY hh:mm:ss	04/17/2011 12:30:15	2011/04/17 12:30:15
MM/DD/YYYY	04/17/2011	2011/04/17 0:00:00
hh:mm	12:30	1970/01/01 12:30:00

***18**　年月日の順番が海外の順番になっているので注意してください。月/日/年というテキストをセットすると正しく日時が作れます。

Image

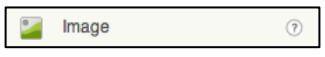


Image コンポーネントは、画像を表示するコンポーネントです。イベントやメソッドはないので純粋に画像を表示するために使うシンプルなコンポーネントです。

プロパティ

Picture	表示する画像をセットします。
Visible	チェックされている場合、画像が表示されます。
Width	画像の幅です。

Height	画像の高さです。

Label 

Label コンポーネントは、テキストを表示するコンポーネントです。イベントやメソッドはないので純粋にテキストを表示するために使うシンプルなコンポーネントです。

プロパティ

BackgroundColor	ラベルの背景色です。
FontBold	チェックされている場合、ラベルのテキストが太字になります。
FontItalic	チェックされている場合、ラベルのテキストが斜体になります。
FontSize	ラベルのテキストの大きさです。
FontTypeface	ラベルのテキストのフォントです。
Text	ラベルのテキストです。
TextAlignment	ラベルのテキストの表示位置です。
TextColor	ラベルのテキストの色です。
Visible	チェックされている場合、ラベルが表示されます。
Width	ラベルの幅です。
Height	ラベルの高さです。

ListPicker 

ListPicker コンポーネントは、ユーザがリストから何かを選択する時に使うコンポーネントです。

ユーザがボタンをクリックするとリストを選択する画面が表示され、リストの中から 1 つをタップすると元の画面に戻ります。この時、AfterPicking イベントでユーザが選択したものを取得することができるので、選ばれたものによって何か動作をするなどができます。

プロパティ

BackgroundColor	ボタンの背景色です。
ElementsFromString	リストの要素を半角カンマ区切りで入力します（*19）。
FontBold	チェックされている場合、ボタンのテキストが太字になります。
FontItalic	チェックされている場合、ボタンのテキストが斜体になります。
FontSize	ボタンのテキストの大きさです。

FontTypeface	ボタンのテキストのフォントです。
Image	ボタンに表示する画像です。
Selection	選択されたリストの要素です（*20）。例えば、バナナが選択された場合は、そのまま「バナナ」とセットされています。
Text	ボタンのテキストです。
TextAlignment	ボタンのテキストの表示位置です。
TextColor	ボタンのテキストの色です。
Visible	チェックされている場合、ボタンが表示されます。
Width	ボタンの幅です。
Height	ボタンの高さです。

*19　例えば、バナナ,からあげ,おでん缶と入力すると、以下のように表示されます。

▲画面の表示

*20　あらかじめセットすることもできますが、Blocks Editor で選択されたリストの要素を見ることの方が多いです。

イベント

AfterPicking()	ユーザがリストの要素を選択した後に呼び出されます。
BeforePicking()	ユーザがリストの要素を選択する前に呼び出されます。
GotFocus()	ボタンがフォーカスされた時に呼び出されます。
LostFocus()	ボタンがフォーカスされなくなった時に呼び出されます。

メソッド

Open()	リストを選択する画面を表示します。

PasswordTextBox 

PasswordTextBox コンポーネントは、パスワードを入力するためのコンポーネントです。ユーザが入力した文字が・で表示されます。

プロパティ

BackgroundColor	パスワードテキストボックスの背景色です。
Enabled	チェックされている場合、ユーザが入力できます。
FontBold	チェックされている場合、テキストが太字になります。
FontItalic	チェックされている場合、テキストが斜体になります。
FontSize	テキストの大きさです。
FontTypeface	テキストのフォントです（*21）。
Hint	何も入力されていない時に表示されるヒント用のテキストです（*22）。
TextAlignment	テキストの表示位置です。
TextColor	テキストの色です。
Visible	チェックされている場合、パスワードテキストボックスが表示されます。
Width	パスワードテキストボックスの幅です。
Height	パスワードテキストボックスの高さです。

*21　PasswordTextBox コンポーネントは、入力した文字が「・」で表示されるのでフォントの設定はあまり意味がありません。

*22　例えば、「パスワードを入力してください」と設定しておくと以下のように表示されます。

▲画面の表示

イベント

GotFocus()	パスワードテキストボックスがフォーカスされた時に呼び出されます。
LostFocus()	パスワードテキストボックスがフォーカスされなくなった時に呼び出されます。

◆◆TextBox 

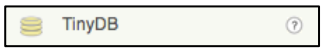
TextBox コンポーネントは、テキストを入力するためのコンポーネントです。

プロパティ

BackgroundColor	テキストボックスの背景色です。
Enabled	チェックされている場合、ユーザが入力できます。
FontBold	チェックされている場合、テキストが太字になります。
FontItalic	チェックされている場合、テキストが斜体になります。
FontSize	テキストの大きさです。
FontTypeface	テキストのフォントです。
Hint	何も入力されていない時に表示されるヒント用のテキストです。
NumbersOnly	チェックされている場合、数字のみ入力が可能になります。（数字以外が入力できなくなります。）
Text	テキストです。あらかじめ入力しておくこともできます。
TextAlignment	テキストの表示位置です。
TextColor	テキストの色です。
Visible	チェックされている場合、テキストボックスが表示されます。
Width	テキストボックスの幅です。
Height	テキストボックスの高さです。

イベント

GotFocus()	テキストボックスがフォーカスされた時に呼び出されます。
LostFocus()	テキストボックスがフォーカスされなくなった時に呼び出されます。

TinyDB 

TinyDB コンポーネントは、データを保存するためのコンポーネントです。画面には表示されないコンポーネントです。

Android アプリを終了すると、それまで入力したデータや選択した状態などは自動的には保存されません。アプリを起動した時に、それまでに入力したデータや選択した状態などのデータを使いたい場合は、この TinyDB コンポーネントを使ってデータを保存したり、取り出したりします。

プロパティ

ありません。

メソッド

StoreValue(text tag, valueToStore)	指定したタグにデータを保存します。
tag	保存するデータのタグ
valueToStore	保存するデータ
GetValue(text tag)	指定したタグのデータを取り出します。指定したタグでデータが保存されていなかった場合は、空のテキストが取得できます。
tag	取り出したいデータのタグ

Screen

Screen コンポーネントは、画面全体の設定を行うコンポーネントです。他のコンポーネントのように任意で置くものではなく、必ず 1 つ初めから置いてあり、追加、削除したり、名前を変更したりできない特殊なコンポーネントです。

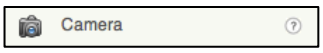
プロパティ

BackgroundColor	画面全体の背景色です。
BackgroundImage	画面全体の背景画像です。
Icon	アプリのアイコン画像です。端末にインストールした時に、アプリアイコンとして表示されます。
ScreenOrientation	画面の向きです（*23）。
Scrollable	チェックされている場合、画面の高さがディスプレイの高さを超える時にスクロールバーが表示され、スクロールできるようになります。
Title	タイトルバーに表示されるテキストです。

***23** ScreenOrientation プロパティの値と意味は以下の通りです。

Unspecified	画面の向きを指定しません。端末のディスプレイの向きと同じ向きで表示されます。
Portrait	画面を縦向きに固定で表示します。端末を横向きにしても画面が横向きになりません。
Landscape	画面を横向きに固定で表示します。端末を縦向きにしても画面が縦向きになりません。

メディアコンポーネント

Camera 

Camera コンポーネントは、カメラを起動するコンポーネントです。画面には表示されないコンポーネントです。

撮影した写真を画面に表示したりできます。

プロパティ

ありません。

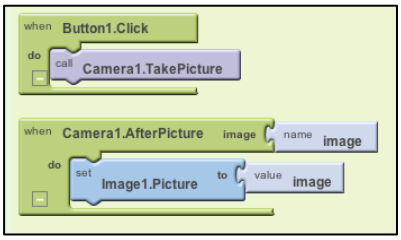
イベント

AfterPicture(Text image)	写真が撮影された後に呼び出されます。image に撮影された写真のパスが入っているので Image コンポーネントの Picture プロパティなどにセットすると写真が表示されます。
image	撮影された写真のパスです。

メソッド

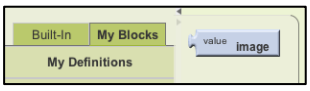
TakePicture()	カメラを起動します（*24）（*25）。

*24　ボタンがクリックされたらカメラを起動し、撮影された写真を Image コンポーネントに表示する例は以下のとおりです。



▲ブロックの例

*25　イベントの値を見るブロックは以下のように My Blocks の My Definitions にあります。他のコンポーネントも同様です。



▲イベントの値を見るブロック

ImagePicker 

ImagePicker コンポーネントは、端末に保存されている写真や画像を選ぶコンポーネントです。ユーザがボタンをクリックすると、写真を選ぶ画面が表示されます。

選ばれた写真を画面に表示したりできます。

プロパティ

BackgroundColor	ボタンの背景色です。
Enabled	チェックされている場合、ボタンをクリックできます。
FontBold	チェックされている場合、ボタンのテキストが太字になります。
FontItalic	チェックされている場合、ボタンのテキストが斜体になります。
FontSize	ボタンのテキストの大きさです。
FontTypeface	ボタンのテキストのフォントです。
Image	ボタンに表示する画像です。
Text	ボタンのテキストです。
TextAlignment	ボタンのテキストの表示位置です。
TextColor	ボタンのテキストの色です。
Visible	チェックされている場合、ボタンが表示されます。
Width	ボタンの幅です。
Height	ボタンの高さです。

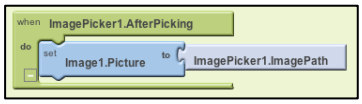
イベント

AfterPicking()	ユーザが写真を選んだ後に呼び出されます。
BeforePicking()	ユーザが写真を選ぶ前に呼び出されます。
GotFocus()	ボタンがフォーカスされた時に呼び出されます。
LostFocus()	ボタンがフォーカスされなくなった時に呼び出されます。

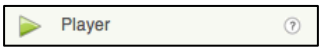
メソッド

Open()	写真を選ぶ画面を表示します（*26）。

***26**　選ばれた写真を Image コンポーネントに表示する例は以下のとおりです。選ばれた写真のパスは、ImagePath プロパティに入っています。



▲ブロックの例

Player 

Player コンポーネントは、音声ファイルを再生したり、端末のバイブレータを振動させたりできるコンポーネントです。画面には表示されないコンポーネントです。

再生時間が長い音声ファイルは、この Player コンポーネントを使い、効果音のような再生時間が短い音声ファイルを再生するには、次の Sound コンポーネントを使ってください。

Player コンポーネントで再生できる音声ファイルのファイル形式は次のとおりです。

3GPP(.3gp)、MPEG-4(.mp4 または.m4a)、MP3(.mp3)、Ogg(.ogg)、WAVE(.wav)、MIDI(.mid など)

詳しくは以下の URL を参照してください。

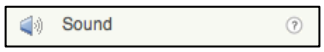
http://developer.android.com/intl/ja/guide/appendix/media-formats.html

プロパティ

Source	再生したい音声ファイルをセットします。

メソッド

Pause()	音声ファイルの再生を一時停止します。
Start()	音声ファイルの再生を開始します。一時停止した後であれば、一時停止したところから再生します。
Stop()	音声ファイルの再生を停止します。
Vibrate(number milliseconds)	端末のバイブレーターを振動させます。
milliseconds	振動させる時間（ミリ秒）。例えば、1 秒間振動させるなら 1000 をセットします。

Sound 

Sound コンポーネントは、Player コンポーネントと同じように音声ファイルを再生したり、端末のバイブレータを振動させたりできるコンポーネントです。画面には表示されないコンポーネントです。

効果音のような再生時間が短い音声ファイルを再生するには、この Sound コンポーネントを使ってください。

再生できる音声ファイルのファイル形式も Player コンポーネントと同じです。

Player コンポーネントとの違い 2 つあります。

1 つ目の違いは、Pause メソッドで一時停止したあとに、Play メソッドを実行すると音声ファイルの先頭から再生され、Resume メソッドを実行すると、一時停止したところ

から再生が再開される点です。

2 つ目の違いは、例えば、画面をタップしたら効果音を鳴らすような使い方の場合に、タップを連続された時に連続して再生されないようにインターバルを設定できる点です。1 秒のインターバルを設定した場合は、どんなに連続してタップしても 1 秒間隔でしか効果音が鳴らないようになります。

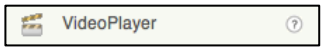
プロパティ

MinimumInterval	音声ファイルを再生するインターバルをミリ秒で設定します。この間隔内で Play メソッドを実行しても音声ファイルが再生されません。
Source	再生したい音声ファイルをセットします。

メソッド

Pause()	音声ファイルの再生を一時停止します。
Play()	音声ファイルの再生を開始します。必ず音声ファイルの先頭から再生します。
Resume()	音声ファイルの再生を再開します。
Stop()	音声ファイルの再生を停止します。
Vibrate(number milliseconds)	端末のバイブレーターを振動させます。
milliseconds	振動させる時間（ミリ秒）。例えば、1 秒間振動させるなら 1000 をセットします。

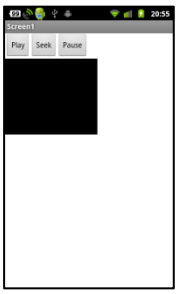
VideoPlayer



VideoPlayer コンポーネントは、動画を再生するコンポーネントです。

以下のように黒い枠で表示されている部分に動画が再生されます。黒い枠をタップすると以下のようなコントローラが表示されるので、このコントローラで動画の再生を操作することも可能です。

ちなみに、なぜか動画を再生する黒い枠の大きさは変更できません（ちょっと小さいですよね……）。



▲再生画面

▲コントローラ

プロパティ

Source	再生したい動画ファイルをセットします（*27）。
Visible	チェックされている場合、画面に表示されます。

*27　あまり大きな動画ファイルをアップロードしようとするとエラーになり、アップロードできません。執筆時点では、3MB ぐらいのファイルサイズが限界のようです。

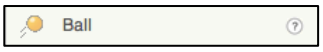
イベント

Completed()	動画の再生が終わったら呼び出されます。

メソッド

GetDuration()	動画ファイルの再生時間をミリ秒で取得します。
Pause()	動画ファイルの再生を一時停止します。
Seekto(number ms)	動画ファイルの再生位置をシーク（移動）します。
ms	シーク（移動）したい再生時間をミリ秒でセットします。
Start()	動画ファイルの再生を開始します。

アニメーションコンポーネント

Ball 

Ball コンポーネントは、Canvas コンポーネントに置いて使う特殊なコンポーネントです。以下のような特徴があります。

・見た目はボールのように円で、色や大きさを変更できます。
・タッチしたりドラッグできます。
・指定した向きに向かって自動的に移動します。
・ブロックから手動で移動することもできます。
・他のボールやイメージスプライトとぶつかった時のイベントがあります。

プロパティ

Enabled	チェックされている場合、ボールが移動したりタッチできる状態になります。逆に、チェックされていない（false をセットした）場合は、ボールが移動しなくなり、タッチやドラッグできなくなります。
Heading	ボールの移動方向です。0 が右、90 が上、180 が左、270 が下です。
Interval	ボールが移動する間隔です。ミリ秒で設定します。例えば、1 秒ごとに移動させたい場合は 1000 にします。
PaintColor	ボールの色です。
Radius	ボールの半径です。
Speed	ボールが 1 回で移動するピクセル数です。0 の場合は、自動的に移動しません。
Visible	チェックされている場合、ボールが表示されます。
X	ボールの X 座標です。
Y	ボールの Y 座標です。

イベント

CollidedWith(component other)	他のボールまたはイメージスプライトとぶつかった時に呼び出されます。
other	ぶつかった相手（ボールまたはイメージスプライト）
Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY)	ボールがドラッグされている時に呼び出されます。
startX	ドラッグを開始した X 座標です。
startY	ドラッグを開始した Y 座標です。
prevX	前回の Dragged イベントの X 座標です。
prevY	前回の Dragged イベントの Y 座標です。
currentX	ドラッグ中の現在の X 座標です。
currentY	ドラッグ中の現在の Y 座標です。
EdgeReached(number edge)	ボールがキャンバスの枠に触れた時に呼び出されます。
edge	触れた向きによって値が入ります（*28）。
NoLongerCollidingWith(component other)	Google のリファレンス（解説）には、「Called when two animated components have stopped colliding.」と、あるのです

	が、いろいろ試しても呼び出されない謎のイベントでした。
Touched(number x, number y)	ボールがタッチされた時に呼び出されます。
x	タッチされたX座標です。
y	タッチされたY座標です。

***28** 以下の値が入ります。

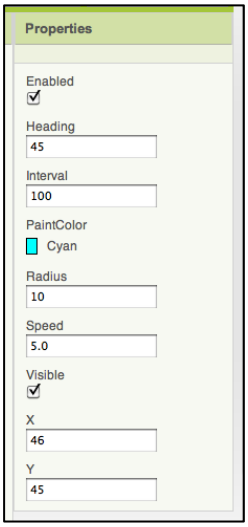
触れた向き	値
上	1
左	-3
下	-1
右	3

メソッド

Bounce(number edge)	ボールを跳ね返らせます。
edge	ぶつかった向き。向きの値の意味は EdgeReached イベントと同じです（***29**）。
CollidingWith(component other)	他のボールかイメージスプライトとぶつかっているか調べます。ぶつかっている場合は true、そうでない場合は false が取得できます。
other	ぶつかっているか調べたい相手（ボールかイメージスプライト）
MoveIntoBounds()	Google のリファレンス（解説）には、「If the ball is out of bounds, this method moves it back in bounds.」と、あるのですが、ボールがキャンバスを超えることがないので、あまり使いどころのないメソッドです。
MoveTo(number x,number y)	ボールを移動します。
x	移動先のX座標
y	移動先のY座標
PointTowards(target)	ボールを target に向かって移動するように移動方向を変化させます。
target	対象になるボールかイメージスプライト

***29** ボールがキャンバスの枠にぶつかったら跳ね返る例は以下のとおりです。画面を見るとボールの移動方向である Heading プロパティの値が自動的に計算されているのが分かります。ボール（Ball コンポ

ーネント）のプロパティを以下のように設定しました。



▲プロパティ

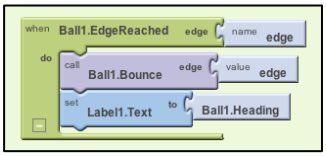
・Heading プロパティを 45 に設定：右上に移動する

・Interval プロパティを 100 に設定：0.1 秒ごとに移動する

・Radius プロパティを 10 に設定：半径を 10px にして見やすいように大きく

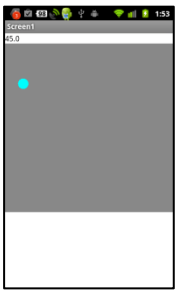
・Speed プロパティを 5.0 に設定：1 回あたり 5px 移動する

キャンバス（Canvas コンポーネント）とラベル（Label コンポーネント）は適当に置いてください。ブロックは以下のとおりです。

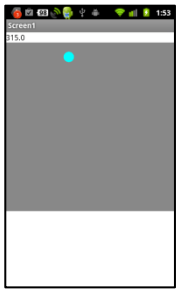


▲ブロックの例

ボールがキャンバスの枠に触れた時に呼び出される EdgeReached イベントで、Bounce メソッドに EdgeReached イベントの edge ブロックを、そのままセットしています。Heading プロパティの値が見えるように Label コンポーネントの Text プロパティにセットしています。以下のように画面に表示されます。



◂表示例①



◂表示例②

ImageSprite 

ImageSprite コンポーネントは、ボールと同等のプロパティ・イベント・メソッドを持つイメージスプライトです。簡単に言うとボールが画像がセットできるように進化したと思ってください。

プロパティ

Enabled	チェックされている場合、イメージスプライトが移動したりタッチできる状態になります。逆に、チェックされていない（false をセットした）場合は、イメージスプライトが移動しなくなり、タッチやドラッグできなくなります。
Heading	イメージスプライトの移動方向です。0 が右、90 が上、180 が左、270 が下です。
Interval	イメージスプライトが移動する間隔です。ミリ秒で設定します。例えば、1 秒ごとに移動させたい場合は 1000 にします。
Picture	イメージスプライトに表示したい画像をセットします。
Speed	イメージスプライトが 1 回で移動するピクセル数です。0 の場合は、自動的に移動しません。
Visible	チェックされている場合、イメージスプライトが表示されます。
X	イメージスプライトの X 座標です。
Y	イメージスプライトの Y 座標です。
Width	イメージスプライトの幅です。
Height	イメージスプライトの高さです。

イベント

CollidedWith(component other)	他のボールまたはイメージスプライトとぶつかった時に呼び出されます。
other	ぶつかった相手（ボールまたはイメージスプライト）
Dragged(number startX, number startY, number prevX, number prevY, number currentX, number currentY)	イメージスプライトがドラッグされている時に呼び出されます。
startX	ドラッグを開始した X 座標です。
startY	ドラッグを開始した Y 座標です。
prevX	前回の Dragged イベントの X 座標です。
prevY	前回の Dragged イベントの Y 座標です。

currentX	ドラッグ中の現在のX座標です。
currentY	ドラッグ中の現在のY座標です。
EdgeReached(number edge)	イメージスプライトがキャンバスの枠に触れた時に呼び出されます。
edge	触れた向きによって値が入ります（*30）。
NoLongerCollidingWith(component other)	Google のリファレンス（解説）には、「Called when two animated components have stopped colliding.」と、あるのですが、いろいろ試しても呼び出されない謎のイベントでした。
Touched(number x, number y)	イメージスプライトがタッチされた時に呼び出されます。
x	タッチされたX座標です。
y	タッチされたY座標です。

*30　以下の値が入ります。

触れた向き	値
上	1
左	-3
下	-1
右	3

メソッド

Bounce(number edge)	イメージスプライトを跳ね返らせます。
edge	ぶつかった向き。向きの値の意味は EdgeReached イベントと同じです。
CollidingWith(component other)	他のボールかイメージスプライトとぶつかっているか調べます。ぶつかっている場合は true、そうでない場合は false が取得できます。
other	ぶつかっているか調べたい相手（ボールかイメージスプライト）
MoveIntoBounds()	Google のリファレンス（解説）には、「If the ball is out of bounds, this method moves it back in bounds.」と、あるのですが、イメージスプライトがキャンバスを超えることがないので、あまり使いど

	ころのないメソッドです。
MoveTo(number x,number y)	イメージスプライトを移動します。
x	移動先の X 座標
y	移動先の Y 座標
PointTowards(target)	イメージスプライトを target に向かって移動するように移動方向を変化させます。
target	対象になるボールかイメージスプライト

ソーシャルコンポーネント

ContactPicker 

ContactPicker コンポーネントは、端末の電話帳から連絡先を選ぶコンポーネントです。ユーザがボタンをクリックすると、連絡先を選ぶ画面が表示されます。

選ばれた連絡先の名前やメールアドレス、写真を取得することができます。ただし、そもそも連絡先の情報が設定されていない場合は、値が取得できないことがあります。(例えば、写真が設定されていない場合などがあります。)

プロパティ

BackgroundColor	ボタンの背景色です。
Enabled	チェックされている場合、ボタンをクリックできます。
FontBold	チェックされている場合、ボタンのテキストが太字になります。
FontItalic	チェックされている場合、ボタンのテキストが斜体になります。
FontSize	ボタンのテキストの大きさです。
FontTypeface	ボタンのテキストのフォントです。
Image	ボタンに表示する画像です。
Text	ボタンのテキストです。
TextAlignment	ボタンのテキストの表示位置です。
TextColor	ボタンのテキストの色です。
Visible	チェックされている場合、ボタンが表示されます。
Width	ボタンの幅です。
Height	ボタンの高さです。
ContactName	連絡先の名前です。

EmailAddress	連絡先のメールアドレスです。
Picture	連絡先の写真です（*31）。

*31　選んだ連絡先の情報は以下のプロパティにセットされています。

▲連絡先情報

イベント

AfterPicking()	ユーザが連絡先を選んだ後に呼び出されます。
BeforePicking()	ユーザが連絡先を選ぶ前に呼び出されます。
GotFocus()	ボタンがフォーカスされた時に呼び出されます。
LostFocus()	ボタンがフォーカスされなくなった時に呼び出されます。

メソッド

Open()	連絡先を選ぶ画面を表示します。

EmailPicker

EmailPicker コンポーネントは、TextBox コンポーネントと同様にテキストを入力すると端末の電話帳のメールアドレスを自動的に検索してくれるコンポーネントです。

ただし、このコンポーネントは執筆時点では正しく動作しませんでした。メールアドレスを途中まで入力すると、自動的に検索して、電話帳の名前がテキストボックスの下に一瞬、表示されるのですが、すぐに消えてしまいます。将来のバージョンアップで修正されることを期待して待ちましょう。

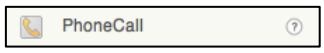
プロパティ

BackgroundColor	テキストボックスの背景色です。
Enabled	チェックされている場合、ユーザが入力できます。
FontBold	チェックされている場合、テキストが太字になります。
FontItalic	チェックされている場合、テキストが斜体になります。
FontSize	テキストの大きさです。
FontTypeface	テキストのフォントです。

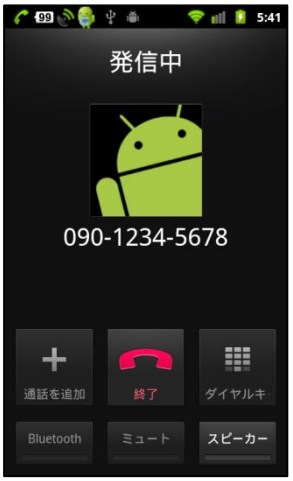
Hint	何も入力されていない時に表示されるヒント用のテキストです。
Text	テキストです。あらかじめ入力しておくこともできます。
TextAlignment	テキストの表示位置です。
TextColor	テキストの色です。
Visible	チェックされている場合、テキストボックスが表示されます。
Width	テキストボックスの幅です。
Height	テキストボックスの高さです。

イベント

GotFocus()	テキストボックスがフォーカスされた時に呼び出されます。
LostFocus()	テキストボックスがフォーカスされなくなった時に呼び出されます。

PhoneCall 

PhoneCall コンポーネントは、電話をかけるコンポーネントです。とてもシンプルなコンポーネントで、PhoneNumber プロパティにセットされている電話番号に、MakePhoneCall メソッドで電話をかけるだけです。電話をかける時は以下のような画面が表示されるので、この画面で発信を停止するなどします。



▲電話をかける

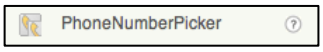
プロパティ

PhoneNumber	電話をかける電話番号です。

メソッド

MakePhoneCall()	電話をかけます。

PhoneNumberPicker



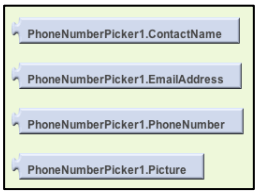
PhoneNumberPicker コンポーネントは、端末の電話帳から電話番号を選ぶコンポーネントです。ユーザがボタンをクリックすると、連絡先を選ぶ画面が表示されます。ContactPicker コンポーネントと似ていますが、こちらのコンポーネントの場合は、電話番号が設定されている連絡先のみ画面に表示されます。

選ばれた連絡先の電話番号はもちろん、名前やメールアドレス、写真を取得することができます。ただし、ContactPicker コンポーネントと同様に、そもそも連絡先の情報が設定されていない場合は、値が取得できないことがあります。（例えば、写真が設定されていない場合などがあります。）

プロパティ

BackgroundColor	ボタンの背景色です。
Enabled	チェックされている場合、ボタンをクリックできます。
FontBold	チェックされている場合、ボタンのテキストが太字になります。
FontItalic	チェックされている場合、ボタンのテキストが斜体になります。
FontSize	ボタンのテキストの大きさです。
FontTypeface	ボタンのテキストのフォントです。
Image	ボタンに表示する画像です。
Text	ボタンのテキストです。
TextAlignment	ボタンのテキストの表示位置です。
TextColor	ボタンのテキストの色です。
Visible	チェックされている場合、ボタンが表示されます。
Width	ボタンの幅です。
Height	ボタンの高さです。
ContactName	連絡先の名前です。
EmailAddress	連絡先のメールアドレスです。
PhoneNumber	連絡先の電話番号です。
Picture	連絡先の写真です（*32）。

***32**　選んだ連絡先の情報は以下のプロパティにセットされています。



▲連絡先情報

イベント

AfterPicking()	ユーザが連絡先を選んだ後に呼び出されます。
BeforePicking()	ユーザが連絡先を選ぶ前に呼び出されます。
GotFocus()	ボタンがフォーカスされた時に呼び出されます。
LostFocus()	ボタンがフォーカスされなくなった時に呼び出されます。

メソッド

Open()	連絡先を選ぶ画面を表示します。

Texting 

Texting コンポーネントは、SMS を送信するコンポーネントです。

このコンポーネントを使う時には SMS の料金に気を付けてください。例えば、docomo の場合は、送信　1 回あたり 5.25 円の料金がかかります。また、ユーザーに確認なく SMS が送信されます。私もテスト中に何通か間違えて送ってしまいました。

プロパティ

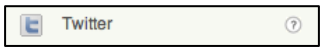
Message	SMS の本文のテキストです。
PhoneNumber	SMS の送信先の電話番号です。
ReceivingEnabled	チェックされている場合、SMS を受信することができます。

イベント

MessageReceived(text number, text messageText)	SMS を受信した時に呼び出されます。
number	受信した電話番号
messageText	受信した SMS の本文

メソッド

SendMessage()	PhoneNumber プロパティの電話番号に、Message プロパティのテキストを本文にして、SMS を送信します。

Twitter 

Twitter コンポーネントは、その名のとおり Twitter にアクセスするためのコンポーネントです。画面には表示されないコンポーネントです。

Twitter につぶやいたり、タイムラインを取得したりできます。

Twitter はログインして使う Web サービスですが、このコンポーネントもログインして使う機能が多くあります。このログインのことを認証と呼びます。

プロパティ

ConsumerKey	コンシューマーキーです。
ConsumerSecret	コンシューマーシークレットです。
DirectMessages	ダイレクトメッセージのリストです。RequestDirectMessages メソッドを呼び出して、DirectMessagesReceived イベントが呼び出された後に、値が入っています。
Followers	フォロワー（フォローされている人）のリストです。RequestFollowers メソッドを呼び出して、FollowersReceived イベントが呼び出された後に、値が入っています。
FriendTimeline	タイムラインのリストです。RequestFriendTimeline メソッドを呼び出して、FriendTimelineReceived イベントが呼び出された後に、値が入っています。
Mentions	メンション（@関連）のリストです。RequestMentions メソッドを呼び出して、MentionsReceived イベントが呼び出された後に、値が入っています。
SearchResults	Twitter を検索した結果のリストです。SearchTwitter メソッドを呼び出して、SearchSuccessful イベントが呼び出された後に、値が入っています。
Username	認証したユーザのユーザ名です。認証していない時は、空の値が入っています。

イベント

DirectMessagesReceived(list messages)	RequestDirectMessages メソッドを呼び出して、ダイレクトメッセージが取得できた時に呼び出されます。
messages	取得できたダイレクトメッセージのリスト
FollowersReceived(list followers)	RequestFollowers メソッドを呼び出して、フォロワー（フォローされている人）が取得できた時に呼び出されます。
followers	取得できたフォロワー（フォローされている人）のリスト
FriendTimelineReceived(list user-messages-list)	RequestFriendTimeline メソッドを呼び出して、タイムラインが取得できた時に呼び出されます。

user-messages-list	取得できたタイムラインのリスト。このリストの要素は、さらにリストになっていて、一つ目の要素にユーザ名、二つ目の要素につぶやきが入っています。
IsAuthorized()	Authorize メソッドを呼び出して、認証が完了した時に呼び出されます。さらに、CheckAuthorized メソッドを呼び出して、認証済みだった時にも呼び出されます。
MentionsReceived(list mentions)	RequestMentions メソッドを呼び出して、メンション（@関連）が取得できた時に呼び出されます。
mentions	取得できたメンション（@関連）のリスト
SearchSuccessful(list searchResults)	SearchTwitter メソッドを呼び出して、検索結果が取得できた時に呼び出されます。
searchResults	取得できた検索結果のつぶやきのリスト

メソッド

Authorize()	認証するための Twitter のページを表示します。認証が完了すると、IsAuthorized イベントが呼び出されます。
CheckAuthorized()	認証済みかどうかチェックします。認証済みだった場合は、IsAuthorized イベントが呼び出されます。
DeAuthorize()	ログアウトします。
DirectMessage(text user, text message)	ユーザにダイレクトメッセージを送信します。
user	送信先のユーザ
message	ダイレクトメッセージの本文
Follow(text user)	ユーザをフォローします。
user	フォローしたいユーザ
RequestDirectMessages()	ダイレクトメッセージを取得します。
RequestFollowers()	フォロワー（フォローされている人）を取得します。
RequestFriendTimeline()	タイムラインを取得します。
RequestMentions()	メンション（@関連）を取得します。

SearchTwitter(text query)	Twitter を検索します。
query	検索したいテキスト（検索ワード）
SetStatus(text s)	認証しているユーザでつぶやきを投稿します。
s	つぶやきの本文
StopFollowing(text user)	フォローを解除します。
user	フォローを解除したいユーザ

センサーコンポーネント

AccelerometerSensor

AccelerometerSensor コンポーネントは、端末の加速度センサーを扱うためのコンポーネントです。画面には表示されないコンポーネントです。

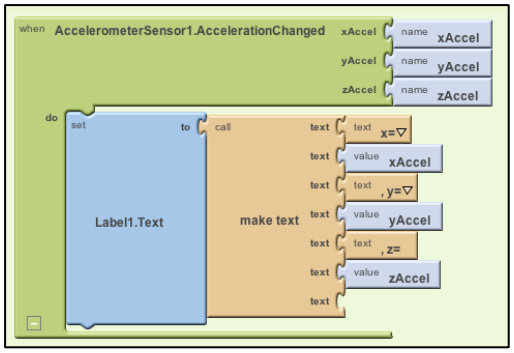
このコンポーネントを使うと、端末の加速度を取得することができます。

プロパティ

Enabled	チェックされている場合、加速度センサーを ON にします。
Available	端末が加速度センサーを搭載している場合は true、搭載していない場合は false です（*33）。
XAccel	X 軸の加速度です。端末を水平に置いた状態で、右に傾けると-1,-2,-3 とマイナスに増えていきます。左に傾けると 1,2,3 とプラスに増えていきます。
YAccel	Y 軸の加速度です。端末を水平に置いた状態で、手前に起こしていくと 1,2,3 とプラスに増えていきます。底の方を持ち上げていくと-1,-2,-3 とマイナスに増えていきます。
ZAccel	Z 軸の加速度です。端末を水平に置いた状態で、約 9.8、裏返すと約-9.8 になります（*34）。

***33**　ほとんどの端末が加速度センサーを搭載しているので、あまり気にしないでください。

***34**　以下のようなブロックで、それぞれの値をラベルに表示すると実際の値が見られてイメージが湧きやすいと思います。



▲ブロックの例

イベント

AccelerationChanged(number xAccel, number yAccel, number zAccel)	加速度が変化した時に呼び出されます。かなりの頻度で呼び出されるので注意してください。
xAccel	X 軸の加速度です。
yAccel	Y 軸の加速度です。
zAccel	Z 軸の加速度です。
Shaking()	端末がシェイクされた時に呼び出されます（*35）。

*35　このイベントはちょっと面白くて、端末をシャカシャカとシェイクすると呼び出されます。

LocationSensor

LocationSensor コンポーネントは、端末の位置情報を取得するコンポーネントです。画面には表示されないコンポーネントです。

ただし、このコンポーネントは、アプリを端末にダウンロードしないと動作しません。さらに、Android エミュレータでも動作しないので Android 携帯が必要です。アプリを端末にダウンロードする方法は、のちほど紹介します。

プロパティ

Enabled	チェックされている場合、位置情報センサーを ON にします。
ProviderLocked	チェックされている場合、サービスプロバイダを自動的に変更しないようにロックします。
ProviderName	サービスプロバイダ（*36）の名前です。
Accuracy	現在の位置情報の精度（メートル）です。
Altitude	現在の位置情報の高度です。高度が取得できない場合もあります。
AvailableProviders	利用できるサービスプロバイダのリストが入っています。

CurrentAddress	現在の位置情報の住所です（*37）。（例）日本 愛知県名古屋市西区菊井１丁目２４－１８
HasAccuracy	精度を取得できる場合は true、そうでない場合は false が入っています。
HasAltitude	高度を取得できる場合は true、そうでない場合は false が入っています。
HasLongitudeLatitude	緯度、経度を取得できる場合は true、そうでない場合は false が入っています。
Latitude	現在の位置情報の緯度です。
Longitude	現在の位置情報の経度です。

*36　サービスプロバイダというのは、位置情報を取得する方法のことです。以下の種類があります。

値	意味
gps	GPS から位置情報を取得します。
network	ネットワーク（主に WiFi）から位置情報を取得します。

サービスプロバイダは上級者向けの設定なので、よく分からない場合は何も設定しないでください。何も設定しない場合は、自動的に設定されるので特に問題なく位置情報が取得できます（そもそも地下やビルの近くなどで位置情報が取得できないことがありますが……）

*37　ほとんどの場合、位置情報は誤差が数メートルあるので、だいたい間違った住所が取得できます。

イベント

LocationChanged(number latitude, number longitude, number altitude)	位置情報が取得できるたびに呼び出されます。
latitude	現在の位置情報の緯度です。
longitude	現在の位置情報の経度です。
altitude	現在の位置情報の高度です。
StatusChanged(text provider, text status)	サービスプロバイダのステータスが変化した時に呼び出されます。
provider	変化したサービスプロバイダです。
status	ステータスです。

メソッド

LatitudeFromAddress(text locationName)	建物の名前や住所から緯度を計算します。例えば、東京タワーというテキストを指定

	すると、35.65861 という緯度が取得できます。
locationName	緯度を計算したい建物の名前や住所などのテキスト
LongitudeFromAddress(text locationName)	建物の名前や住所から経度を計算します。
locationName	経度を計算したい建物の名前や住所などのテキスト

OrientationSensor

OrientationSensor コンポーネントは、端末の傾きセンサーを扱うためのコンポーネントです。画面には表示されないコンポーネントです。

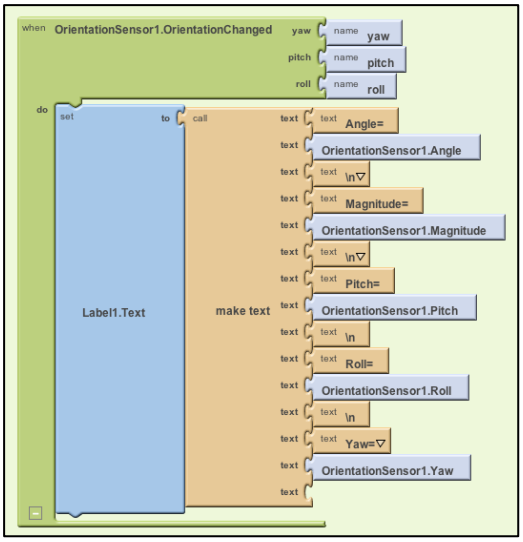
このコンポーネントを使うと、端末の傾きを取得することができます。

プロパティ

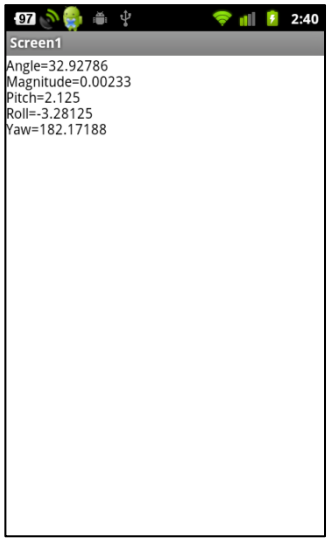
Enabled	チェックされている場合、傾きセンサーを ON にします。
Angle	水平状態を基準にして傾いている方向を角度で取得できます。右に傾いている場合は約 0（または、約 360）、上に傾いている場合は約 90、左に傾いている場合は約 180、下に傾いている場合は約 270。ボール転がしゲームをイメージすると分かりやすいです。
Available	端末が傾きセンサーを搭載している場合は true、搭載していない場合は false です（*38）。
Magnitude	傾き具合が 0 から 1 の間の数値で取得できます。水平状態で 0、だんだん傾けると値が増えていき、垂直状態で 1 になります。
Pitch	縦方向の傾きを角度で取得できます。端末を水平に置いた状態で、手前に起こしていくと-1,-2,-3 とマイナスに増えていき、垂直の状態で約-90 になります。逆に、奥に倒していくと 1,2,3 とプラスに増えていき、垂直の状態で約 90 になります。
Roll	横方向の傾きを角度で取得できます。端末を水平に置いた状態で、右に傾けると-1,-2,-3 とマイナスに増えていき、垂直の状態で約-90 になります。逆に、左に傾けると 1,2,3 とプラスに増えていき、垂直の状態で約 90 になります。
Yaw	方位を角度で取得できます。北が 0、東が 90、南が 180、西が 270 という値が取得できます（*39）。

*38　ほとんどの端末が傾きセンサーを搭載しているので、あまり気にしないでください。

***39**　以下のようなブロックで、それぞれの値をラベルに表示すると実際の値が見られてイメージが湧きやすいと思います。



▲ブロックの例



▲表示の例

イベント

OrientationChanged(number yaw, number pitch, number roll)	傾きが変化した時に呼び出されます。かなりの頻度で呼び出されるので注意してください。
yaw	方位
pitch	縦方向の傾き
roll	横方向の傾き

画面配置コンポーネント

HorizontalArrangement

HorizontalArrangement コンポーネントは、コンポーネントを横に並べるためのコンポーネントです。このコンポーネントの内側に置いたコンポーネントは左から右に並んで表示されます。

プロパティ

Visible	チェックされている場合、内側のコンポーネントが表示されます。チェックをはずすことで、内側のコンポーネントを含めて非表示にできます。

Width	HorizontalArrangement コンポーネントの幅です。
Height	HorizontalArrangement コンポーネントの高さです。

TableArrangement 

TableArrangement コンポーネントは、コンポーネントを表のように並べるためのコンポーネントです。列と行の数を設定できます

プロパティ

Columns	表の列の数です。
Rows	表の行の数です。
Visible	チェックされている場合、内側のコンポーネントが表示されます。チェックをはずすことで、内側のコンポーネントを含めて非表示にできます。
Width	TableArrangement コンポーネントの幅です。
Height	TableArrangement コンポーネントの高さです。

VerticalArrangement 

VerticalArrangement コンポーネントは、コンポーネントを縦に並べるためのコンポーネントです。このコンポーネントの内側に置いたコンポーネントは上から下に並んで表示されます。

プロパティ

Visible	チェックされている場合、内側のコンポーネントが表示されます。チェックをはずすことで、内側のコンポーネントを含めて非表示にできます。
Width	VerticalArrangement コンポーネントの幅です。
Height	VerticalArrangement コンポーネントの高さです。

レゴマインドストームコンポーネント

NxtColorSensor 

NxtColorSensor コンポーネントは、カラーセンサーを扱うためのコンポーネントです。

プロパティ

AboveRangeEventEnabled	チェックされている場合、明るさの値が TopOfRange プロパティの値を上回った時に AboveRange イベントが呼び出されます。
BelowRangeEventEnabled	チェックされている場合、明るさの値が BottomOfRange プロパティの値を下回った時に BelowRange イベントが呼び出されます。
BluetoothClient	Bluetooth 通信に使用する BluetoothClient コンポーネントをセットします。
BottomOfRange	BelowRange、WithinRange イベントが呼び出される条件になる明るさの範囲の下限値です。
ColorChangedEventEnabled	チェックされている場合、センサーが認識している色が変化した時に ColorChanged イベントが呼び出されます。
DetectColor	チェックされている場合、色を検知します。チェックされていない場合、明るさを検知します。さらに、チェックされている場合、BelowRange、WithinRange、AboveRange イベントが発生せず、GenerateColor でセットしたカラーランプは点灯しません。チェックされていない場合、ColorChanged が発生しません（*40）。
GenerateColor	カラーランプを点灯します。
SensorPort	センサーが接続されているポート番号をセットします。
TopOfRange	WithinRange、AboveRange イベントが呼び出される条件になる明るさの範囲の上限値です。
WithinRangeEventEnabled	チェックされている場合、明るさの値が BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

***40**　つまり、色か明るさのどちらかしか検知できないということですね。

イベント

AboveRange()	明るさが TopOfRange プロパティの値を下回った時に呼び出されます。
ColorChanged(number color)	カラーセンサーが認識している色が変化した時に呼び出されます。
color	カラーセンサーが認識している色
BelowRange()	明るさが BottomOfRange プロパティの値を下回った時に

	呼び出されます。
WithinRange()	明るさが BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

メソッド

GetColor()	カラーセンサーが認識している色を返します。色を検知できない場合は、None を返します。
GetLightLevel()	カラーセンサーが認識している明るさを 0 から 1023 の値で返します。明るさを検知できない場合は、-1 を返します。

NxtDirectCommands

NxtDirectCommands コンポーネントは、様々な操作を行うコンポーネントです。センサーと移動以外の操作は、このコンポーネントで行います。

ロボットの状態の取得、設定や、サウンドの再生などができます。

プロパティ

BluetoothClient	Bluetooth 通信に使用する BluetoothClient コンポーネントをセットします。

メソッド

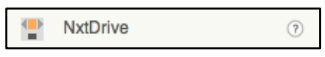
DeleteFile(text fileName)	ロボット上のファイルを削除します。
fileName	削除したいファイル名
DownloadFile(text source, text destination)	ロボットにファイルをダウンロードします。
source	ダウンロードしたい Android 携帯上のファイルパス
destination	ダウンロード先のロボット上のファイル名
GetBatteryLevel()	ロボットのバッテリーの電圧を取得します。単位はミリボルトです。
GetBrickName()	ロボットの Bluetooth で利用される名前を取得します。デフォルトは NXT。
GetCurrentProgramName()	ロボットで現在実行中のプログラム名を取得します。
GetFirmwareVersion()	ファームウェアとプロトコルのバージョンをリストで取得します。リストの 1 つ目の要素がファームウェアのバージョンで、2 つ目の要素がプロトコルのバージョンです。

GetInputValues(text sensorPortLetter)	ロボットの入力ポートの値を取得します。あらかじめ、SetInputMode メソッドで設定する必要があります。
sensorPortLetter	値を取得したい入力ポート
GetOutputState(text motorPortLetter)	ロボットの出力ポートの値を取得します。
motorPortLetter	値を取得したい出力ポート
KeepAlive()	ロボットの電源が切れないように信号を送ります。スリープするまでの時間がミリ秒で取得できます。
ListFiles(text wildcard)	ロボットにあるファイルから指定した文字列を含むファイル名のリストを取得します。
wildcard	検索したい文字列
LsGetStatus(text sensorPortLetter)	読み込み可能なバイト数を取得します。
sensorPortLetter	取得したいポート
LsRead(text sensorPortLetter)	ロボットの入力センサーから符合なし低速データを読み込みます。あらかじめ、SetInputMode メソッドで設定する必要があります。
LsWrite(text sensorPortLetter, list list, number rxDataLength)	ロボットの入力センサーに低速データを書き込みます。あらかじめ、SetInputMode メソッドで設定する必要があります。
MessageRead(number mailbox)	ロボットのメールボックスからメッセージを読み込みます。
mailbox	読み込みたいメールボックスを 1 から 10 の間で指定します。
MessageWrite(number mailbox, text message)	ロボットのメールボックスにメッセージを書き込みます。
mailbox	書き込みたいメールボックスを 1 から 10 の間で指定します。
message	書き込むメッセージ
PlaySoundFile(text fileName)	ロボットにあるサウンドファイルを再生します。
fileName	再生したいサウンドファイルのファイル名
PlayTone(number frequencyHz, number durationMs)	トーン音を再生します。
frequencyHz	周波数

durationMs	再生時間
ResetInputScaledValue(text sensorPortLetter)	ロボットの入力センサーのスケールさせた値をリセットします。
sensorPortLetter	リセットしたい入力ポート
ResetMotorPosition(text motorPortLetter, boolean relative)	モーターの位置をリセットします。
motorPortLetter	リセットしたいモーターのポート
relative	true の場合、相対的にリセットします。false の場合、絶対的にリセットします（*41）。
SetBrickName(text name)	ロボットの Bluetooth で利用される名前をセットします。
name	セットしたい名前
SetInputMode(text sensorPortLetter, number sensorType, number sensorMode)	ロボットの入力センサーを設定します。
sensorPortLetter	設定した入力ポート
sensorType	センサータイプ
sensorMode	センサーモード
SetOutputState(text motorPortLetter, number power, number mode, number regulationMode, number turnRatio, number runState, number tachoLimit)	ロボットのモーターの出力状態をセットします（*42）。
StartProgram(text programName)	プログラムを実行します。
programName	実行したいプログラム名
StopProgram()	実行しているプログラムを停止します。
StopSoundPlayback()	サウンドの再生を停止します。

***41、*42**　ドキュメントを読んだだけでは、意味がわかりませんでした。

NxtDrive



NxtDrive コンポーネントは、ロボットの動きを操作するコンポーネントです。

プロパティ

BluetoothClient	Bluetooth 通信に使用する BluetoothClient コンポーネントをセットします。
DriveMotors	モーターのポートです。
StopBeforeDisconnect	チェックされている場合、Bluetooth 接続が切断される前にモーターを停止します。
WheelDiameter	タイヤの直径です。

メソッド

MoveBackward(number power, number distance)	指定した距離だけバックします。
power	モーターの強さをパーセントで指定します。
distance	バックする距離
MoveBackwardIndefinitely(number power)	他の命令やストップ命令を受け取るまでバックします。
power	モーターの強さをパーセントで指定します。
MoveForward(number power, number distance)	指定した距離だ前進します。
power	モーターの強さをパーセントで指定します。
distance	前進する距離
MoveForwardIndefinitely(number power)	他の命令やストップ命令を受け取るまで前進します。
power	モーターの強さをパーセントで指定します。
Stop()	モーターを停止します。
TurnClockwiseIndefinitely(number power)	右回転します。
power	モーターの強さをパーセントで指定します。
TurnCounterClockwiseIndefinitely(number power)	左回転します。
power	モーターの強さをパーセントで指定します。

NxtLightSensor

NxtLightSensor コンポーネントは、光センサーを扱うためのコンポーネントです。

プロパティ

AboveRangeEventEnabled	チェックされている場合、明るさの値が TopOfRange プロパティの値を上回った時に AboveRange イベントが呼び出されます。
BelowRangeEventEnabled	チェックされている場合、明るさの値が BottomOfRange プロパティの値を下回った時に BelowRange イベントが呼び出されます。
BluetoothClient	Bluetooth 通信に使用する BluetoothClient コンポーネントをセットします。
BottomOfRange	BelowRange、WithinRange イベントが呼び出される条件になる明るさの範囲の下限値です。
GenerateLight	チェックされている場合、ランプを点灯します。
SensorPort	センサーが接続されているポート番号をセットします。
TopOfRange	WithinRange、AboveRange イベントが呼び出される条件になる明るさの範囲の上限値です。
WithinRangeEventEnabled	チェックされている場合、明るさの値が BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

イベント

AboveRange()	明るさが TopOfRange プロパティの値を下回った時に呼び出されます。
BelowRange()	明るさが BottomOfRange プロパティの値を下回った時に呼び出されます。
WithinRange()	明るさが BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

メソッド

GetLightLevel()	カラーセンサーが認識している明るさを 0 から 1023 の値で返します。明るさを検知できない場合は、-1 を返します。

NxtSoundSensor

NxtSoundSensor コンポーネントは、サウンドセンサーを扱うためのコンポーネントです。

プロパティ

AboveRangeEventEnabled	チェックされている場合、音の大きさが TopOfRange プロパティの値を上回った時に AboveRange イベントが呼び出されます。
BelowRangeEventEnabled	チェックされている場合、音の大きさが BottomOfRange プロパティの値を下回った時に BelowRange イベントが呼び出されます。
BluetoothClient	Bluetooth 通信に使用する BluetoothClient コンポーネントをセットします。
BottomOfRange	BelowRange、WithinRange イベントが呼び出される条件になる音の大きさの範囲の下限値です。
SensorPort	センサーが接続されているポート番号をセットします。
TopOfRange	WithinRange、AboveRange イベントが呼び出される条件になる音の大きさの範囲の上限値です。
WithinRangeEventEnabled	チェックされている場合、音の大きさが BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

イベント

AboveRange()	音の大きさが TopOfRange プロパティの値を下回った時に呼び出されます。
BelowRange()	音の大きさが BottomOfRange プロパティの値を下回った時に呼び出されます。
WithinRange()	音の大きさが BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

メソッド

GetSoundLevel()	音の大きさを 0 から 1023 の値で取得します。音の大きさが取得できない場合は、-1 を返します。

NxtTouchSensor

NxtTouchSensor コンポーネントは、タッチセンサーを扱うためのコンポーネントです。

プロパティ

BluetoothClient	Bluetooth 通信に使用する BluetoothClient コンポーネントをセットします。
PressedEventEnabled	チェックされている場合、タッチセンサーが押された時に Pressed イベントが呼び出されます。
ReleasedEventEnabled	チェックされている場合、タッチセンサーが離された時に Pressed イベントが呼び出されます。
SensorPort	センサーが接続されているポート番号をセットします。

イベント

Pressed()	タッチセンサーが押された時に呼び出されます。
Released()	タッチセンサーが離された時に呼び出されます。

メソッド

IsPressed()	タッチセンサーが押されている場合は true、押されていない場合は false を返します。

NxtUltrasonicSensor

NxtUltrasonicSensor コンポーネントは、超音波センサーを扱うためのコンポーネントです。

プロパティ

AboveRangeEventEnabled	チェックされている場合、対象物との距離が TopOfRange プロパティの値を上回った時に AboveRange イベントが呼び出されます。
BelowRangeEventEnabled	チェックされている場合、対象物との距離が BottomOfRange プロパティの値を下回った時に BelowRange イベントが呼び出されます。
BluetoothClient	Bluetooth 通信に使用する BluetoothClient コンポーネントをセットします。
BottomOfRange	BelowRange、WithinRange イベントが呼び出される条件にな

	る対象物との距離の範囲の下限値です。
SensorPort	センサーが接続されているポート番号をセットします。
TopOfRange	WithinRange、AboveRange イベントが呼び出される条件になる対象物との距離の範囲の上限値です。
WithinRangeEventEnabled	チェックされている場合、対象物との距離が BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

イベント

AboveRange()	対象物との距離が TopOfRange プロパティの値を下回った時に呼び出されます。
BelowRange()	対象物との距離が BottomOfRange プロパティの値を下回った時に呼び出されます。
WithinRange()	対象物との距離が BottomOfRange プロパティの値を上回った時か TopOfRange プロパティの値を下回った時に WithinRange イベントが呼び出されます。

メソッド

GetDistance()	対象物との距離を 0 から 254 センチメートルの値で取得します。対象物との距離が取得できない場合は、-1 を返します。

その他のコンポーネント

ActivityStarter 

ActivityStarter コンポーネントは、他のアクティビティ（***43**）を起動するためのコンポーネントです。画面には表示されないコンポーネントです。

このコンポーネントを使って自分のアプリから、メールアプリを起動したり、Google マップを起動したり、ブラウザを起動したりできます。

***43** アクティビティというのは聞きなれない言葉だと思いますが、簡単に言うとアプリとほぼ同じ意味です。

プロパティ

Action	起動したいアクティビティのアクションを設定します。例えば、メールアプリを起動したい場合は、android.intent.action.VIEW と入力します。ちなみに、ブラウザを起動したい場合も、android.intent.action.VIEW です。
ActivityClass	起動したいアクティビティのクラスの名前を設定します。
ActivityPackage	起動したいアクティビティのパッケージの名前を設定します。
DataType	起動したいアクティビティに渡すデータタイプです。
DataUri	起動したいアクティビティに渡す URI です。例えば、メールアプリを起動したい場合は、mailto:hoge@example.com?subject=hello と入力します。ブラウザを起動する時に URL を渡すのであれば、http://google.com のように URL を入力します（***44**）。
ExtraKey	起動したいアクティビティに渡す追加情報のキーです。
ExtraValue	プロパティの値を識別するキーを設定します。
ExtraValue	起動したいアクティビティに渡す追加情報の値です。
ResultName	起動したアクティビティから返される値を識別するための名前です。
Result	起動したアクティビティから返される値です。
ResultType	起動したアクティビティから返される値のタイプです。
ResultUri	起動したアクティビティから返される URI です（***45**）。

***44** URI というのは、上記の例のように様々なデータを表すものです。例えば、位置情報であれば geo:37.8,-122.23 などもあります。

***45** Result 系のプロパティは開発者向けのプロパティです。あまり使うこともないので、混乱を招かないためにも説明を省略します。

補足

プロパティにセットする値は自分で考えるものではなく、実際の作り方としては、Web サイトやブログに以下のように書いてあるテキストをセットして使うことが多いです。

例 1 Google マップに位置情報を渡して呼び出します。

プロパティ	値
Action	android.intent.action.VIEW
DataUri	geo:35.65904,139.74545?z=18

これ以外のプロパティは空のままで OK です。

geo:緯度,経度?z=ズーム率 という書式で設定します。ズーム率は、2（最小ズーム）か

ら 23（最大ズーム）の値で指定します。これで、以下のように Google マップが呼び出されて、指定した緯度,経度が中心になるように表示されます。ちなみに、例の緯度,経度は、東京タワーの緯度,経度です。

▲Google マップが呼び出される

例 2　Google マップを呼び出して、検索します。

プロパティ	値
Action	android.intent.action.VIEW
DataUri	geo:0,0?q=東京タワー

これ以外のプロパティは空のままで OK です。

geo:0,0?q=検索ワード という書式で設定します。これで、以下のように Google マップのアプリが呼び出されて、指定した検索ワードで検索して表示されます。

▲検索して表示される

例 3 ブラウザに URL を渡して呼び出します。

プロパティ	値
Action	android.intent.action.VIEW
DataUri	http://www.google.co.jp/

これ以外のプロパティは空のままで OK です。

URL をそのまま設定します。これで、以下のようにブラウザが呼び出されて、指定した URL が表示されます。

▲ブラウザを呼び出す

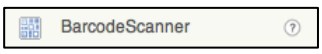
イベント

AfterActivity(text result)	起動したアクティビティが終了した後に呼び出されます。
result	起動したアクティビティから返される値

メソッド

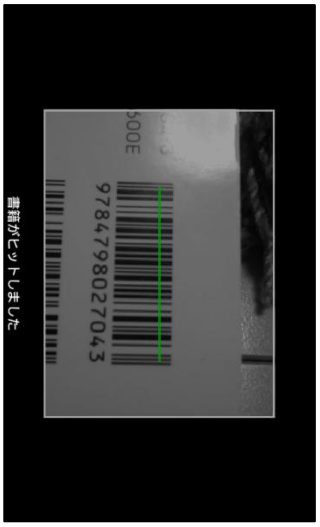
ResolveActivity()	アクティビティが起動できるかチェックできます。起動できる場合は、何かしらの文字列が取得できます。起動できない場合は、空の文字列が取得できます。
StartActivity()	アクティビティを起動します。

BarcodeScanner



BarcodeScanner コンポーネントは、QR コードスキャナーアプリを起動して、バーコードや QR コードを読み取るためのコンポーネントです。

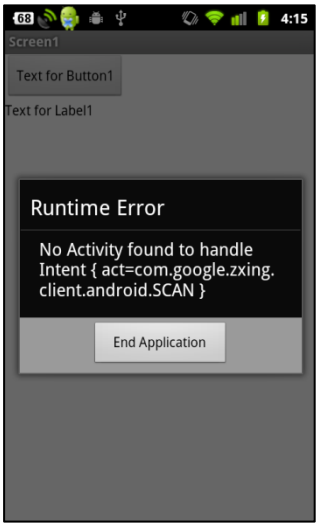
▲商品のバーコード



▲書籍のバーコード

▲URL などの QR コード

QR コードスキャナーがインストールされていない端末では、以下のようなエラーになってしまうので、あらかじめ、Android マーケットから「QR コードスキャナー」をインストールしてください。



▲エラー画面

プロパティ

Result	読み取ったバーコードや QR コードの値が入っています。

イベント

AfterScan(text result)	読み取りが終わった後に呼び出されます。
result	読み取ったバーコードや QR コードの値が入っています（*46）。

***46**　読み取ったバーコードや QR コードによって、以下のような値が入っています。

対象	値（例）	応用
商品	1923055026001	いろいろと
書籍	9784798027043	Amazon で書籍検索
QR コード	http://www.google.co.jp/	ブラウザで表示

メソッド

DoScan()	読み取りを開始します。

BluetoothClient

BluetoothClient コンポーネントは、Bluetooth（***47**）クライアントとして通信を行うためのコンポーネントです。

***47**　Bluetooth とは、デジタル機器用の近距離無線通信規格の 1 つで、Bluetooth マウスやヘッドフォンなどが有名ですね。Android 携帯でも Bluetooth が搭載されているので、この Bluetooth を使って通信を行うことができます。Bluetooth の通信を行うには、Bluetooth やプログラミングの知識が必要なので、ちょっと難しいです。初心者の方は読み飛ばしても問題ありません。

このコンポーネントで扱うことのできる Bluetooth のプロファイルは、Serial Port Profile (SPP)です。

このコンポーネントからペアリングを行うことはできないので、あらかじめ端末側でペアリングを行っておく必要があります。

プロパティ

CharacterEncoding	キャラクターエンコーディングです。
DelimiterByte	デリミタのバイト値です。ReceiveText/ReceiveSignedBytes/ReceiveUnsignedBytes メソッドの numberOfBytes パラメータにマイナスの値が指定された時に、このバイト値を受信するまで読み込みを続けます。
HighByteFirst	チェックされている場合は、ビッグエンディアンとしてデータを扱います。逆に、チェックされていない場合は、リトルエンディアンとしてデータを扱います。通信したい Bluetooth 機器や Bluetooth サーバ側の仕様と合わせて設定します。
AddressesAndNames	ペア設定済みで接続可能な Bluetooth 機器のアドレスと名前のリストです。このリストの要素を Connect メソッドのパラメータに指定

	すると接続できます。
Available	Android 端末で Bluetooth が利用可能な場合は true、利用できない場合は false です。
Enabled	この Bluetooth コンポーネントが有効な場合は true、無効な場合は false です。
IsConnected	接続済みの場合は true、未接続の場合は false です。

メソッド

number BytesAvailableToReceive()	受信できるバイト数を返します。
boolean Connect(text address)	Bluetooth 機器と Serial Port Profile (SPP)で接続します。接続に成功した場合は true を返します。
address	接続したい Bluetooth 機器の MAC アドレスを指定します。このパラメータは MAC アドレスの後ろにスペースが入っていれば、それ以降の文字列は無視します。なので、AddressesAndNames プロパティの要素を MAC アドレスと名前にスプリットなどせずに、そのまま指定することができます。
boolean ConnectWithUUID(text address, text uuid)	UUID を指定して Bluetooth 機器と接続します。接続に成功した場合は true を返します。
address	接続したい Bluetooth 機器の MAC アドレスを指定します。
UUID	UUID を指定します。
Disconnect()	Bluetooth 機器との接続を解除します。
boolean IsDevicePaired(text address)	Bluetooth 機器がペアリング済み（ペア設定済み）の場合は true を返します。
number ReceiveSigned1ByteNumber()	1 バイト符号付き整数（signed 1-byte）を受信して返します。
number ReceiveSigned2ByteNumber()	2 バイト符号付き整数（signed 2-byte）を受信して返します。
number ReceiveSigned4ByteNumber()	4 バイト符号付き整数（signed 4-byte）を受信して返します。
list ReceiveSignedBytes(number numberOfBytes)	指定されたバイト数を受信して 1 バイト符号付き整数のリストを返します。

numberOfBytes	受信したいバイト数。負数を指定した場合は、DelimiterByte プロパティの値を受信するまで読み込みを続けます。
text ReceiveText(number numberOfBytes)	指定されたバイト数のテキストを受信して返します。
numberOfBytes	受信したいバイト数。負数を指定した場合は、DelimiterByte プロパティの値を受信するまで読み込みを続けます。
number ReceiveUnsigned1ByteNumber()	1 バイト符号なし整数 (unsigned 1-byte) を受信して返します。
number ReceiveUnsigned2ByteNumber()	2 バイト符号なし整数 (unsigned 2-byte) を受信して返します。
number ReceiveUnsigned4ByteNumber()	4 バイト符号なし整数 (unsigned 4-byte) を受信して返します。
list ReceiveUnsignedBytes(number numberOfBytes)	指定されたバイト数を受信して 1 バイト符号なし整数のリストを返します。
numberOfBytes	受信したいバイト数。負数を指定した場合は、DelimiterByte プロパティの値を受信するまで読み込みを続けます。
Send1ByteNumber(text number)	1 バイト整数を送信します。
Send2ByteNumber(text number)	2 バイト整数を送信します。
Send4ByteNumber(text number)	4 バイト整数を送信します。
SendBytes(list list)	1 バイト整数のリストを送信します。
SendText(text text)	テキストを送信します。

BluetoothServer

BluetoothServer コンポーネントは、Bluetooth サーバとして通信を行うためのコンポーネントです。

基本的には BluetoothClient コンポーネントと同じですが、サーバとして接続待ちをしたり、接続された時のイベントなどが追加されています。

プロパティ

CharacterEncoding	キャラクターエンコーディングです。
DelimiterByte	デリミタのバイト値です。ReceiveText/ReceiveSignedBytes/ReceiveUnsignedBytes メソ

	ッドの numberOfBytes パラメータにマイナスの値が指定された時に、このバイト値を受信するまで読み込みを続けます。
HighByteFirst	チェックされている場合は、ビッグエンディアンとしてデータを扱います。逆に、チェックされていない場合は、リトルエンディアンとしてデータを扱います通信したい Bluetooth 機器や Bluetooth サーバ側の仕様と合わせて設定します。
Available	Android 端末で Bluetooth が利用可能な場合は true、利用できない場合は false です。
Enabled	この Bluetooth コンポーネントが有効な場合は true、無効な場合は false です。
IsAccepting: boolean	接続待ち状態の場合は true、接続待ち状態でない場合は false です。
IsConnected	接続済みの場合は true、未接続の場合は false です。

イベント

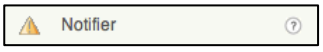
ConnectionAccepted()	接続ができた時に呼び出されます。

メソッド

AcceptConnection(text serviceName)	接続待ちを開始します。
serviceName	Bluetooth サーバに適当な名前を付けます。
AcceptConnectionWithUUID(text serviceName, text uuid)	UUID を指定して接続待ちを開始します。
serviceName	Bluetooth サーバに適当な名前を付けます。
UUID	UUID を指定します。
number BytesAvailableToReceive()	受信できるバイト数を返します。
boolean Connect(text address)	Bluetooth 機器と接続します。接続に成功した場合は true を返します。
address	接続したい Bluetooth 機器の MAC アドレスを指定します。このパラメータは MAC アドレスの後ろにスペースが入っていれば、それ以降の文字列は無視します。なので、AddressesAndNames プロパティの要素を MAC アドレスと名前にスプリットなどせずに、そのまま指定することができます。

boolean ConnectWithUUID(text address, text uuid)	UUID を指定して Bluetooth 機器と接続します。接続に成功した場合は true を返します。
address	接続したい Bluetooth 機器の MAC アドレスを指定します。
UUID	UUID を指定します。
Disconnect()	Bluetooth 機器との接続を解除します。
boolean IsDevicePaired(text address)	Bluetooth 機器がペアリング済み（ペア設定済み）の場合は true を返します。
number ReceiveSigned1ByteNumber()	1 バイト符号付き整数（signed 1-byte）を受信して返します。
number ReceiveSigned2ByteNumber()	2 バイト符号付き整数（signed 2-byte）を受信して返します。
number ReceiveSigned4ByteNumber()	4 バイト符号付き整数（signed 4-byte）を受信して返します。
list ReceiveSignedBytes(number numberOfBytes)	指定されたバイト数を受信して 1 バイト符号付き整数のリストを返します。
numberOfBytes	受信したいバイト数。負数を指定した場合は、DelimiterByte プロパティの値を受信するまで読み込みを続けます。
text ReceiveText(number numberOfBytes)	指定されたバイト数のテキストを受信して返します。
numberOfBytes	受信したいバイト数。負数を指定した場合は、DelimiterByte プロパティの値を受信するまで読み込みを続けます。
number ReceiveUnsigned1ByteNumber()	1 バイト符号なし整数（unsigned 1-byte）を受信して返します。
number ReceiveUnsigned2ByteNumber()	2 バイト符号なし整数（unsigned 2-byte）を受信して返します。
number ReceiveUnsigned4ByteNumber()	4 バイト符号なし整数（unsigned 4-byte）を受信して返します。
list ReceiveUnsignedBytes(number numberOfBytes)	指定されたバイト数を受信して 1 バイト符号なし整数のリストを返します。
numberOfBytes	受信したいバイト数。負数を指定した場合は、DelimiterByte プロパティの値を受信するまで読み込みを続けます。

Send1ByteNumber(text number)	1バイト整数を送信します。
Send2ByteNumber(text number)	2バイト整数を送信します。
Send4ByteNumber(text number)	4バイト整数を送信します。
SendBytes(list list)	1バイト整数のリストを送信します。
SendText(text text)	テキストを送信します。
StopAccepting()	接続待ちを停止します。

Notifier 

Notifier コンポーネントは、ダイアログを表示したり、ログを出力したりできるコンポーネントです。

メソッド

LogError(Text message) LogInfo(Text message) LogWarning(Text message)	ログを出力します。ログはユーザには見えない開発者用の情報です。
ShowAlert(Text message)	メッセージを表示します。メッセージは数秒後に自動的に消えます。
message	表示したいメッセージ
ShowChooseDialog(Text message, Text title, Text button1Text, Text button2Text)	ダイアログを表示します。ユーザがどちらかのボタンをタップすると、AfterChoosing イベントが呼び出され、このイベントでユーザがどちらのボタンをタップしたかを取得することができます。
message	表示したいメッセージ
title	表示したいタイトル
button1Text	左側のボタンのテキスト
button2Text	右側のボタンのテキスト
ShowMessageDialog(Text message, Text title, Text buttonText)	ダイアログを表示します。
message	表示したいメッセージ
title	表示したいタイトル
buttonText	ボタンのテキスト
ShowTextDialog(Text message, Text title)	ダイアログを表示します。ユーザが OK ボタンを

	タップすると、AfterTextInput イベントが呼び出され、このイベントでユーザが入力したテキストを取得することができます。
message	表示したいメッセージ
title	表示したいタイトル

イベント

AfterChoosing(Text choice)	ShowChooseDialog メソッドで表示されたダイアログのボタンがタップされた後に呼び出されます。
choice	ユーザがタップしたボタンのテキストが入っています。
AfterTextInput(Text response)	ShowTextDialog メソッドで表示されたダイアログのボタンがタップされた後に呼び出されます。
response	ユーザが入力したテキストが入っています。

SpeechRecognizer

SpeechRecognizer コンポーネントは、音声認識を行うコンポーネントです。音声認識とは、ユーザの音声を認識して、テキストに変換する技術です。

GetText メソッドを呼び出すと、以下のような画面が表示されます。ユーザの発声が終わったことを自動的に認識して AfterGetting イベントが呼び出されます。音声をテキストに変換した結果は、このイベントの result ブロックや Result プロパティに入っています。

▲SpeechRecognizer コンポーネント使用画面

プロパティ

Result	音声認識した結果のテキストが入っています。

イベント

AfterGetting(Text result)	音声認識が終わったあとに呼び出されます。
result	音声認識した結果のテキストが入っています。
BeforeGettingText()	音声認識が始まる前に呼び出されます。

メソッド

GetText()	音声認識を開始します。

TextToSpeech 

TextToSpeech コンポーネントは、音声合成を行うコンポーネントです。音声合成とは、テキストをコンピュータが読み上げて音声として再生する技術で、Text-to-Speech とも呼ばれます。

例えば、Good morning というテキストを音声合成すると「ぐっどもーにんぐ」と読み上げてくれます。

英語、フランス語、ドイツ語、イタリア語、スペイン語などの様々な言語で音声合成が可能です。が、残念ながら日本語は対応されていません。

プロパティ

Country	音声合成したいテキストの国を指定します。
Language	音声合成したいテキストの言語を指定します。音声合成は、上記のプロパティで指定した言語と国に合わせて読み上げてくれます（*48）。

*48　指定できる国と言語は以下のとおりです。

言語	Language プロパティ	Country プロパティ
スペイン語	spa	ESP, USA
ドイツ語	deu	AUT, BEL, CHE, DEU, LIE, LUX
フランス語	fra	BEL, CAN, CHE, FRA, LUX
イタリア語	ita	CHE, ITA
英語	eng	AUS, BEL, BWA, BLZ, CAN, GBR, HKG, IRL, IND, JAM, MHL, MLT, NAM, NZL, PHL, PAK, SGP, TTO, USA, VIR, ZAF, ZWE

Language プロパティは、必ず設定してください。Country プロパティは、設定しなくても動作します。もし、細かく国の違いを指定したい場合は、上記から 1 つ選んで設定してください。例えば、英語には、

たくさん Country プロパティに設定できる値がありますが、特にこだわりがなければ USA などをオススメします。

イベント

AfterSpeaking(Text result)	音声合成して読み上げが終わった後に呼び出されます。
result	音声合成の結果が true か false で入っています。
BeforeSpeaking()	音声合成して読み上げが始まる前に呼び出されます。

メソッド

Speak(Text message)	テキストを音声合成して読み上げます。
message	音声合成したいテキスト