Qt Creatorでアニメーション

このウェビナーでは

- 第零部：Qt Creatorについて
  - UI ツアー
- 第一部：C++とQMLの関係
  - クイックレビュー
  - Qt C++のどの部分がQtQuickに使われる？
  - 手続き型言語で3日の作業を宣言型言語で数時間？
  - そもそもQtQuickって何？
- 第二部：アニメーションを使って画面にアクセントを
  - プロパティ、シグナル、スロット、バインディング、イベント、ハンドラ、アイテム、 (Q)オブジェクト
  - アニメーションの基礎
  - 状態(State)、移行(Transition)、ふるまい(Behavior)
  - イージングカーブ

第零部: Qt Creatorについて

Qt CreatorはC++, JavaScriptそしてQMLを用いてGUIアプリケーション開発を容易にさせる統合型クロスプラットフォームである

Qt Creator is a cross-platform C++, JavaScript and QML integrated development environment which simplifies GUI application development.

Source: https://en.wikipedia.org/wiki/Qt_Creator

ティップス

- [ESC] 編集へ戻る
- [F1] ヘルプ
- [Ctrl+Tab] & [Shift+Ctrl+Tab] ドキュメント参照履歴
- [Alt-Left] & [Alt-Right] 編集場所履歴

- **[Ctrl-K]** ロケーター

- 早見表 (KDAB社作)
  http://www.cheat-sheets.org/saved-copy/qtcreator.pdf

第一部: C++とQMLの関係

- QObject
  - Q_PROPERTIES
  - イベント
  - シグナル
  - スロット
  - メモリ管理とQObjectツリー
- QEventLoop
- QVariants と Qt MetaType

QObject:

- メモリー管理ツリー
- イベントとイベントハンドラ
- プロパティ
- シグナル：Qtのリスナー様式？
- スロット：ただの関数、だが`::connect`できる

QObject: ツリー

- QObjectコンストラクタで親QObject *を持つ
- QObjectは親の子供として登録される
- 親が破棄されたとき、すべての子QObjectもデストラクタを呼び出され破棄される

```
QObject::QObject(QObject *parent)
    : d_ptr(new QObjectPrivate)
{
    Q_D(QObject);
```

QObject: イベント

- QObjectsもイベントをハンドルする
- イベントが処理されない時、それは親のハンドラに渡される.
- イベントタイプのチェック、イベントの取り回しなど

```
bool QObject::event(QEvent *e)
{
    switch (e->type()) {
```

QObject: プロパティ

```
class Q_AUTOTEST_EXPORT QQuickLoader : public QQuickImplicitSizeItem
{
    Q_OBJECT
    Q_PROPERTY(bool active READ active WRITE setActive NOTIFY activeChanged)
    Q_PROPERTY(QUrl source READ source WRITE setSource NOTIFY sourceChanged)
    Q_PROPERTY(QQmlComponent *sourceComponent READ sourceComponent
          WRITE setSourceComponent RESET resetSourceComponent NOTIFY sourceComponentChanged)
    Q_PROPERTY(QObject *item READ item NOTIFY itemChanged)
    Q_PROPERTY(Status status READ status NOTIFY statusChanged)
    Q_PROPERTY(qreal progress READ progress NOTIFY progressChanged)
    Q_PROPERTY(bool asynchronous READ asynchronous WRITE setAsynchronous NOTIFY asynchronousChanged)
```

- ランタイムがプロパティを名前でマップできるようにさせるマクロ
- 静的動的問わず

QObject: signals

- コールバックのQtバージョン
- QObject間の変化やイベント通知のフレキシブルな方法
- mocツールによるコンパイル時での自動生成（実装）

QObject: slots

- シグナルと連結できる関数
- スロットは直接呼び出しもしくはキューイングで
  - 直接　＝　関数呼び出し
  - キューイング　＝　QEventloop経由

The QEventLoop

- イベントループがないとQObjectsは無意味
- シグナルとスロットをドライブ
- イベント処理を提供
- Qtのコードに生命を

QVariant と the meta-type システム

- QVariantはQtから提供されたC++のunionクラス
  - すべてのC++スカラータイプ
  - ほとんどのQtCoreの値クラス
  - QtCoreではない他のQtタイプ
- カスタマイズ Qvariant
  - QMetaTypeが新しいタイプの追加を手助け
  - シリアライズ - デシリアライズ

宣言型言語に対して

- QtQuickとQMLは当初は考慮に入っていない
- Qtが宣言型フレームワークに対応できるとしたら？
- オブジェクト指向のフレームワークと宣言型プログラムの相性

Part II: QtQuickと宣言型スタイル

- QtQuick概要
- QtQuickのアニメーション

QtQuickとは？QMLとの違いは?

- QMLはQtメタ言語（Qt Meta Language）
- QtQuickはQMLアプリケーションを実行するランタイム

（Javaの場合、言語、API、ランタイムすべてJavaと呼ばれる）

- Javaと違い、QMLのための１つのランタイム

QtQuick: 構成要素

- QML 言語パーサー
- QML エンジン
- QtQuick ビューポート
  - QQuickView, QQmlApplicationEngine
- JavaScript ランタイム (現在 “V4”)
- QSceneGraph
- シーングラフ描画（レンダラー）
  - バッチ処理でスマートな描画と非バッチ処理
- シーングラフバックエンド
  - OpenGL, Direct3D, OpenVG

QtQuick APIs

- C++
- JavaScript
- QML

QtQuick C++ API

- （ほとんど）すべてのQtQuickはQObject
- QQuickItemはQObjectのサブクラス
- QtQuickで可視できるすべてのクラスはQQuickItemのサブクラス
  - QQuickPaintedItemを経てQPainterがテクスチャーにqpaintしてシンプル実装のもの
  - もしくはQScenegraphNodesで実装したもの
- タイプとして登録してQMLで生成
  - `qmlRegisterType<NiceType>("io.qt.neat_api", 1, 2, "NeatType");`
- もしくはC++内で生成してコンテキストで追加
- QAbstractItemModelもQML Model-Viewとして有用

いつC++ APIを使うべきか？

- ロジック上緊急を要するプログラム部分
  - JavaScriptも可能だが、重たい処理には不向き
- パフォーマンスが重要とされる箇所
- シンプル実装で極小な部品
- モデル・ビュー

QtQuick JavaScript API

- QtQuickの JavaScript エンジン”V4”の導入
- 他のJSランタイムと以下の点が異なる
  - シグナル・スロットがある
  - 独自のプライベートコンテキストがJSファイルにある
  - あやまって変数をグローバルオブジェクトに追加することはない
- *.jsファイルのインポート、QMLバインディングの組込、関数呼び出し

JavaScript functions

```
function hiThere() {
        console.log("Hi There!")
}
```

```
var heloo = function() {
    console.log("This type MUST be in an imported JS file")
}
```

```
property var hello: function() {
      console.log("HELLO")
}
```

```
MouseArea {
    onClicked: {
        JS.heloo()
        hello()
        hiThere()
    }
}
```

```
QML debugging is enabled.
qml: This type MUST be in an imported JS
file
qml: HELLO
qml: Hi There!
```

QtQuick QML API

- JSONのようなコード
- JSONではない
- QMLはML
- HTMLのようなマークアップ言語

```
Row {
    anchors.fill: parent
    Repeater {
        model: internal.displaySpeed.length
        Text {
            text: internal.displaySpeed[index]
            font.family: "Times”
            font.pixelSize: 78
            font.italic: true
            width: 40
            height: font.pixelSize + 6
        }
    }
}
```

Item {}

- QMLで座標を持つすべてのタイプのベースタイプが Item
- Item 自身は一切描画をしない
  - レイアウトの道しるべとして有用
  - QtQuickで描画できるすべてのベースクラスに有用
  - QQuickItem

```
Item {
        id: item
        x: 0; y: 0 // no need to add ; after each line
        width: 100; height: 200
}
```

QtObject {}

- QMLでのQObjectに相当
- QtObject自体はプロパティはobjectNameしか持たない
- id: QMLのプロパティで、QObjectではなくQtQuickからの発祥

```
QtObject {
        id: anObject
        objectName: "myFavoriteObject”
        property string aString: "It's a QString"
        property int anInteger: parent.width * 42
}
```

:

- バイディング演算子
- QtQuickで強力な機能のひとつ

```
property <type> <name> : <scalar | object | formula>
```

- 右が変化すると左が再評価される
  - これがバインディング(binding)、割り当て(assignment)ではない
- [Gauges4 デモコード, ‘BauerGauge’]

シグナルとスロット

- QObjectのシグナルは右のように追加します。
  - QtQuickがハンドラーを追加します

```
property int count: 0
signal clicked(int count)
```

- C++と同じようにスロットは関数です。QMLではパブリックで呼び出し可能です。

- このようにシグナルとスロットは繋ぐことはできますが…

```
MouseArea {
    anchors.fill: parent
    Component.onCompleted: {
        ma.clicked.connect(hiThere)
    }
}
```

```
signalName.connect(slotName)
```

```
property int count: 0
signal signalName(int arg1)
onSignalName: {
    count += arg1
}
```

QtQuickランタイムでのイベントとハンドラ

- プロパティ変化のイベントは自動的に発生します
- シグナルのハンドラも自動的に生成されます

```
Rectangle {
    id: rect
    color: "#0becca”
    property int count: 0
    onCountChanged: console.log("New count", count)
    onWidthChanged: console.log("New width", width)
    onColorChanged: console.log("New color", color)
}
```

何をアニメーションできるか？

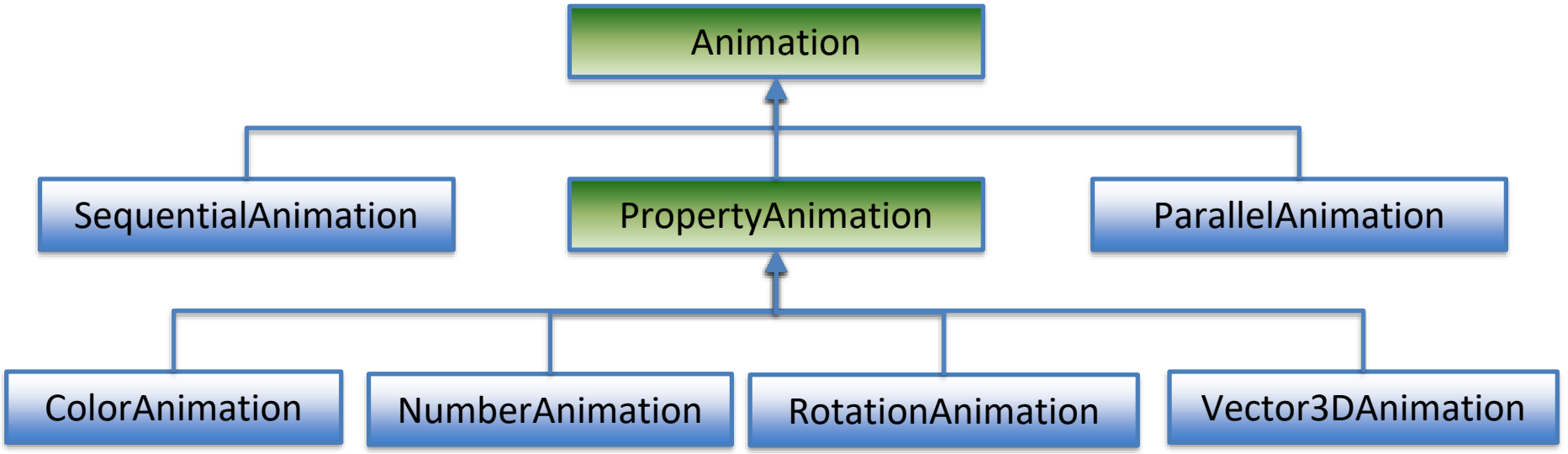
- プロパティに対してすべて
- アニメーションに適さないものもある
  - Booleanのようなもの
- ナンバーものがベター
  - 時間との相性、始まりと終わりの指定
- 色もアニメーション可能
  - GPUに負荷をかけるので注意。特にグラデーション

どのようにアニメーション？

```
PropertyAnimation {
    id: animationSample
    duration: 1500
    target: rect
    properties: "x,y”
//          property: "x”
    from: 0
    to: 720
    easing.type: Easing.InOutQuad
}
```

アニメーション関連QML

- QML アニメーションファミリー



ステート（State）

- 各QML Itemはステート(state)プロパティを持つ
- 各QML ItemはStateオブジェクトの配列（states）も持つ
- Statesはプロパティの変更を含む
- （states demo）

トランジション（Transitions）

- ステートの変更はアニメーションではない
- トランジションはステートが変更した時に駆動するアニメーションを含む
- (transitions demo)

モデルビューフレームワークのトランジション

- トランジションはQMLモデル・ビューでも使われる（追加、移動、削除）
- シンプルデモ：ドラッグと再配列

アニメーションを制御する

- アニメーションはstart()とstop()のメソッドを持つ
  - 他に pause(), resume(), restart(), complete() など
- アニメーションは started, stopped, finished シグナルがあり：
- onStarted, onStopped, onFinished とそれぞれ対応するハンドラも持つ

グループアニメーション

- パラレルアニメーション（ParallelAnimations）
- シーケンシャルアニメーション（SequentialAnimations）
- デモ：PSGAnimation

ポーズアニメーション, パスアニメーション スクリプトアクション, プロパティアクション

- ポーズアニメーション（PauseAnimation）はSequentialAnimation内で使用
  - ParallelAnimationではあまり使われないかと
- パスアニメーション（PathAnimation）は経路（パス）に従ってエレメントを 動かす
  - 経路（パス）をアニメーションするのではない
- スクリプトアクション（ScriptAction）はアニメーションではない
  - シーケンシャルもしくはパラレルアニメーション内でスクリプトの挿入に使われる
- プロパティアクション（PropertyAction）はアニメーションではない
  - スクリプトアクションのプロパティの変更版

より良い挙動のために

- Statesは有用ですが、アニメーションに適さない場合もあります
- プロパティの変更をするだけで挙動の見え方が良くなることもあります
- デモ：[Behavior source code sample: Gauge2]

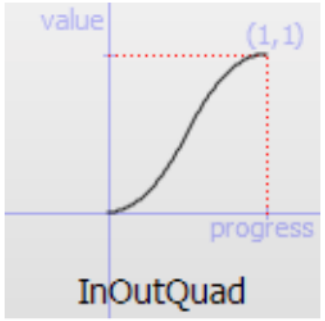
イージングカーブ（変化率カーブ）

› QEasingCurve, 4.6からサポート

› サポートするイージングカーブのドキュメントがあります

https://doc.qt.io/qt-5/qeasingcurve.html

Constant
QEasingCurve::InOutQuad



Easing curve for a quadratic (t^2) function: acceleration until halfway, then deceleration.

まとめ

- アニメーションはQMLアプリケーション内でほとんどどこでも使えます
- 値が時間に応じて変化することを主に利用しています
- イージングカーブを使うことでより良い見栄えになります
- 色んな方法でアニメーションを実行できます
  - JavaScriptで命令する
  - ステート変化時のトランジション
  - 値が変化した時の挙動（Behavior）
  - 他のアニメーションのグループ化

- アニメーションはQtQuickコードが小規模の割にアプリへのインパクトは大

- デモコード：https://github.com/reworled/qtvc2020.git

ご視聴ありがとうございました

- もっと学ぶ：www.resources.qt.io
- コンタクト： www.qt.io/contact-us/

フォロー:

- Twitter: @Qtproject
- Facebook: /qt/
- Linkedin: /company/theqtcompany/
- Youtube: /Qtstudios/