

LLDBを活用したデザインチェック

SwiftとPythonを組み合わせた画像の差分の確認

藤野 翠太 (@Scior)

自己紹介🦙

株式会社AppBrew グロースチーム エンジニア

- コスメアプリ「LIPS」でグロース担当
- iOS、たまにサーバーや分析やPjM
- 写真と旅行が好き
- アイコンはリャマではなくアルパカ
- パワポ派(今回はKeynote)

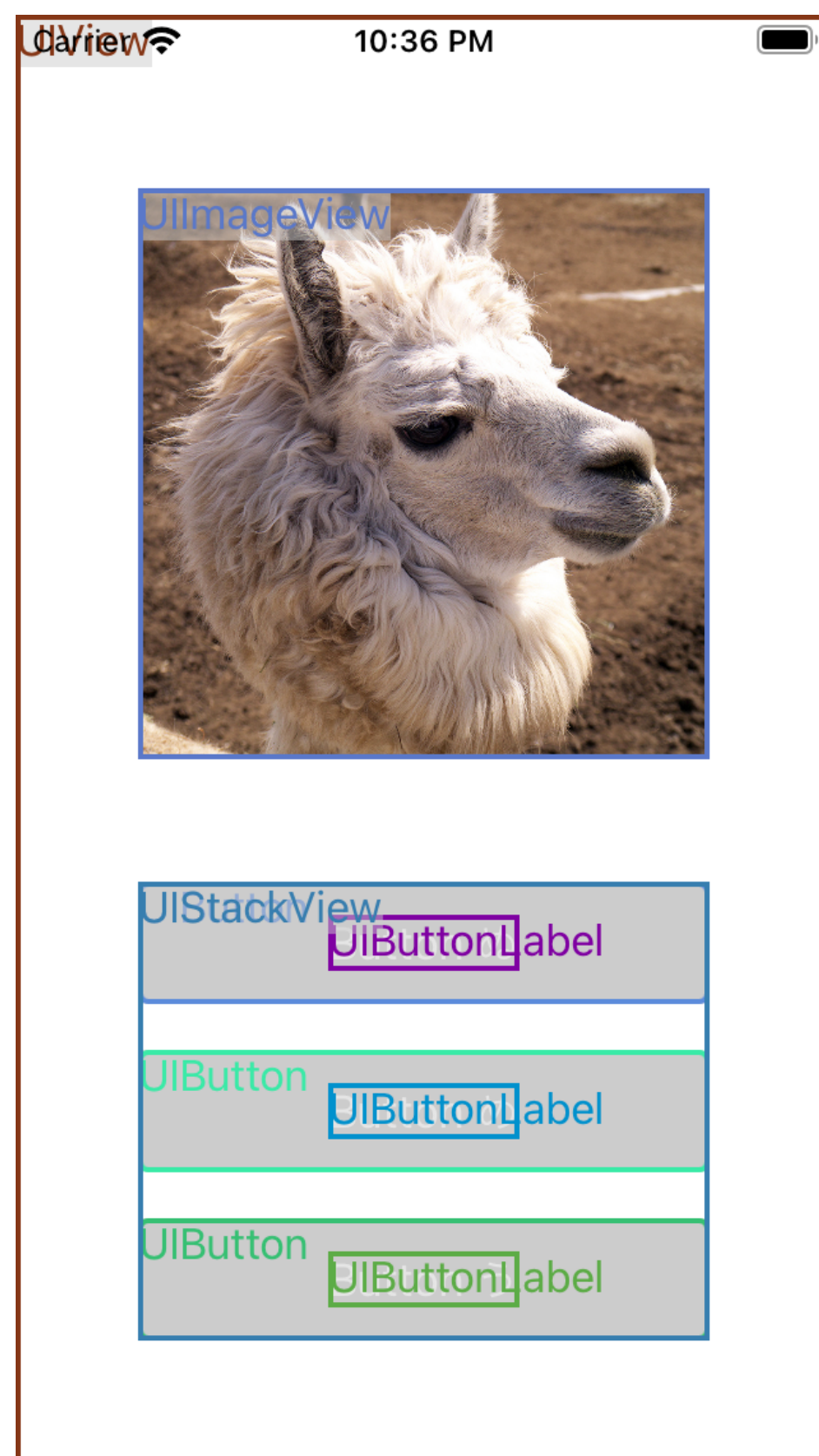


風景や広角好きの人集まれ～📷

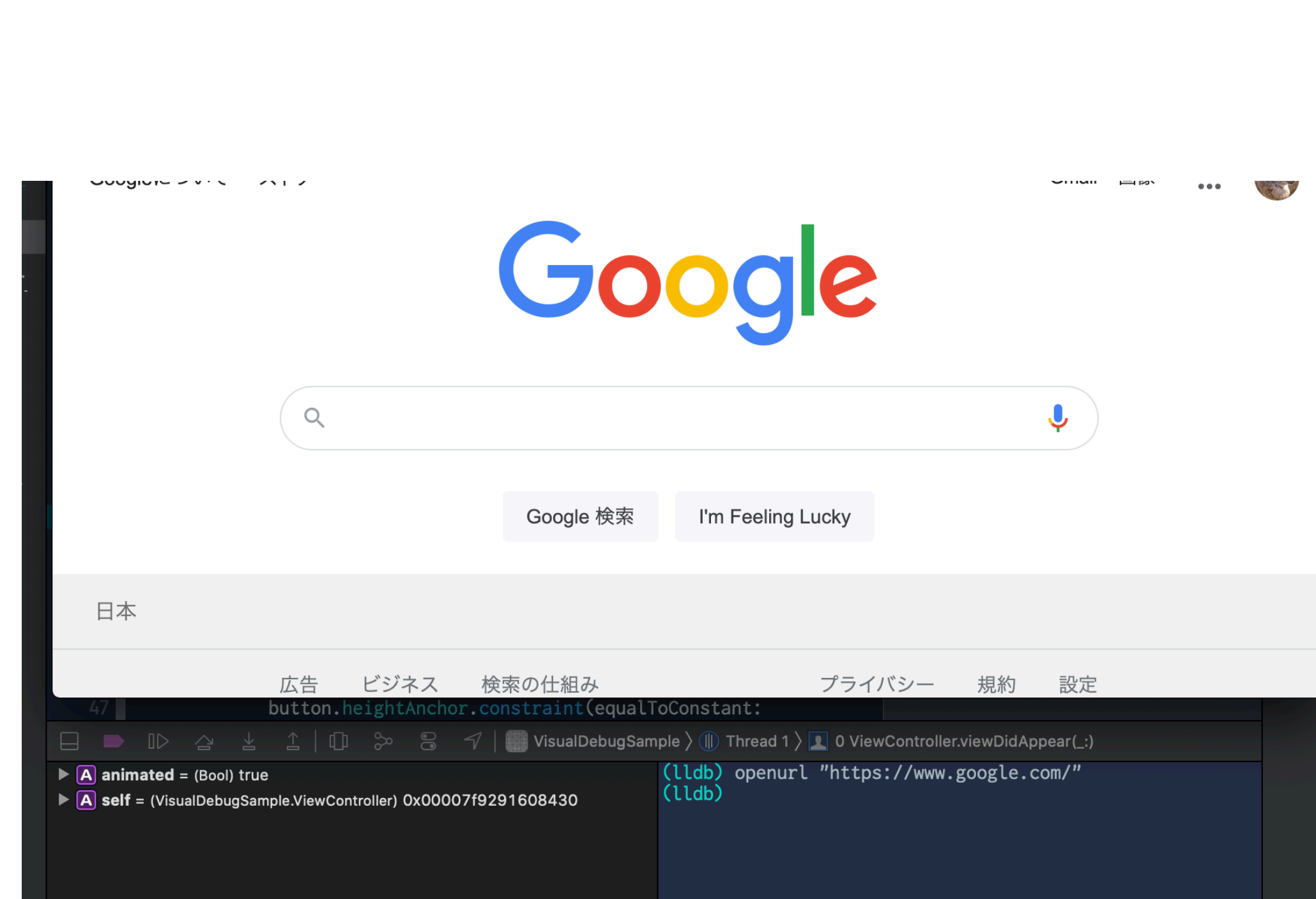
導入

```
(lldb) e view.isHidden = false
```

とにかくいろいろできる



UIコンポーネントのクラス名を表示



なんとなくGoogleを開く

Pythonと組み合わせると
できることが増える

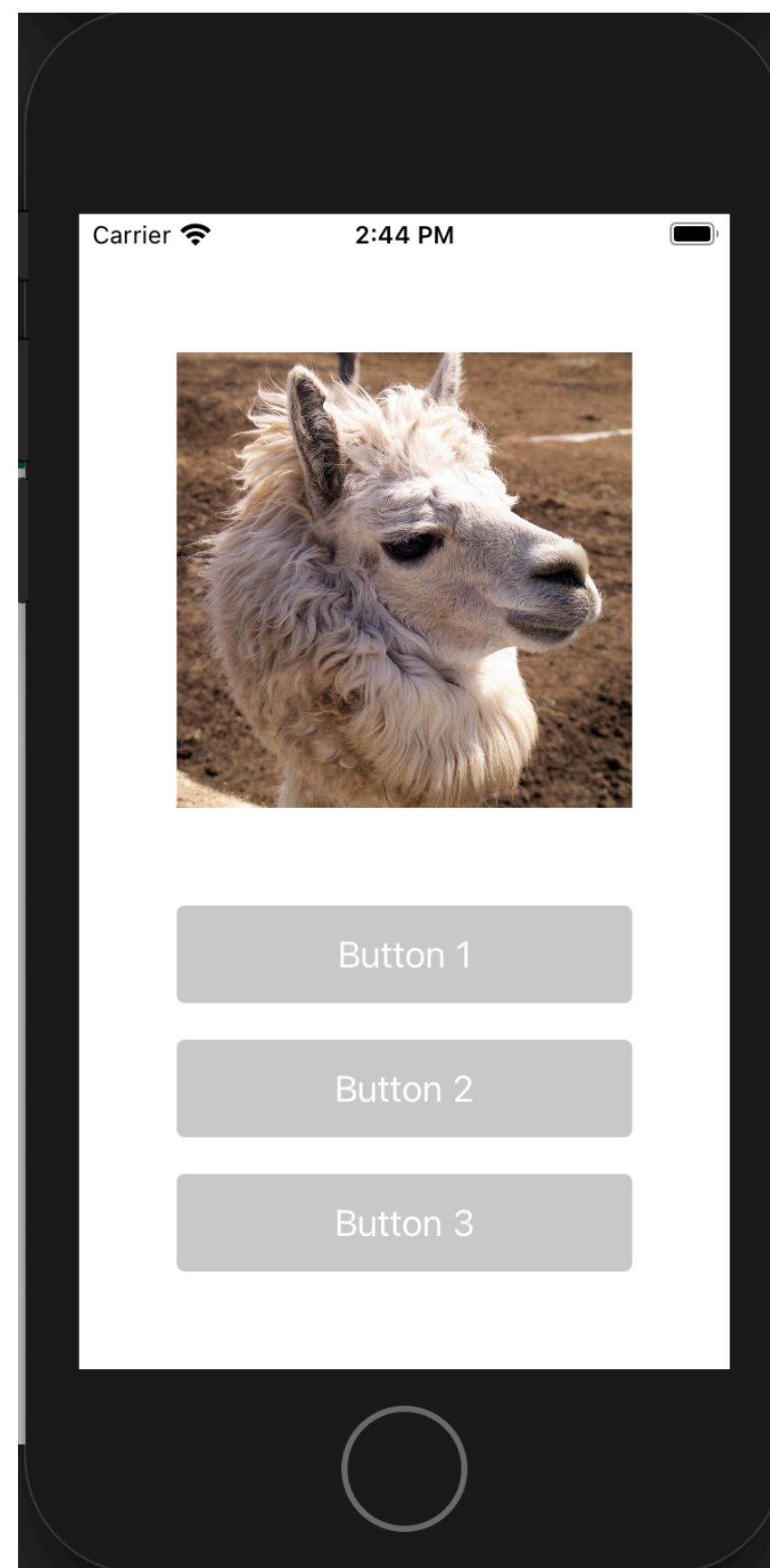
今回のゴール



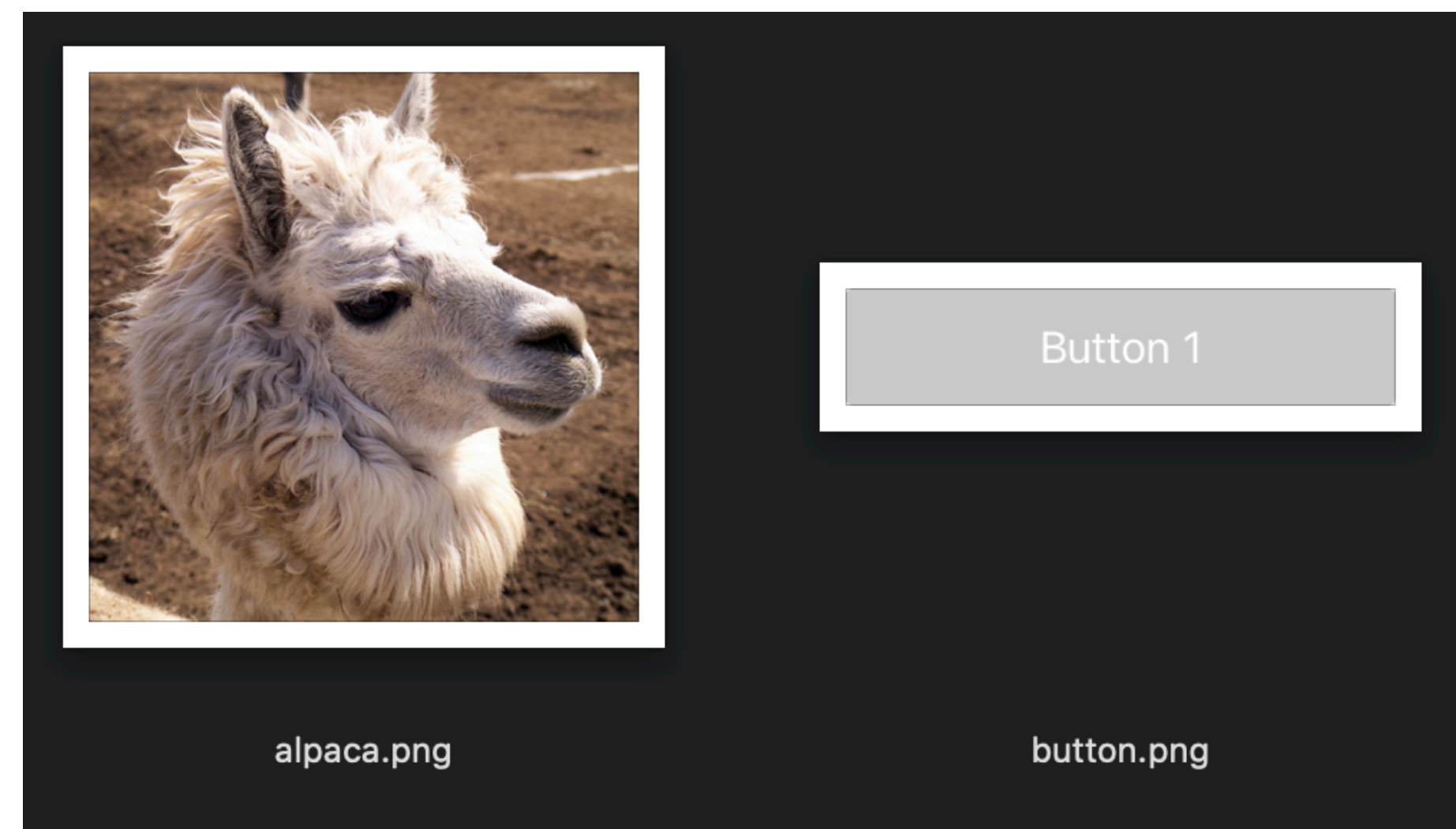
デザインチェックの際に
手動で位置を合わせるのをなくす

実践(基本編)

特定のUIViewを画像ファイルとして書き出す



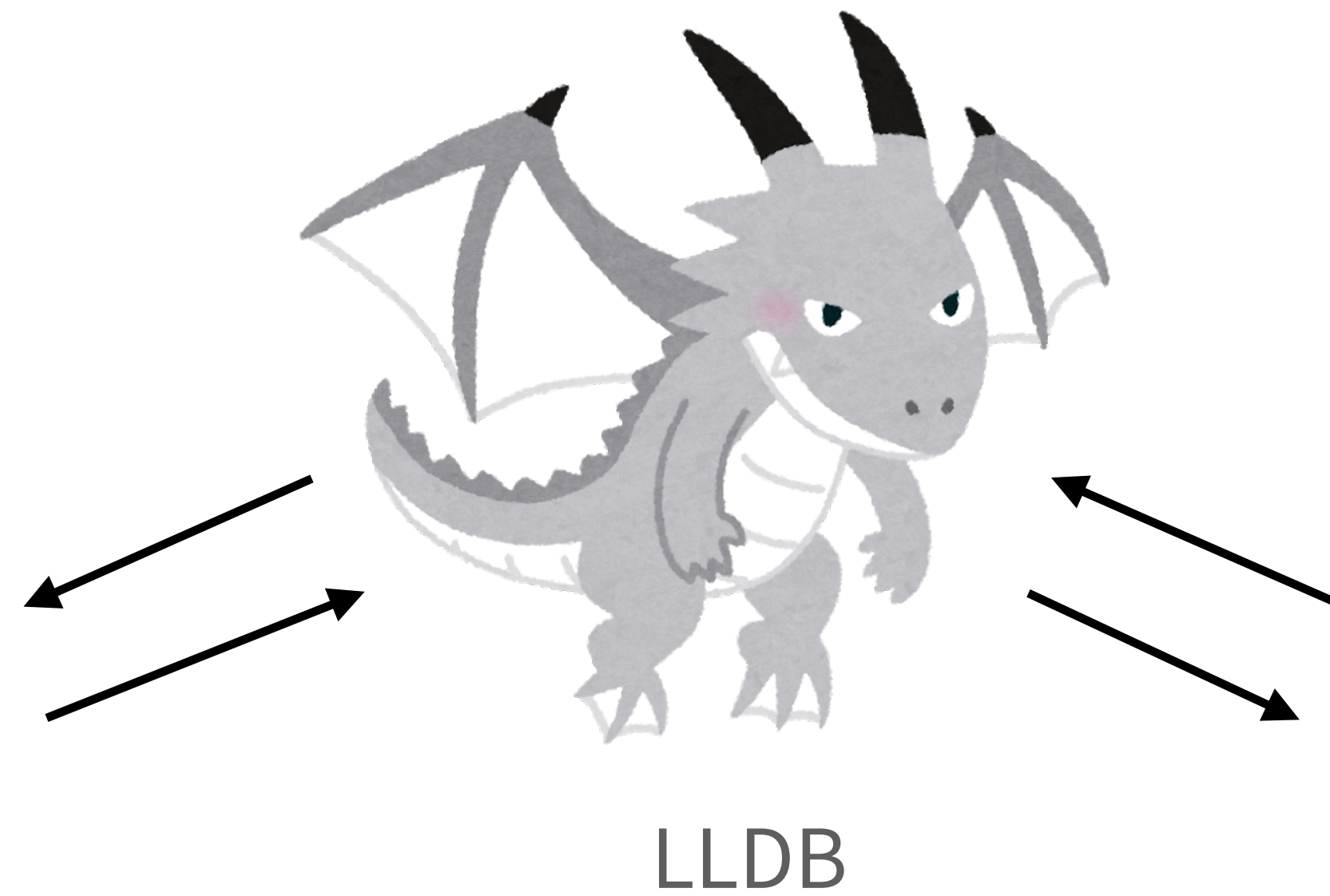
Save



特定のUIViewを画像ファイルとして書き出す



UIViewからDataへの変換
UIGraphicsImageRenderer



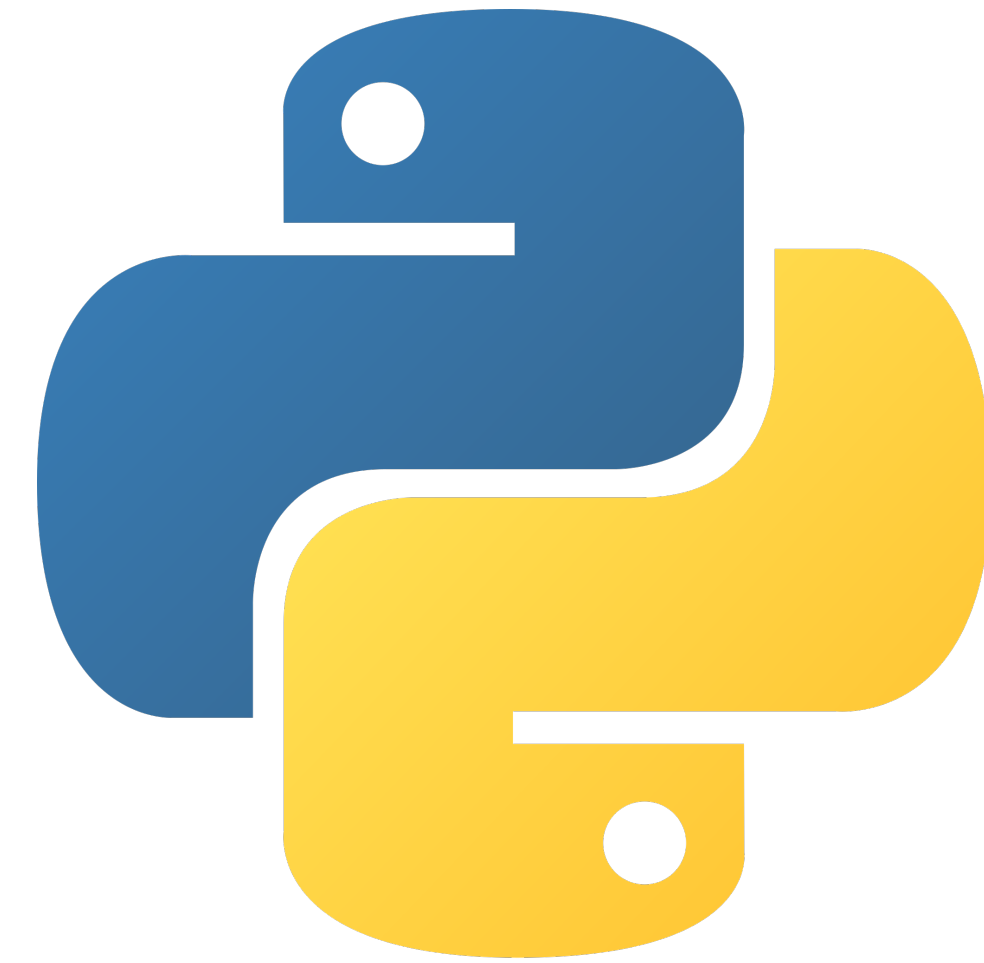
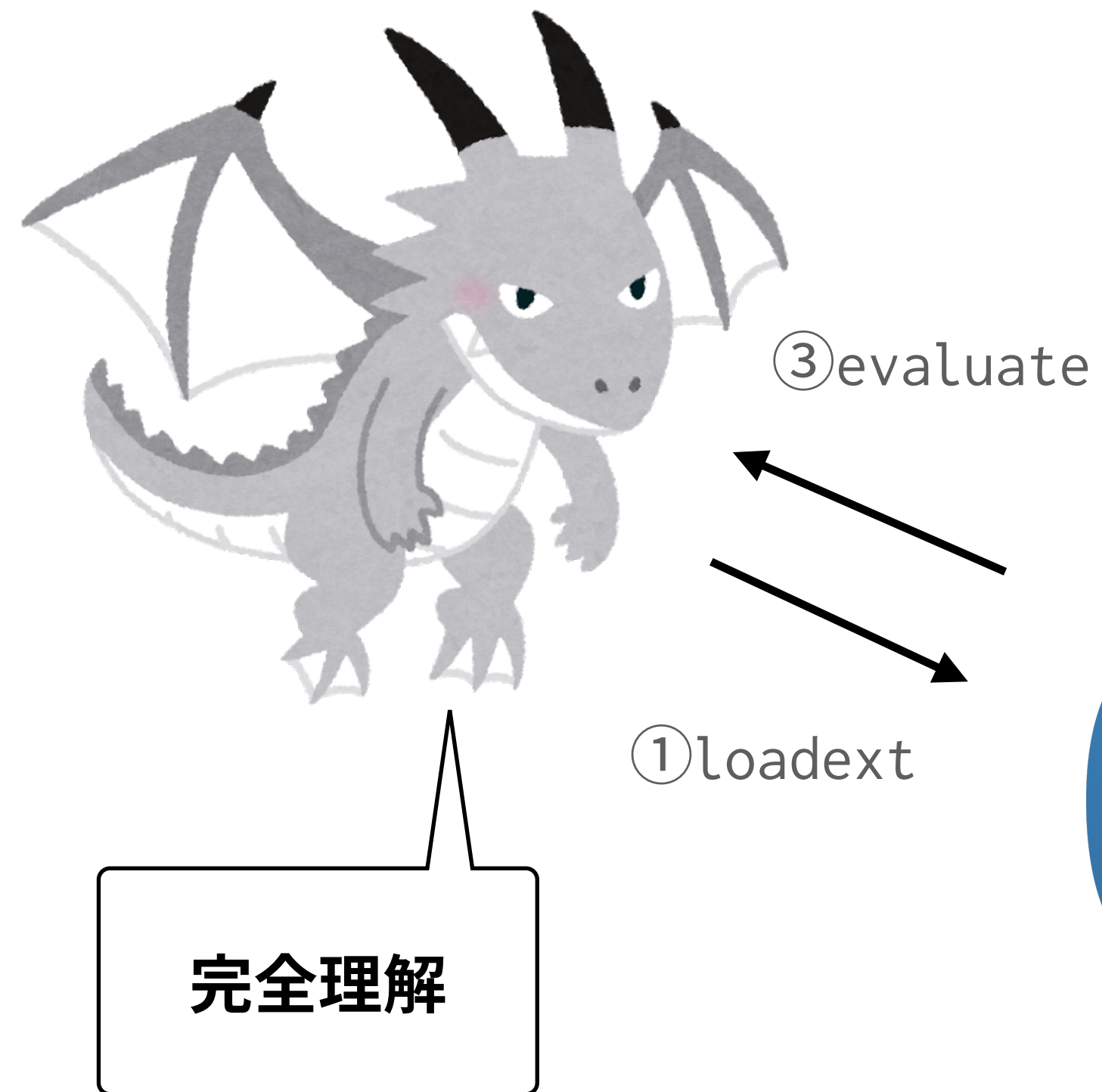
メモリからの読み込みと
ファイルへの書き込み
SBProcess.ReadMemory
Lib/io

```
(lldb) saveimage imageView /tmp/alpaca.png
```

Swiftコードのロード(イメージ)



DebugExtensions.swift



②DebugExtensions.swiftを読み込み

Swiftコードをロードするコマンド(loadext)

```
@lldb.command("loadext")
def load(debugger, command, result, dict):
    path = os.path.join(os.path.dirname(__file__), 'swift/DebugExtensions.swift')
    with open(path, 'r') as f:
        common.evaluate(f.read())
```

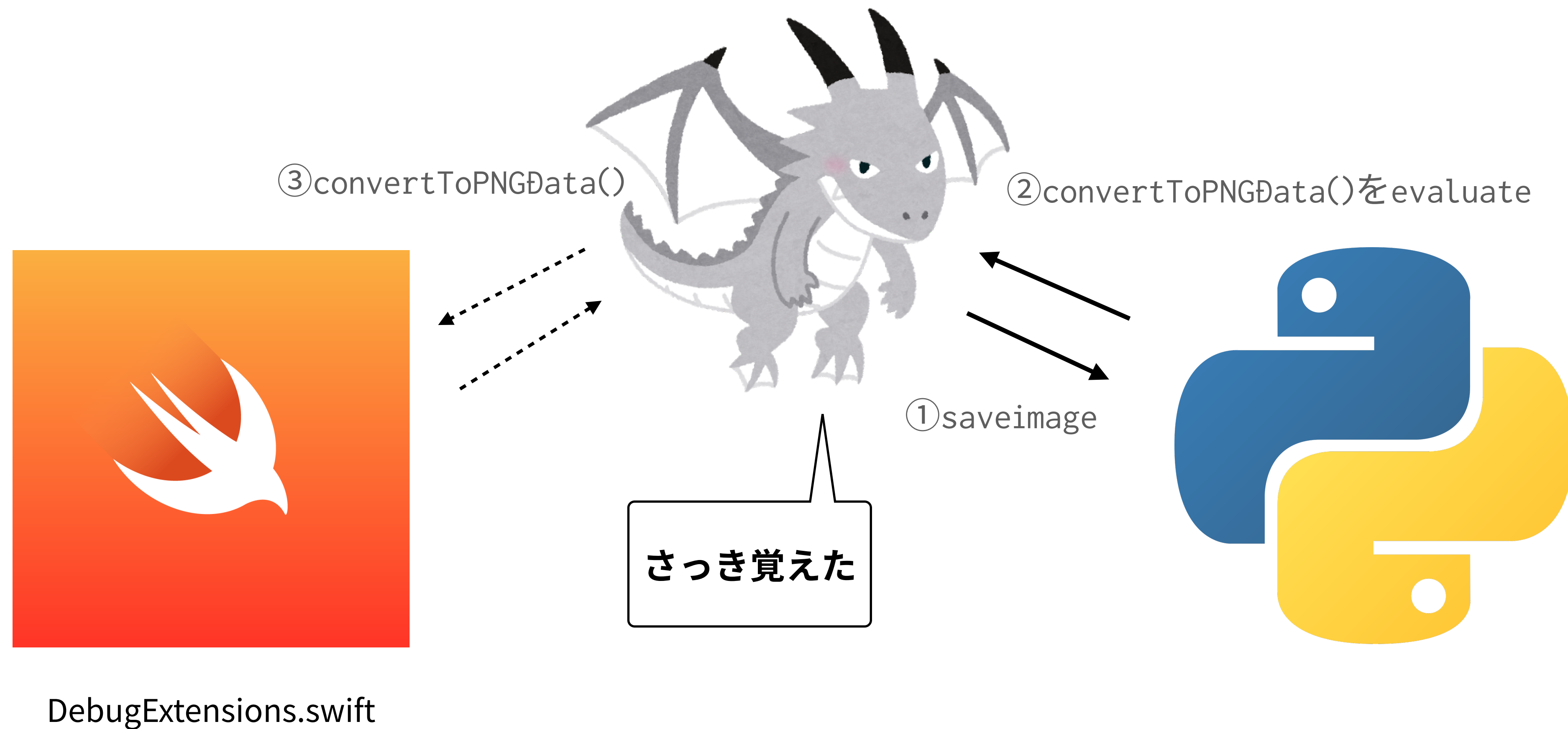
* common.evaluateについては最後に補足

DebugExtensions.swift

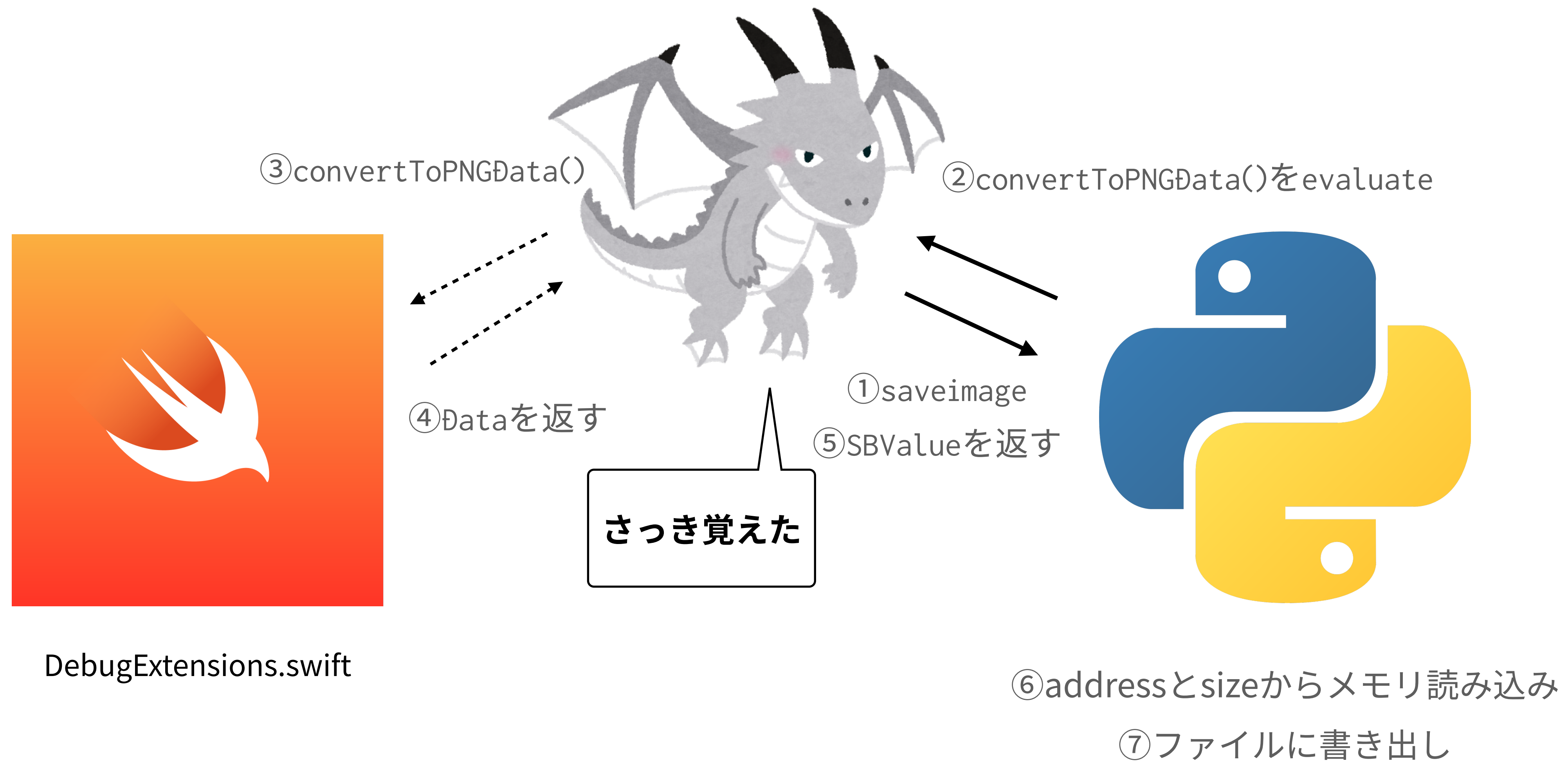
```
extension UIView {  
    func takeSnapshot() -> UIImage? {  
        let renderer = UIGraphicsImageRenderer(bounds: bounds)  
        return renderer.image { context in  
            layer.render(in: context.cgContext)  
        }  
    }  
  
    func convertToPNGData() -> Data {  
        return takeSnapshot()!.pngData()!  
    }  
}
```

もちろん#if DEBUG ~ #endifで囲ってコンパイルしても良い

saveimageの全体像(イメージ)



saveimageの全体像(イメージ)



saveimageの全体像(コード)

```
@lldb.command("saveimage")
def save_image(debugger, arguments, result, dict):
    view, path = arguments.split()

    var_name = str(uuid.uuid4()).replace('-', '')
    common.evaluate('let $%s = %s.convertToPNGData()' % (var_name, view))

    address_str = common.evaluate('($%s as NSData).bytes' % var_name).GetObjectDescription().split()[1]
    address = int(address_str, 16)
    size = int(common.evaluate('$%s.count' % var_name).GetValue())

    process = lldb.debugger.GetSelectedTarget().GetProcess()
    error = lldb.SBError()
    data = process.ReadMemory(address, size, error)
    with open(path, "wb") as f:
        f.write(data)
```

メソッドの定義とコマンドの追加

```
@lldb.command("saveimage")
def save_image(debugger, arguments, result, dict):
```

@lldb.command は command script add -fと同じでLLDBにコマンドを追加する

convertToPNGData()の実行

```
common.evaluate('let $%s = %s.convertToPNGData()' % (var_name, view))
```

let \$hoge = imageView.convertToPNGData()を評価する

Dataのアドレスとサイズの取得

```
address_str = common.evaluate('($%s as NSData).bytes' % var_name).GetObjectDescription().split()[1]
address = int(address_str, 16)
size = int(common.evaluate('%s.count' % var_name).GetValue())
```

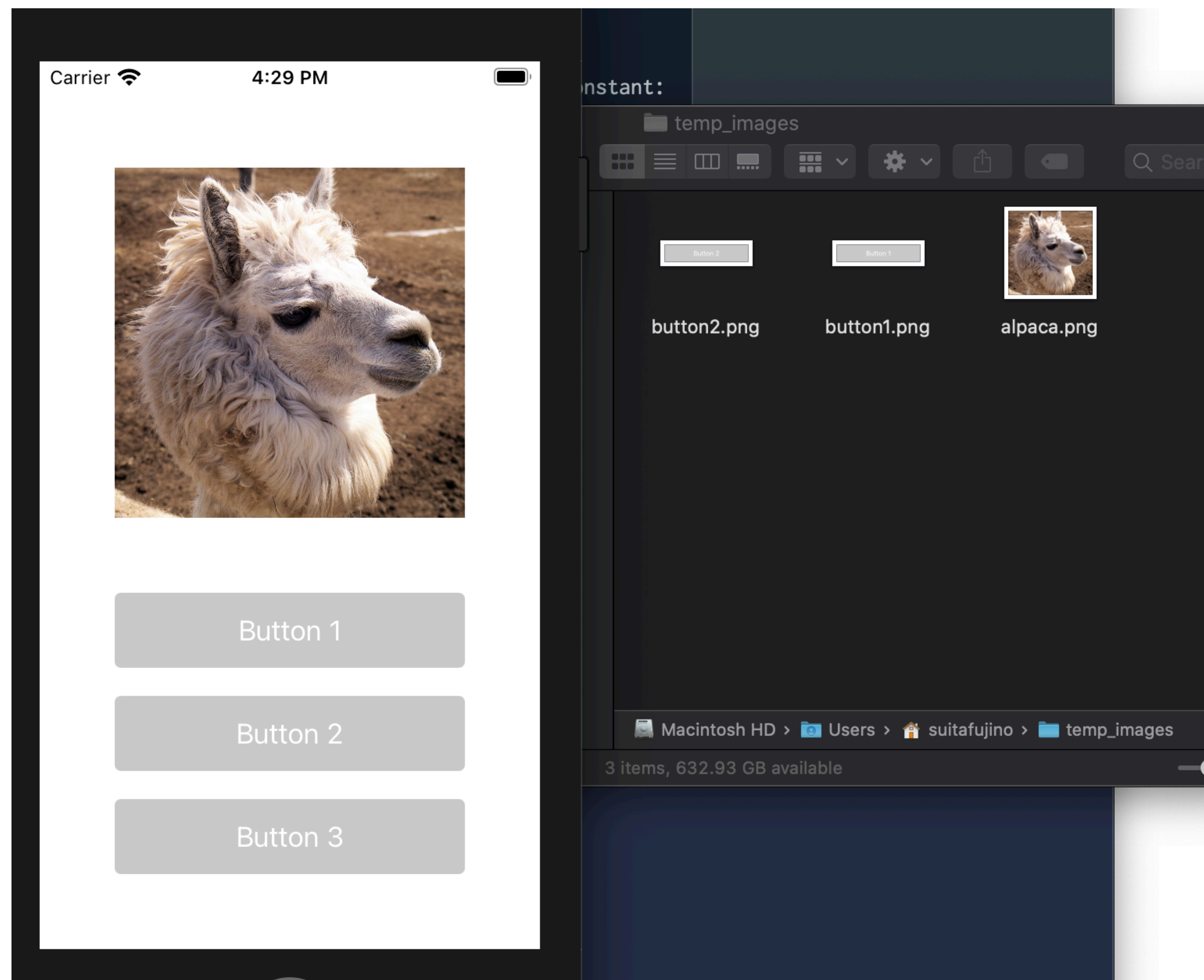
(\$hoge as NSData).bytesと\$hoge.countでアドレスとサイズを取得
(Obj-C++で記述するともう少しキレイになる)

メモリからの読み込みとファイルへの書き込み

```
process = lldb.debugger.GetSelectedTarget().GetProcess()
error = lldb.SBError()
data = process.ReadMemory(address, size, error)
with open(path, "wb") as f:
    f.write(data)
```

SBProcess.ReadMemoryで画像をPython側に読み込み、ファイルに書き込み

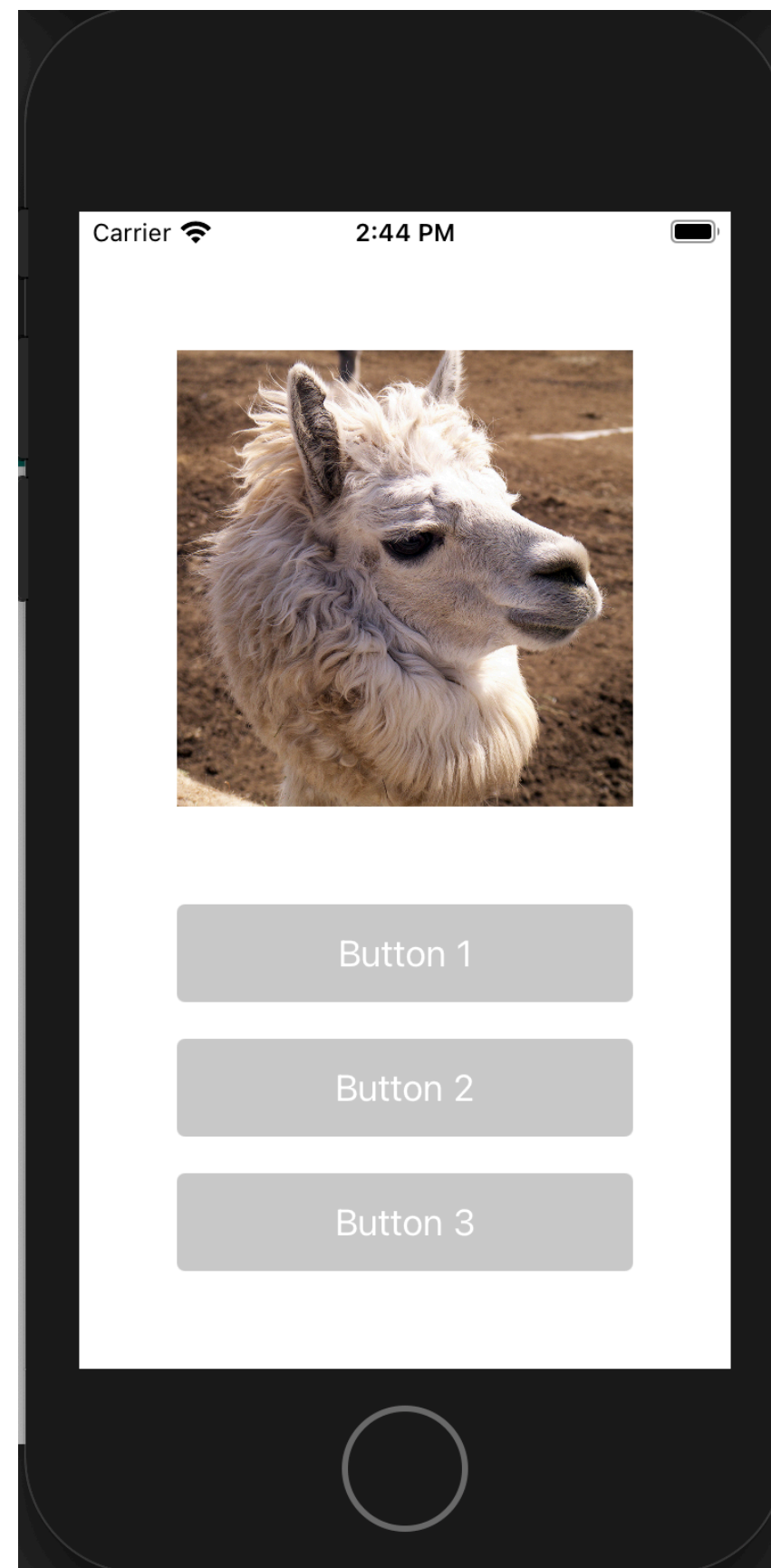
完成



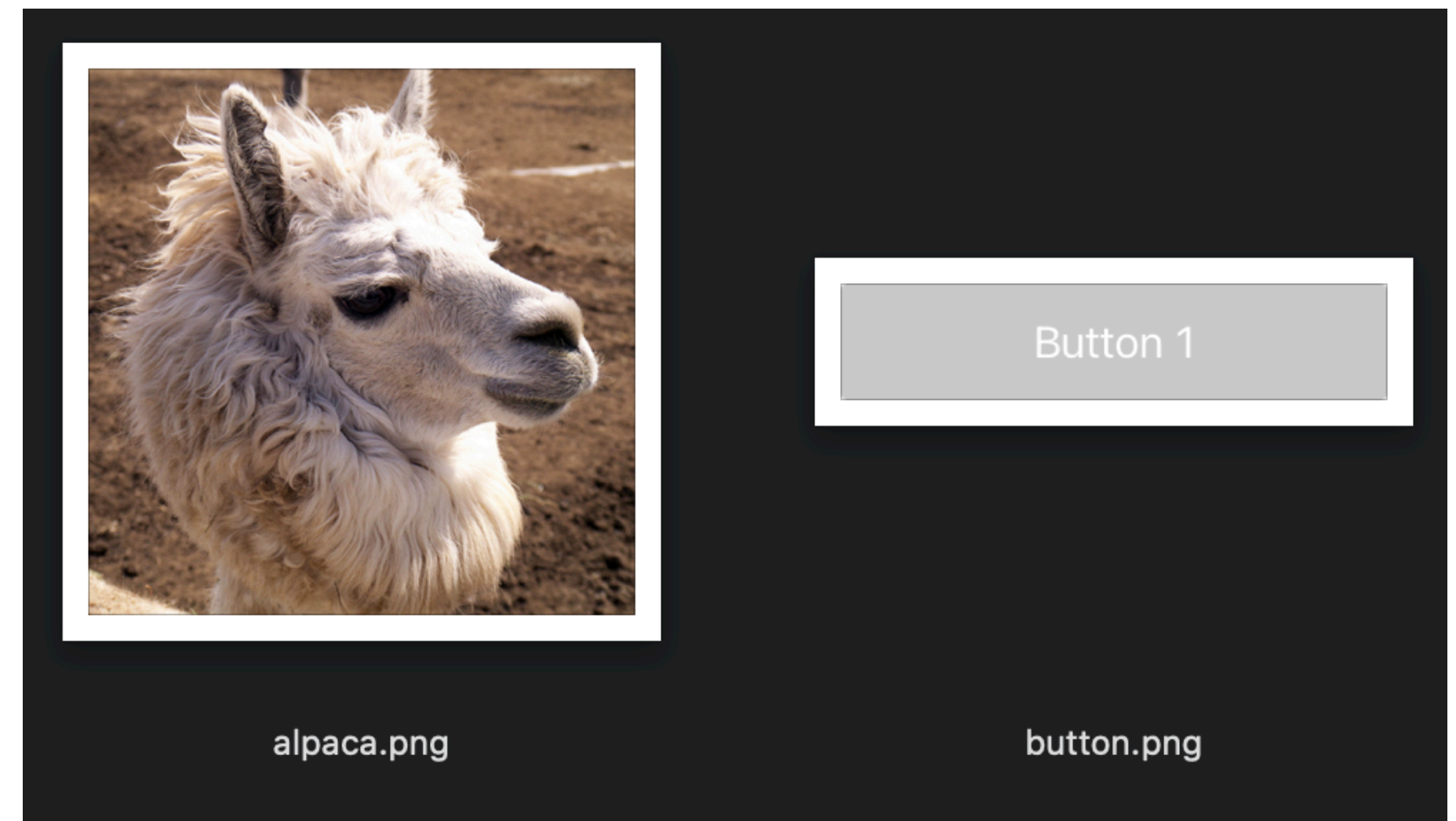
*動画はGitHubに移しています

実践(応用編)

ロードした画像を特定のUIViewに重ねる



Overlay

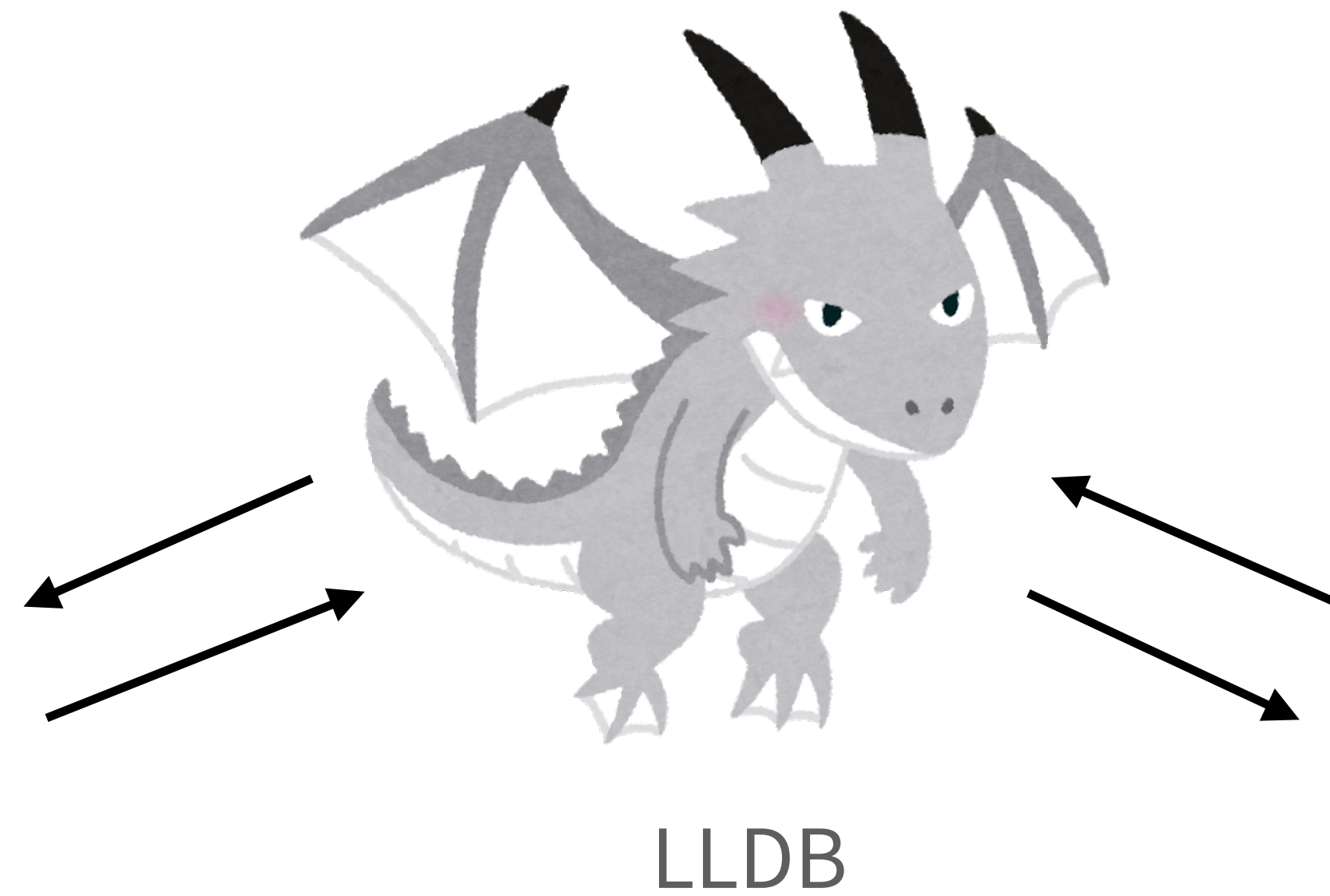


デザインデータ

ロードした画像を特定のUIViewに重ねる



メモリ領域の準備と
DataからUIImageへの変換
UnsafeMutablePointer
UIImage.init(data:)



ファイルからの読み込みと
メモリへの書き込み
Lib/io
SBProcess.WriteMemory

```
(lldb) overlayimage imageView /tmp/alpaca.png
```

ImageBufferクラス

```
final class ImageBuffer {
  typealias Pointer = UnsafeMutablePointer<UInt8>
  private let size: Int
  let pointer: Pointer

  init(size: Int) {
    self.size = size
    pointer = Pointer.allocate(capacity: size)
  }

  deinit {
    pointer.deallocate()
  }

  func getData() -> Data {
    let bufferPointer = UnsafeMutableBufferPointer(start: pointer, count: size)
    return .init(buffer: bufferPointer)
  }
}
```

overlayimageの全体像

```
@lldb.command("overlayimage")
def overlay_image(debugger, arguments, result, dict):
    view, path = arguments.split()
    with open(path, 'rb') as f:
        data = f.read()

    buf_name = common.generateVarName()
    common.evaluate('let $%s = ImageBuffer(size: %s)' % (buf_name, len(data)))
    address_str = common.evaluate('%s.pointer' % buf_name).GetObjectDescription().split()[1]
    address = int(address_str, 16)

    process = lldb.debugger.GetSelectedTarget().GetProcess()
    error = lldb.SBError()
    size = process.WriteMemory(address, data, error)

    view_name = common.generateVarName()
    common.evaluate('let $%s = DebugOverlayView(frame: %s.frame)' % (view_name, view))
    common.evaluate('%s.set(data: %s.getData())' % (view_name, buf_name))
    common.evaluate('%s.superview?.addSubview(%s)' % (view, view_name))
```

メモリ領域の確保

```
with open(path, 'rb') as f:
    data = f.read()
common.evaluate('let $%s = ImageBuffer(size: %s)' % (buf_name, len(data)))
address_str = common.evaluate('$%s.pointer' % buf_name).GetObjectDescription().split()[1]
address = int(address_str, 16)
```

ImageBufferで先に画像のサイズ分メモリを確保する

メモリへの書き込み

```
process = lldb.debugger.GetSelectedTarget().GetProcess()  
error = lldb.SBError()  
size = process.WriteMemory(address, data, error)
```

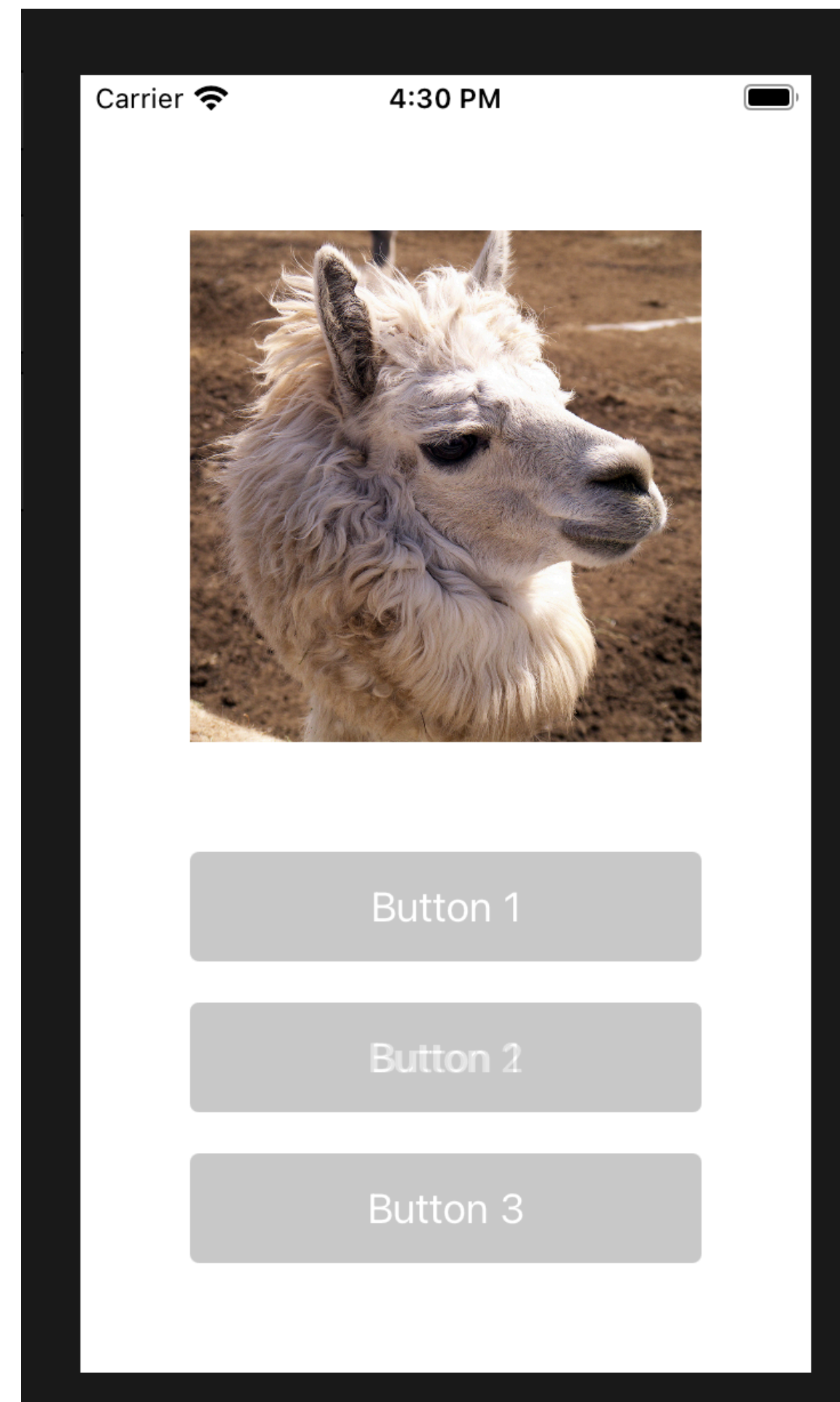
SBProcess.WriteMemoryで先ほどの領域に書き込み

デバッグ用のViewを追加して表示する

```
common.evaluate('let $%s = DebugOverlayView(frame: %s.frame)' % (view_name, view))
common.evaluate('$%s.set(data: %s.getData())' % (view_name, buf_name))
common.evaluate('%s.superview?.addSubview(%s)' % (view, view_name))
```

```
let $view = DebugOverlayView(frame: view.frame)
    $view.set(data: $buf.getData())
    view.superview?.addSubview($view)
```

完成😊



*動画はGitHubに移しています

With OpenCV and scikit-image (in progress)



実装したView



デザインデータ



cv2とskimageで差分検出
(UILabelとかはズレやすいので難しい)

サンプルコード

<https://github.com/Scior/LLDBVisualDebug>

Appendix

common.evaluateの実装例

```
def evaluate(exp):  
    options = lldb.SBExpressionOptions()  
    options.SetLanguage(lldb.eLanguageTypeSwift)  
  
    frame = (  
        lldb.debugger.GetSelectedTarget()  
        .GetProcess()  
        .GetSelectedThread()  
        .GetSelectedFrame()  
    )  
  
    return frame.EvaluateExpression(exp, options)
```

参考になるリンク集

- LLDB Python API: https://lldb.llvm.org/python_api.html
- facebook/chisel: <https://github.com/facebook/chisel>

ありがとうございました 🦙