Decide what language is right for you in Autodesk MotionBuilder

Kristine Middlemiss, Developer Consultant
Autodesk Developer Network

AUTODESK®
MOTIONBUILDER

Contents

Decide what language is right for you

Kristine Middlemiss, Developer Consultant
Autodesk Developer Network

1.0 Introduction to Autodesk MotionBuilder

- Autodesk MotionBuilder is the industry-leading, real-time 3D character animation software for games, feature film, and television productions.
- It is designed to complement Autodesk Maya and Autodesk 3ds Max software, as well as other Autodesk DirectConnect applications that support the Autodesk FBX file format.
- With its core focus on interactive real-time workflows, Autodesk MotionBuilder enables creative and technical artists to take on demanding, animation-intensive projects.
- Autodesk MotionBuilder is a professional package designed for 3D data acquisition, manipulation, and visualization, which makes it not only an animation productivity solution, but a tool that drives the iterative process of creativity.
- The software, available for Windows operating systems, natively supports the platform-independent Autodesk FBX 3D data interchange solution that allows Autodesk MotionBuilder software to integrate with applications in a production pipeline that supports FBX.
- In addition to powerful animation features and an intuitive interface, Autodesk MotionBuilder also includes dedicated tools for creating cameras, lights, shading, cartoon rendering, textures, shadows, and constraints.

1.1 Why use Programming in Autodesk MotionBuilder

- Automate repetitive, time-consuming tasks and extend features without leaving the Autodesk MotionBuilder environment.
- Support for the popular, easy-to-use Python scripting language allows production facilities to better integrate Autodesk MotionBuilder into their production pipeline.
- The Open Reality SDK can be used to create custom tools and features that plug directly into Autodesk MotionBuilder and extend its functionality.
- In-house developers can create project-specific functionality, which accommodates specific workflows and requirements, including custom file types.

1.2 Python Introduction

What is Python all about?

- Object Oriented

    - This means its class model supports advanced concepts such as polymorphism, operator overloading, and multiple inheritance.
    - Don't worry if you don't understand these terms you'll find they are much easier to learn with Python than with just about any other Object Oriented Programming language available.
- Open Source
    - Which means it is free; there are no restrictions on copying it, embedding it in your systems or shipping it with your products.
    - The beauty of free means tons of people are using it so there are lots and lots of forums, communities, extra tool being built, and lots of keen programmers that can help you when you get stuck.
- Mostly Interpreted
    - There is no need for an external compiler or debugger as you do not need to compile your code to run it.
- Used for Both Standalone programs and Scripting Applications
    - This multi-purpose language is great for Autodesk MotionBuilder because it allows you to use the standard Python language to do Windows type functionality and automation if you need too all in the same Autodesk MotionBuilder Python scripts.

Advantages

- Quicker Development Cycle
    - You can make changes on the fly, which means it is well suited to rapid prototyping, because you code, test and debug all within Autodesk MotionBuilder.
- It is Extremely Portable
    - Python is written in portable ANSI C, and compiles and runs on virtually every major platform in use today.
    - It is also backwards compatible.

Disadvantages

- Possibly Slower language compared to C++
    - Python is slower than its sister, Open Reality SDK, because it is not a fully compiled language such as C++ since it is an interpreted language; however speed depends largely on the complexity of the program.
- Discloser
    - There may be occasions when you'd like to release some functionality to an outside party but not disclose exactly how it was implemented, since Python is both the source code as well as what is finally executed, there is no way to separate the two.

1.3 What distinguishes Python from the OpenReality SDK?

Advantages of OpenReality

- C++ plug-ins runs much faster than Python scripts in the majority of situations.
- More class and function access than Python, for example creating custom devices and/or manipulators is only available in Open Reality SDK.
- Can derive from existing classes, e.g. FBBox, FBConstraint, FBDevice, FBShader, etc.
- In some functions or lists where the object returned is of a more general class, you can cast the item to an object of a more specialized class to access functions specific to that specialized class. In Python, certain functions or lists can automatically return an object of the more specialized type, but not all have this ability.

Advantages of Python

- Minimal resources needed to start developing. No need to obtain a compiler like Visual Studio (which can cost money and time to learn using it). All you need to do is use the Python Editor that is built into Autodesk MotionBuilder.
- No need to close and restart Autodesk MotionBuilder to fix code. C++ plug-ins is automatically loaded on start-up and cannot be unloaded so that its code can be edited. As a result, if functionality is available, then it is generally better to prototype in Python, only switching to C++ when speed is essential and the code is more or less locked down.
- No need to figure out if creating a certain object of an Autodesk MotionBuilder class can be done normally or as a pointer.

1.4 How do they both fit in Autodesk MotionBuilder

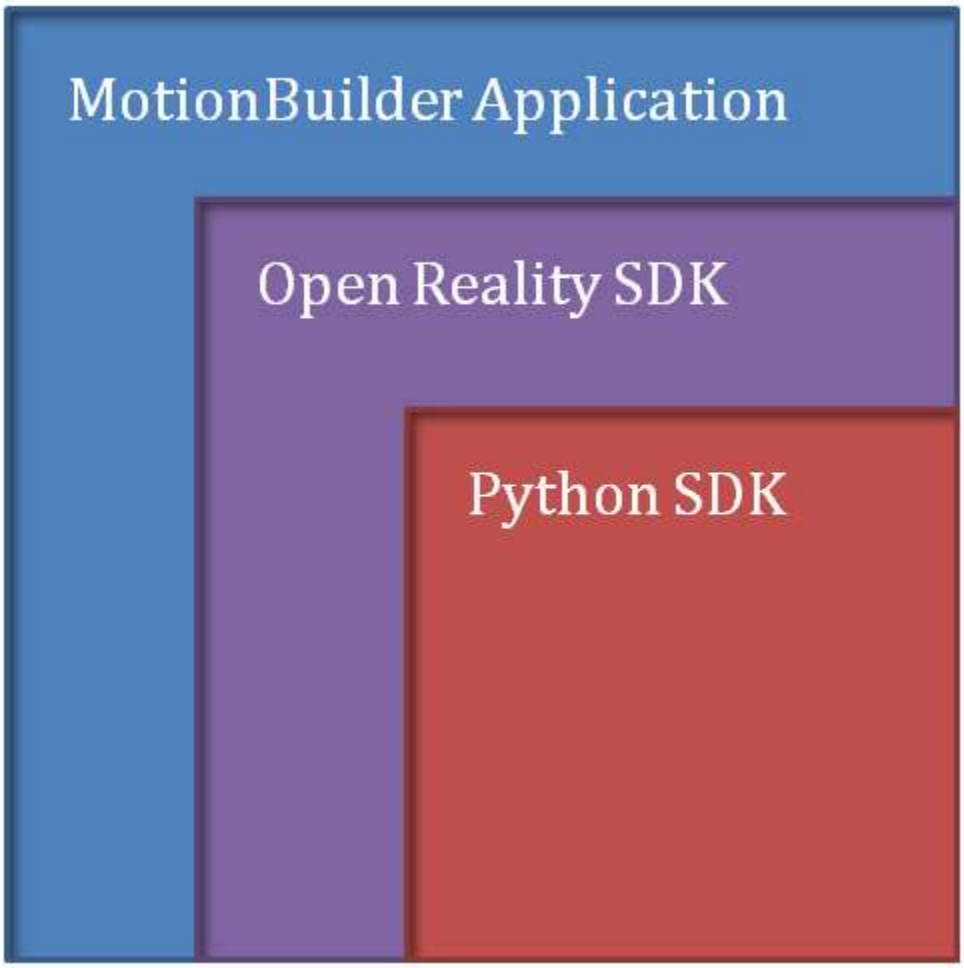

Figure 1 Displays the SDK functionality size compared to Autodesk MotionBuilder

The way that Python fits into Autodesk MotionBuilder is it has a little bit smaller functionality then the OpenReality SDK, and then the OpenReality SDK has a little bit smaller functionality then what the Autodesk MotionBuilder Application can do. So in other words everything Python does, the Open Reality SDK also does (Python is a wrapper to the C++ classes) but everything Open Reality SDK does Python SDK does not necessarily do.

There are a few key classes that are not in Python such as creating devices or manipulators that won't likely get exposed in Python because Python is not optimized enough to perform at the real time speed Autodesk MotionBuilder needs for these tools, but you never know what can happen so don't hold me to this.

One more thing to add here, is that the beauty of Autodesk MotionBuilder or the opposite of beauty depending on what side of the fence you sit on is that the functionality in Open Reality SDK and Python closely mimics the functionality available in the UI, so when you are not sure of

something or what a word means you can always look it up in the User Help Documentation and see the workflow or the definitions, and 9 times out of 10 figure out how to do it in Python or OpenReality (if it is exposed).

Unlike other 3D applications that you might be familiar with, Open Reality and Python generally have the same functionality, so it really comes down to the points listed in the advantages, disadvantages and your language skill for you to choose which SDK you would like to use.