

Python Scripting and Game Development in Blender

Matthew Momjian
OSCON 2011

Why Python in Blender?

Blender is the leading open-source program for 3D content creation

3D games are a natural extension

Python, with its modular extensible design and low barrier to entry, allows for scriptable control of game elements

“Games”

Not AAA titles - not really even games

Long-running (e.g., > 1 frame) 3D outputs

Can interface with user (e.g., moving an object)

Or can be an independent 3D display based on some form of data input

“Games” in Blender

Blender provides the ease of block-based visual programming, with the flexibility of Python scripts

Three main types of “blocks” that can produce almost any result

Sensors

Blender's game engine is based on a constant-loop design

The engine loops over a set of “sensors” which activate components of the game

Sensors can be based on keyboard events, or can be time-based (e.g., once per second)

A pulse from a sensor activates all controllers that the sensor is connected to

Controllers

Controllers are activated by sensors

They come in various varieties of logic gates

- And, Or, Nand, Nor, Xor, Xnor
- An expression combining sensor values and game properties
- Or, a Python script

Actuators

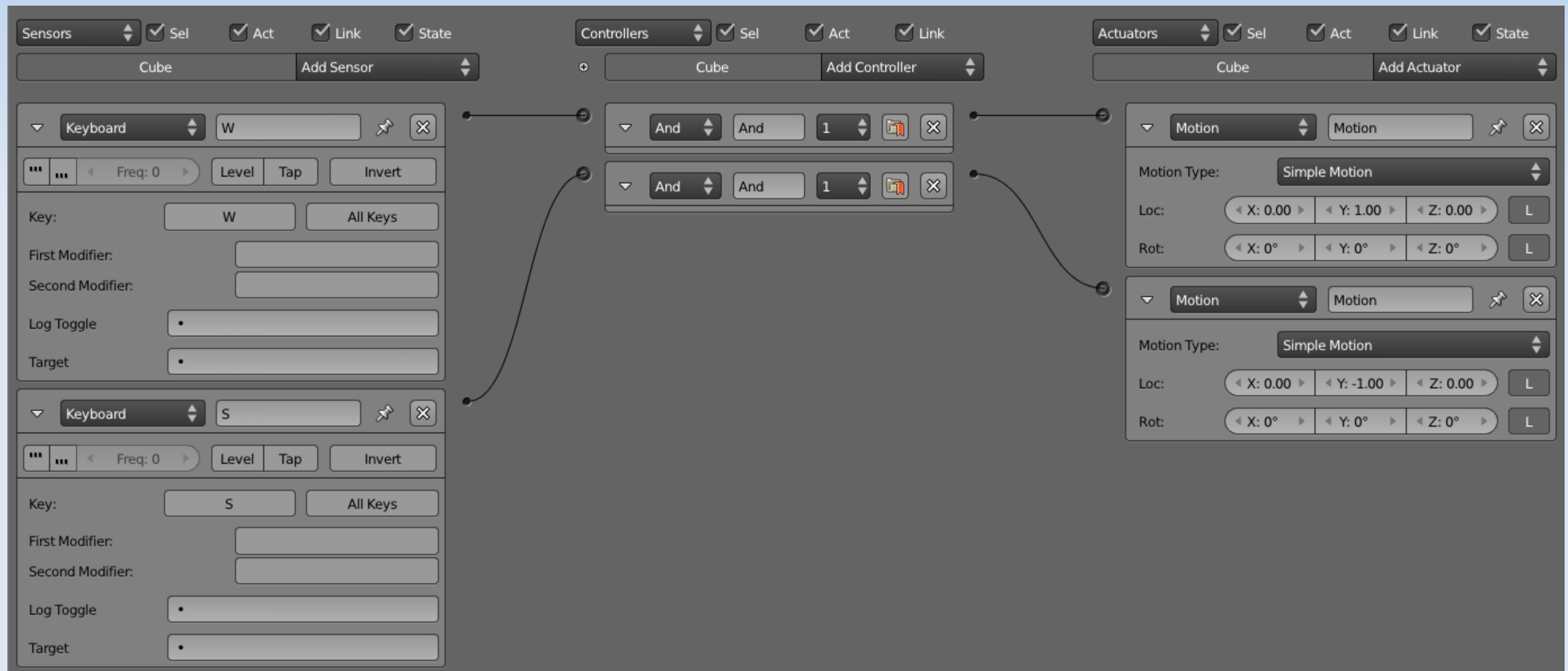
- Produce visible ingame results
- Can move objects, write text to the screen, make sounds, or modify objects
- Often have preset values that are applied when activated
- But can also be manually setup through a script before activation

Game Logic Setup

Sensors

Controllers

Actuators



Blender Python Integration

Blender ships with fully-functional Python 3.2

Most of the Blender game functionality is contained in the module *bge*

```
import bge
```

bge provides sub-modules which allow Python to access and modify object properties

Accessing Scene Data

```
import bge

object = bge.logic.getCurrentController().owner

print(object.position)
> Vector((0.0, 0.0, 0.0)) #(X, Y, Z)
```

Modifying Scene Data

```
import bge
object = bge.logic.getCurrentController().owner
print(object.position)
> Vector((0.0, 0.0, 0.0)) #(X, Y, Z)

object.applyLocation((1.0, 0.0, 0.0))

print(object.position)
> Vector((1.0, 0.0, 0.0))
```

Persistent Data

- Scripts run many times per second
 - Therefore no persistent variable state
 - Blender provides each object with properties that can be written to and accessed between script instances

```
import bge
object = bge.logic.getCurrentController().owner
if object['first_run'] == True:
    #Do init scripts
    object['first_run'] = False
else:
    #Do normal run script
```

More Python

Simple 3D movement attributes can be combined with more advanced Python scripts to produce complex results

- A clock is physically simple but programmatically complex

Compiling

Blender is capable of producing standalone executable 3D binaries that can run on all major platforms (Windows, Mac, Linux)

Thanks!

The Blender Python Reference provides a list of all functions and properties for bpy, bge, and other associated Blender Python modules

http://www.blender.org/documentation/blender_python_api_2_58_release/

- <http://goo.gl/2ENg9>

Questions?

- matthew@momjian.us
- [@mpmmomjian](#)
- [#blender](#) or [#blenderpython](#) on [irc.freenode.net](#)