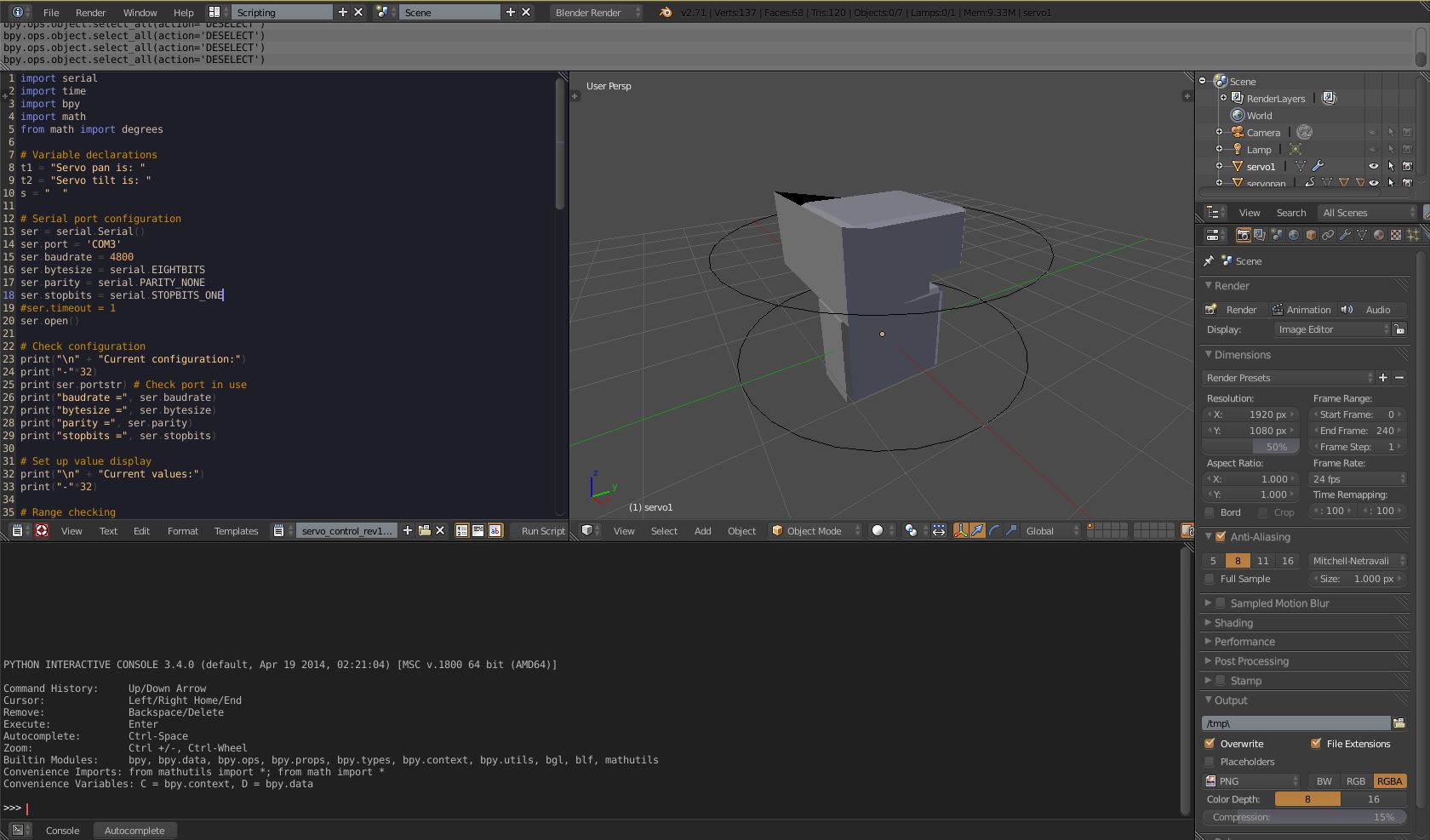
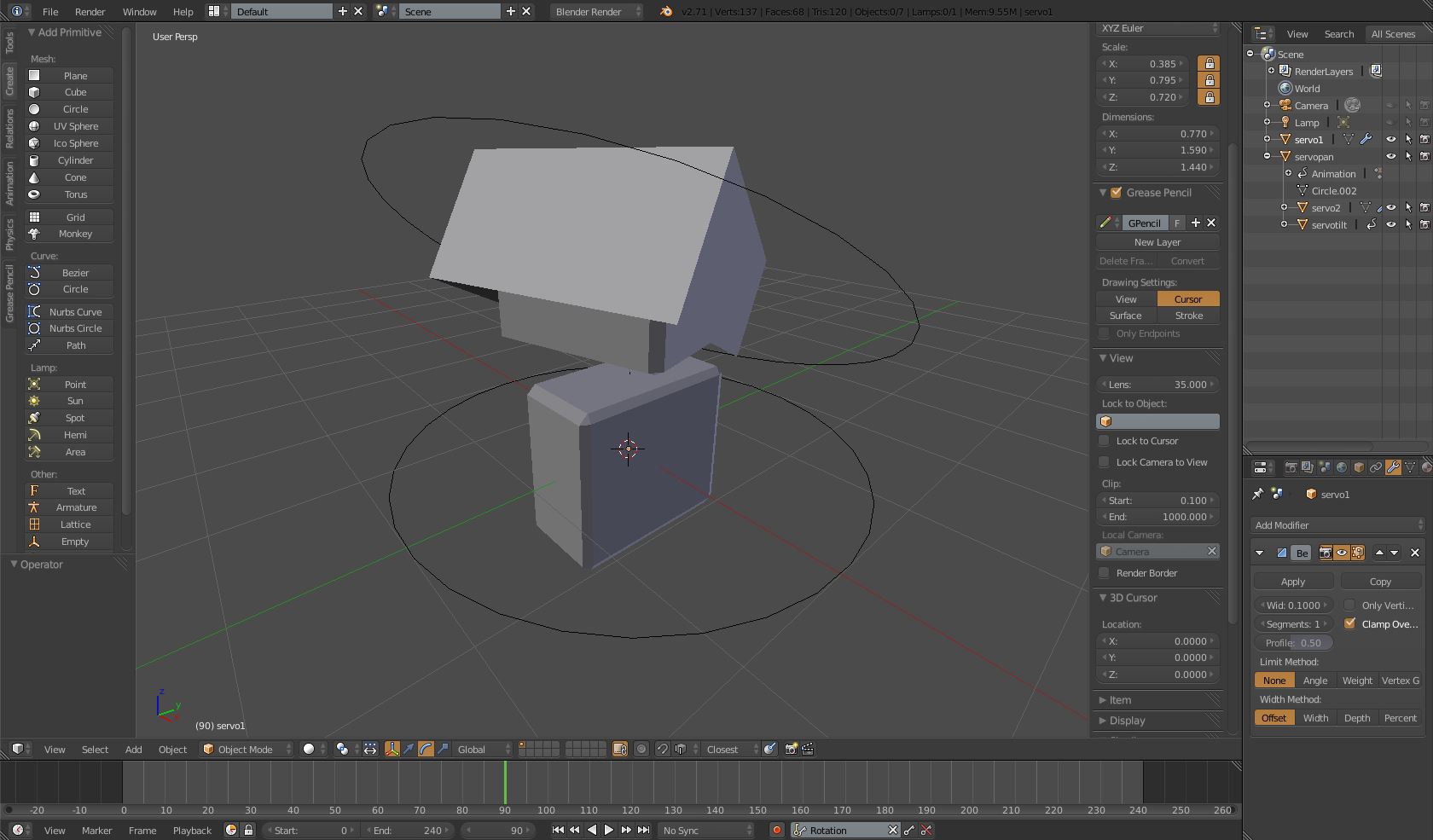
Chad Sebesta

Mechatronics 90

Animatronic Servo Controller

12-10-14





Flowchart for PICAXE program

Start

Initialize variables

Center servos

B

Serial data?

N

Y

Move serial data to variables B1, B2

Values in range?

LED on for 250ms

Y

A

Pause 20ms

A

Pulse values of B1, B2 to pins c.1, c.2

B

Summary

The purpose of this project is to demonstrate that modern modeling and animation software may be used to control mechanical motion. This endeavor is accomplished using an rs232 serial connection to share the state of computer model with of position of two servo motors. The model was created and animated using the free and open source Blender software. A python script allows this software to communicate with a PICAXE microcontroller using a serial connection.

A significant portion of this project involved creating code to allow the animation software and PICAXE microcontroller to communicate. This has never, to my knowledge according to my research, been done on any pic-based or PICAXE microcontroller before, and with very good reason. The PICAXE chip possesses a critical flaw which prevents desirable behavior when communicating with a serial port. The timer which is typically used to send continuous pulses to the servos with the “servo” and “servopos” commands is interrupted whenever the chip receives serial input. During testing, I discovered that using these commands produce erratic and uncontrollable results. To work around this issue, I chose to instead use the “pulsout” command, which only sends one pulse per execution. The program is structured so that this command executes within the servo’s response limit. However, both the python and PICAXE programs contain delays, to allow each device time to complete their respective actions, resulting in a pulse repetition of 40-50ms. This delay may be mitigated by either a finer tuning of the code or the replacement of the PICAXE chip with a more appropriate microcontroller.

I am most proud of the code I wrote. Even though similar projects have existed in the past, I could find very little in the way of examples to help me. All previous examples, of the python portion, are written in python version 2. The Blender animation software uses version 3, which meant I had to determine a way to implement these Ideas in that version. This took me the most time, going through 18 distinct revisions before being satisfied. Along the way, I attempted to focus on readability as a core aspect. The critical block of code was originally written as:

|  |
| --- |
| def my\_handler(scene):  ServoAngle = degrees(bpy.data.objects['Cube'].rotation\_euler.x)  data = int(remap(ServoAngle, -90, 90, 75, 225))  ser.write(bytes(str(chr(data)), encoding='latin-1'))  #ser.write((bytes(str(data), 'ascii')))  time.sleep(.1)  print(data) |

While this is functionally operable, it is unreadable. It took me many more revisions to make it more readable and precise, as I aspire the finished code to be.

Regarding board design, I wanted the control board to be small. I like the elegance of compact and efficient electrical design (although I make no claim to being able to produce elegant things). I initially underestimated the difficulty of soldering the board without shorting various contacts. Many elements are very close together, less than the width of an iron tip, and prone to damage without careful work. I made it work, but I would allow myself room in future builds, if only to reduce the risk of electrical damage to any of the more sensitive components.

If presented with the opportunity to repeat or further enhance the project, there are a number of things I would change. Among them, I would attempt to fabricate all servo brackets myself, which would cut out a significant amount of cost (the brackets and servos were the costliest parts). As well, I would spend time making the polishing the aluminum and staining the wood to add a more finished quality to the project. The most drastic change I would make is the use of a different microcontroller, which would hopefully allow a smoother and more reliable motion of the servos motors. This may also allow the implementation of a more common USB connection, without the use of an adaptor.