October

Hello Benefactors. Here's what I've been working on in October.

This month has been all about getting the Virt-a-Mate random stroker plugin working, so that is going to be what I talk about this month. It may not have shown so much on the devstream, but I have put a lot of work and thought into that plugin over the last year or so.

I have also been asked a few questions about inverse kinematics on the SR6, and I was going to include some stuff on that in this month's dispatches, however I've run out of time so I will be posting that in the devstream over the next few days and I will include it in the next dispatches.

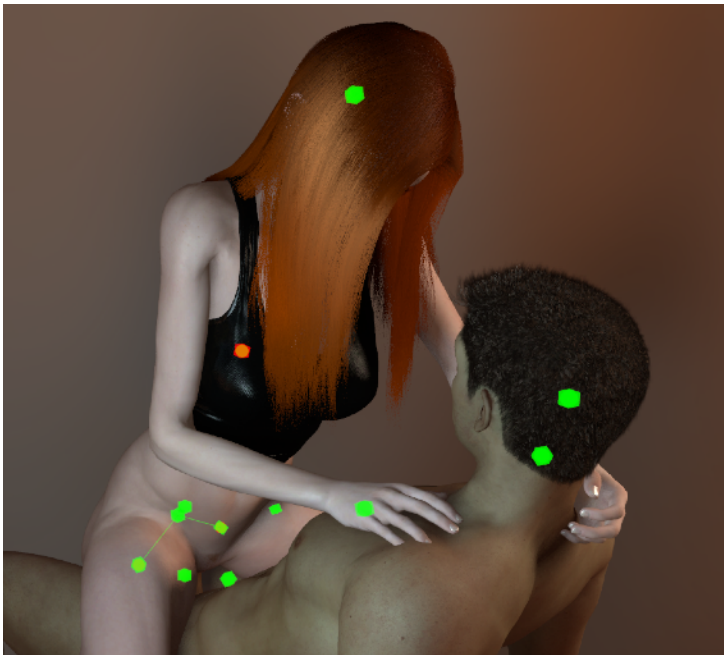
I should also say that we're now three quarters of the way through the year and I've already written over 21,000 words on 58 pages! If only I had been able to churn out that kind of word count on the novel I was working on! Oh well, maybe that's why I'm a designer and not a novelist.

Random Stroker

For the last three months I have been slowly working on a plugin for Virt-a-Mate, which this month I have finally been able to accelerate and release to my user base. This is my second attempt at a Random Stroker plugin.

To tell you the story of the random stroker first of all I have to define what I mean when I talk about a random stroker.

One of the reasons that my devices seem to be as popular as they are is that there's a general recognition that sex isn't something that happens in one axis. If we're just talking about physical movements between two bodies during sex in reality there are a wide range of movements that take place. The massive advantage that a device like SR6 has over commercial stroker devices like the Fleshlight Launch, Handy, Kiiroo Keon, etc, is that it can emulate these kinds of movements.

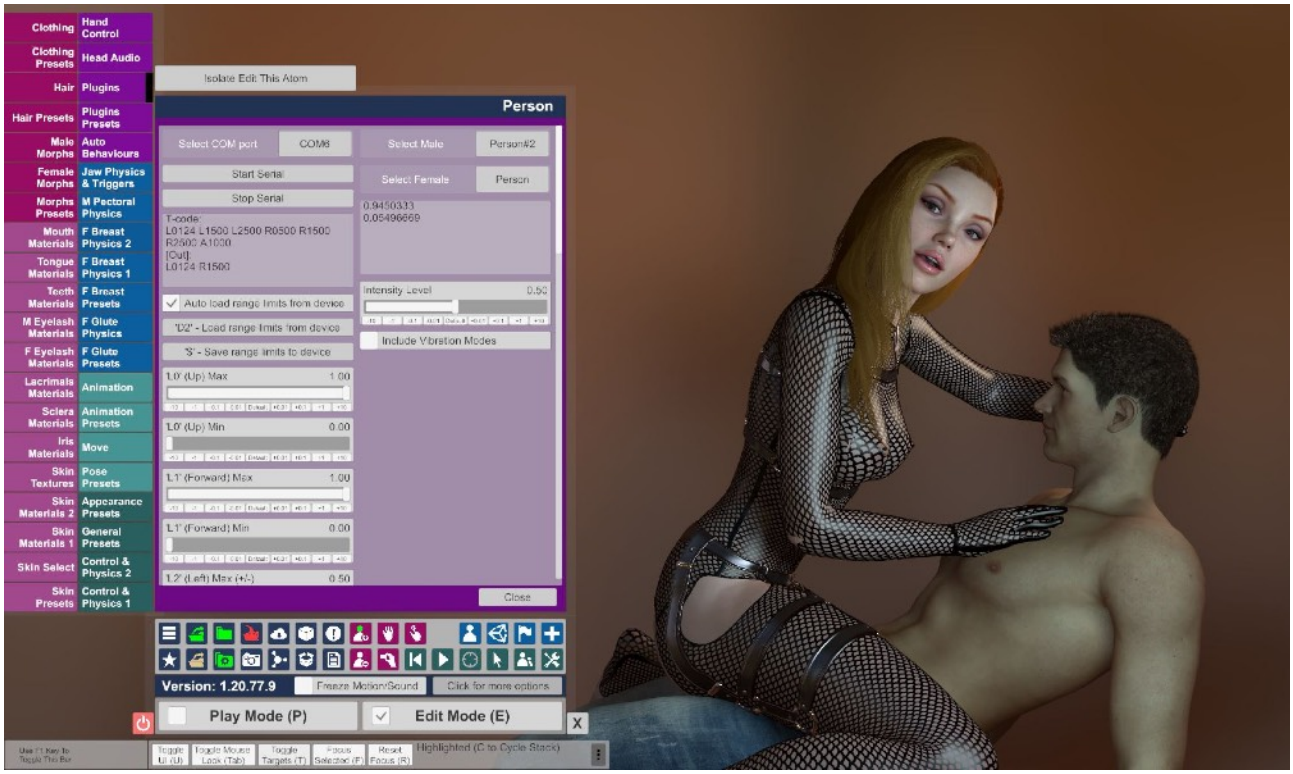


JULY '22: EARLY ATTEMPTS AT ANIMATING THE STROKER ALGORITHM. NOTE THAT IN THIS EARLY ITERATION THE HIP NODES ARE LINKED TO THE PELVIS

The massive disadvantage is of course that there is limited content out there that can drive this. It's the massive chicken-egg question that I have had to deal with ever since I designed my first multi-axis device, which is: who would want a device there's no content for; who would make content for a device that doesn't exist. Multi-axis scripting takes a lot of work, especially if it's the full 5 or 6 degrees of freedom for an SR6.

One antidote to this is of course Virt-a-Mate. By nature of being a computer simulation VaM already has all of the tracking data and all that's needed is to be able to output it in the form of T-code commands that the robot will recognise. Obviously there are several plugins now that can do that, including my own, and this works well. The problem is that there are a limited number of suitable scenes available, and each scene is of a fixed duration and will play out in the same way, stroke by stroke, each time.

What we want, therefore, is a scene where the user can experience a wide range of movements from the stroker robot in a way that can continue indefinitely, and is different each time user experiences it. This kind of unpredictability in the movement is definitely desirable. We also want some kind of structure to this experience, so that the movements start of slow and teasing, and over time become faster and more insistent. What we mean then, when we say “random stroker”, is not actually random, but essentially a procedurally generated progressive multi-axis stroking algorithm.



EARLY ITERATION OF THE V2 PLUGIN. THE “VIBRATION MODES” CHECKBOX IS LEGACY CODE THAT I HAVE SINCE REMOVED

As I’ve said before, it’s one thing deciding that you want to write some code to drive a stroker robot, but the actual practice of writing the code is initially very daunting. For the commercial strokers, their approach to free play is literally to move the receiver from A to B at a constant speed, then drive it back at constant speed. If the user is lucky the device maker will give them a couple of buttons to change speed and maybe even the stroke length. They don’t even begin to contemplate any kind of randomisation to make things more interesting for the user.

It’s not an easy task: things are bad enough when there’s only one axis to consider. Adding in extra degrees of freedom only adds to the scale of how intimidating the whole thing feels. For my first random stroker I used a sine wave for the thrusts on the main axis, and then added in some other fairly complicated code to add in movements to compliment it in pitch and roll. By the time I had come up with the SR6 I realised that what I needed was actually to simplify everything to a generic system of parameterised cyclic movements on parallel axes. That way interesting movements could be specified by the user in the form of parameters - numbers entered into and stored in a matrix of simple numbers - rather than being hard coded.

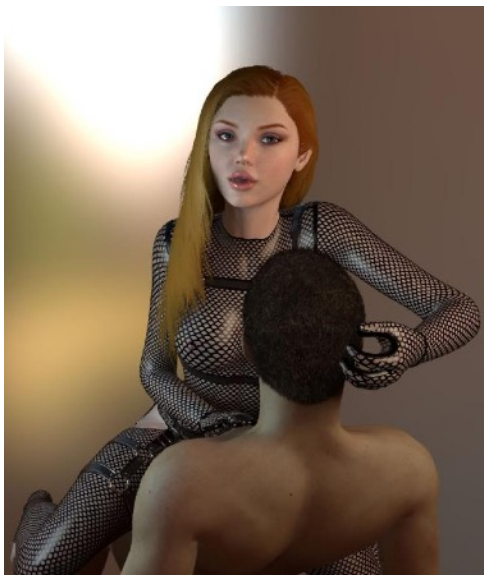
The particular stroking algorithm that I have implemented in this latest plugin actually already has quite a bit of history. I first showcased it to my subscribers back in March last year in one of my videos but I never found the time to develop it. I actually gave the algorithm to soriteparadox and he has done some really impressive stuff with it on his Ayva project recently. You can check out my interview with him back in August if you want to know more about that. I have also written about the algorithm in these dispatches over the last few months of course.

Having the algorithm is one thing, but implementing it as a Virt-a-Mate plugin requires a much broader implementation.

The starting point for my plugin was the same code that already existed in my Serial Controller plugin. This already included all of the code I needed to establish the serial connection with the device, send and receive stored axis preferences, and so on. It was a relatively simple matter to delete the tracking code and replace it with the basic stroker code. Not quite cut and paste, because the algorithm was originally written in Python and VaM plugins are in C#, but it was mostly a case of changing the syntax from one programming language to another.

Voila! A random stroker algorithm that sends out T-code commands running in a VaM plugin instead of in Python.

The next task was to use the same position commands created for the robot to animate the movement of the female in the scene in VaM. Anybody who has animated a scene in VaM will know that this is awkward at the best of times. The character models in VaM are animated using various nodes that correspond to certain parts of the body, but the connection is like having a spring between body and node, and that means that if a node is in the correct place there's no guarantee that the body will be too, which can make precise animation problematic.



REN'S JESSICA CHARACTER IS STILL ONE OF MY FAVOURITE VAM MODELS. MAKING A PLUGIN THAT WILL WORK FOR ANY MODEL ADDS TO THE PROGRAMMING CHALLENGE

The best solution to the animation problem, which I used in this implementation, is to animate using multiple atoms, specifically the pelvis and both thighs. You can thank issacnewtongue for that little insight, as he used a similar setup in his old lap dance scenes. What he did was gang together the two thighs by linking them to the pelvis or hips as a parent atom, then just animating the parent. This process of controlling by multiple atoms gives much more control authority over the female's body to the animator. The approach I used doesn't require linking the atoms in VaM, rather I just get the plugin to control the position of all three atoms.

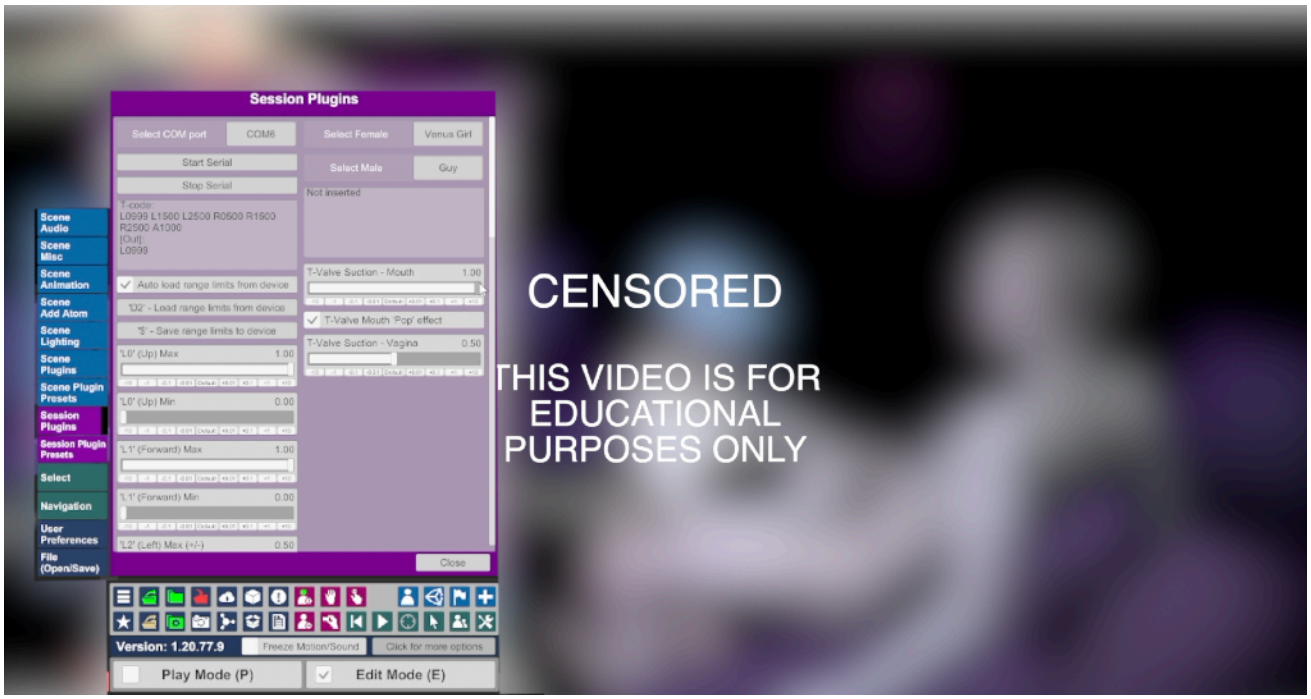
Having established that I would be animating the female's position using nodes I had to come up with a way of determining how to position those nodes. I needed some kind of coordinate system, so what I used was to use two atoms associated with the male character's penis: the base and the tip. Not only did this give me the position and direction of the main axis, but it also defines the full stroke length. Next, to complete the coordinate system I used the male's head, his actual head, in relation to this penis line to define the other directions: forward, back, left, right, etc.

The positions of the pelvis and thigh atoms are then assumed to be at certain distances above and to either side to the female character's vulva, so by placing them at those relative positions it is possible to position the female model onto a position someone along the length of the male model's penis. It is also possible to rotate or translate those nodes around that position, to reflect movement of the woman's hips in multiple axes.

Setting the parameters for this process required a lot of trial and error. It was important to get the right amount of translation or rotation to get a good visual effect, but not constantly have the penis pop out. Not a problem entirely confined to VaM of course, but it can definitely break the immersion for the user and so best avoided if possible. I added a couple of position modifier sliders to the interface in case the user found they had to do any fine tuning.

Having converged on the right set of animation parameters using different kinds of strokes I then had to consider the transition between different strokes. This isn't always straightforward because the next stroke will not necessarily begin at the same location that the previous one ended. What I had to do was introduce a transition stroke, during which the various stroke parameters, and also the stroke speed, would ramp from the previous type of stroke to the new stroke. This made the shift from one stroke type to another pretty seamless.

Having added this functionality I took it a stage further. One of the problems with automatic strokers is that they can feel very mechanical, and a major reason for this is that the movements are too perfectly timed and each stroke is identical. What I did therefore was add an element of randomisation to each stroke. So each stroke is from a set is based on the same set of parameters, but the movements are randomised, as is the speed. This is controllable by a slider with the default being +/- 10%.



AS SEEN ON YOUTUBE. THE LIVESTREAM TO DEMO THE PLUGIN WENT VERY WELL, BUT THERE WAS A LOT OF IT I THAT COULDN'T PUBLISH AFTERWARD!

As well as this low-level control, I wanted to do something interesting for the overall progression of the scene. Conceptually the way I thought a scene should play out is that to begin with the movements would be slower, and would involve a lot more teasing motions. Over time, these would give way to much faster, mostly stroking movements.

What I did then was create four categories of movement: long stroke, short stroke, tease and grind. I populated each with similar but different movements of that type. Long strokes would be 70%-100% of a full length up-down stroke, at a medium speed with various interesting rotations. Short strokes would 30%-50% of a full length up-down stroke but move a lot faster. Teasing movements would be movements that dwelled at the top of the stroke length and made good use of the SR6's rotational abilities. Grinding movements would be movements that dwelled at the bottom of the stroke length, and would make good use of the SR6's translational ability.

I considered making use of extensive flow diagrams to control how the plugin moved from one phase of the encounter to another, but in the end I settled with a much simpler system. The stroke type is indirectly determined by a parameter called "intensity", which can be set using a slider in the plugin menu.

The intensity influences the base speed that is chosen randomly by the plugin for each set of new strokes: the higher the intensity, the higher the speed is likely to be. The intensity level also affects the distribution of probabilities for what kind of stroke will be chosen for the next set.

At "0" intensity the probabilities are: 30% tease, 10% grind, 50% long strokes, 10% short strokes. At "1" intensity the probabilities are: 10% tease, 30% grind, 30% long strokes, 30% short strokes.

These probabilities can be further biased by changing sliders in the plugin menu. So for example if a user doesn't want tease motions, or wants more of them, this can be changed.

I started out using the original movements from my random stroker python script, which are the ones currently featured on Ayva, but I quickly decided that I needed to come up with some new ones. What I wanted was six or so different movements that were distinct to the category that they were in.

The stroke types are currently hard coded into the plugin. One definite area of improvement for the plugin in future would be a way for the player to install custom strokes into each category.

Having got the plugin working, complete with its various stroke types, I set about bundling the scene up so that I could release it. I made a point of using a mostly empty scene that only uses native VaM resources. I think VaM is amazing but my least favourite part of it is opening a scene and discovering that the creator has used a list of obscure dependencies a mile long, which I will have to go and trawl the internet to find.

My expectation with this plugin is that people will load their personal favourite characters into my scene, or they will use the plugin in a scene of their own. Keep things simple.

That said I hope people find the plugin, installed where it is on a sphere on the sofa next to the happy couple.

I'm actually very pleased with how the plugin turned out. It would have been very easy for it to have become quite a monster, but through a lot of careful thought I have been able to keep it relatively simple, but effective.

```
// Random stroker movement modes
// Short strokes
private float[][,] ShortStrokes = new float[6][,]
{
    new float[,] { // low roll forward
        { 0.0f, 0.5f, 0f, 0.25f },
        { 0.5f, 0.5f, 1f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.4f, 0.6f, 1f, 0f }
    },
    new float[,] { // low roll forward
        { 0.0f, 0.5f, 0f, 0.25f },
        { 0.5f, 0.5f, 1f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.4f, 0.4f, 1f, 0f }
    },
    new float[,] { // mid roll forward
        { 0.25f, 0.75f, 0f, 0.25f },
        { 0.5f, 0.5f, 1f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.4f, 0.6f, 1f, 0f }
    },
    new float[,] { // mid roll forward
        { 0.25f, 0.75f, 0f, 0.25f },
        { 0.5f, 0.5f, 1f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.4f, 0.6f, 1f, 0f }
    },
    new float[,] { // high roll backward
        { 0.5f, 1.0f, 0f, 0.25f },
        { 0.5f, 0.5f, 1f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.4f, 0.6f, 1f, 0f }
    },
    new float[,] { // high roll backward
        { 0.5f, 1.0f, 0f, 0.25f },
        { 0.5f, 0.5f, 1f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.5f, 0.5f, 0f, 0f },
        { 0.4f, 0.4f, 1f, 0f }
    }
};
```

THE SHORT STROKES USED BY THE PLUGIN, AS WRITTEN INTO THE CODE

THE BLANK SCENE CONTAINING THE PLUGIN. DEFAULT FEMALE A HAVING FUN WITH DEFAULT MALE A

I do have in mind several improvements that I could make to this plugin.

The first of course is some kind of functionality for adding or removing stroke types. This would probably be based on .json files, and compatible with the export/import function on Ayva.

Another function that I want to incorporate is some kind of edging button. At the present time the way that a user can avoid getting too much stimulation and ending the session is to use the power switch on the SR6 to stop the machine. What I would like to do is to introduce some kind of button on the SR6 that the user can press, which will communicate via serial with the plugin and cause the random stroker to stop all motion for a period of time, then resume things slowly.

This would also potentially be a way to integrate something like the nogasm/protogasm into the experience. Essentially it could be rigged up to be able to press the button for the user at the appropriate time. I really need to find some more time to look at that device though.

I am also thinking about how to add in vibration and maybe squeezing too, once I have introduced a squeezing function to my devices.

Finally...

I'd like to say a big thanks to you guys for the support you give me. I'm actually enjoying sharing this glimpse into my creative process with you, so I hope that it's interesting to see and read. I would like to invite you guys to engage more on the discord server, especially if you see something I've posted and you have any questions or suggestions. I'm also definitely open to organising livestreams and hangouts on there too if there's anything particular you'd like to see more of. Let me know.

Thanks again!
Tempest