The All-New 2015 VP Plunger

This guide explains how to use the new features I recently added to VP's built-in plunger. As of this writing, these features are experimental and not part of any official VP release, but the intention is that they will be soon, if that the community feels they're an improvement worth keeping.

The current release includes testing versions of VP 9.9.1, Physmod 5, and the 10.0 Beta release. The testing releases are based on up-to-date snapshots of the official source code, so they include all original features in addition to the new plunger features. The testing versions shouldn't change anything else in the way VP runs your existing tables, so you should be able to use them as drop-in replacements for the respective official releases.

The image at right is a sample of what the new visual features can do. This is a VP rendering of the new "Custom" plunger type, which is a new 3D model plunger that gives you considerable control over the shape and appearance. A nice bonus is that when you pull back this plunger in VP, the spring compresses just like the real thing.

Please post your feedback in my vpforums.org New Plunger Features thread (I'm "mjr" on vpforums). I'm interested in general opinions on how well the new features work, as well as bug reports (on both the software and any errors you find in this write-up).

How to install the testing version

You'll need to install the official VP before you can install the plunger-mod versions, because the test release only contains the modified VP executables. The regular VP installer includes supporting files that you'll also need; these are unchanged in the test version, so I didn't include them in the ZIP file. One you have the official VP version(s) installed, simply extract the corresponding .exe files from the ZIP file and place them in the corresponding VP install directories. The test exes have different names to help you distinguish them from the originals. You can keep both the modified and original versions installed on your system, and freely switch back and forth at any time by running one .exe or the other.

If you have any special performance tuning in place for vpinball.exe (such as process CPU affinity settings), remember to apply the same settings to the testing versions.

New visual features

There are two new visual styles available for the plunger, along with several new properties. Some of the new properties apply specifically to the new visual styles, and a couple of them apply to all styles.

The new Flat style (select PlungerTypeFlat in the Type box) renders the plunger as an image (which can have alpha transparency) drawn onto a rectangular, horizontal surface. In its simplest form, this simply draws the image, moving the image as the plunger moves. More interestingly, you can do custom animation by providing an image file that contains multiple cells. VP will render the appropriate cell according to how far back the plunger is pulled. This lets you animate spring compression or shadow effects, for example.

The Custom style (select PlungerTypeCustom) is essentially a customizable version of the PlungerTypeModern style from past versions. Like the Modern style, the Custom plunger uses a 3D model with texture mapping to generate the visuals. This has the advantage over the Flat style that it automatically adjusts to perspective shifts, so it will still look right (or has a better chance of looking right, at least) if you change the layback settings or switch between desktop and cabinet layouts. What the Custom style has that the Modern style doesn't is a big set of properties to control the shape of the 3D object. It also has a fully animated 3D spring that compresses and expands with the plunger – and it's an actual 3D model of a spring, not just a texture mapped onto a cylinder.

In addition to the new styles (and some new properties that apply specifically to each of the new styles), there are two new properties that apply to all styles. First, Width lets you control the drawing width of the object. This was hard-coded in past versions, which usually made it impossible to match the aspect ratio needed for a photo-realistic re-creation of a real table. Now you can control the width to get the right appearance. Second, Z Adjustment lets you change the vertical position of the plunger rendering. This is especially useful for the flat plunger, where its flatness can create the appearance of odd alignment discrepancies.

Width property

This property lets you set the width of the drawing area of the plunger. Note that it acts like a radius: it sets the horizontal extent on *each side* of the plunger's centerline (given by the X position property). So adding 1 to the Width value makes the rendering area 2 units wider (in table distance units).

For the Flat style, the Width value sets the exact width of the rendering surface. The image you supply will be stretched or compressed to exactly fill this area, so you can use this to adjust the aspect ratio of your image.

For the 3D-modeled plungers, the model doesn't necessarily fill the nominal width exactly. Instead, the models use the Width value as the basis for the the radii of their parts. Everything is proportional to the Width value, so doubling the Width will double the radii of all of the individual parts.   The Width also serves as the Z height (above the playfield) of the center axis of the plunger rod in all of the 3D models.

Z Adjustment property

This property moves the rendering up or down vertically. Setting this to 0 gives you the default height. Setting a positive value moves the rendering up from the default position by the given distance, in table distance units. A negative value moves the rendering downwards.

For the 3D styles (Original, Modern, Custom), the default is to place the center axis of the plunger rod above the playfield by the Width distance. If for some reason this doesn't produce the right perspective on your table, you can change the Z Adjust value – a positive value will move the plunger upwards further above the playfield, a negative value will move it downwards.

For the Flat style, the default height of the rendering surface is 1.25 times the Width value. The 3D models have their center axis at 1.0 times the Width value, but they're cylindrical so they extend above (and below) this axis. This means that the visible top surface of a 3D plunger is actually a bit higher than the Width. The 1.25 multiplier is meant to approximate the height of the top surface of a 3D plunger, to provide roughly the same perspective effect. But this doesn't always look right, which is why I added the Z Adjustment property in the first place. If the perspective comes out wrong with the default settings, you can tweak it with this property.

PlungerTypeFlat

The Flat plunger draws a custom image onto a horizontal, rectangular surface.

Image: Select an image using the Image property as you would for any other object. VP will render this image onto the surface area you specify. You can set the length of the drawing area via the Stroke Length property, and the width via the Width property. Note that the Width property acts like a radius – this is how far the plunger extends on each side of its centerline, so the actual rendering area width is twice this value.

Vertical position: VP uses a 3D model, so the rendering surface has a vertical position. This affects its apparent position relative to nearby objects (like the ball). By default, the vertical position is 1.25 times the Width property (in table distance units) above the playfield (or above the surface the plunger object is attached to via its Surface property, in the Position panel). You can use the Z Adjustment property to move this up or down as needed; a positive Z Adjustment value moves it up by that many table distance units, and a negative value moves it down. The 1.25x Width default is meant to make the perspective look about right in most cases, but it might look off in some cases, depending on how you have the camera angle set (via the backdrop formatting settings). If it does look misaligned, you can tweak it to get the exact effect you want via the Z Adjustment.

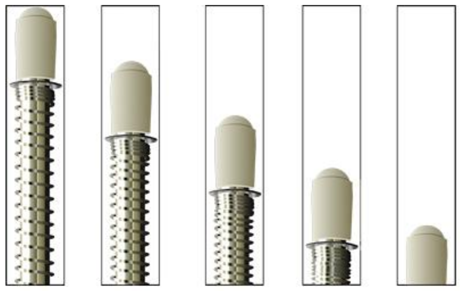
The reason that the rendering surface isn't just drawn at 0 height, right onto the playfield, is that this can create the odd appearance that the ball is rolling over the top of the plunger. The default 1.25x Width height places it just above the center of the ball, which is closer to where the top surface of the plunger would really be.

Cell-based animation: By default, VP will just draw the static image you supply. The plunger will be "animated" in the most basic possible way, in that VP will move the drawing surface back and forth as the plunger moves.

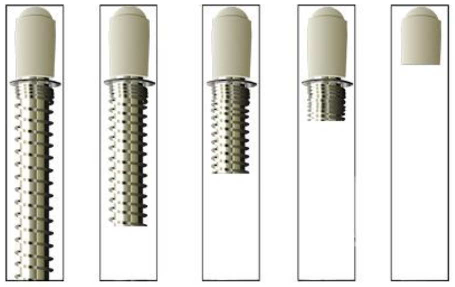
If you want to do more realistic animation, you can supply an image made up of multiple animation cells rather than just one static image. To do this, create your animation image file by lining up all of the animation cells side by side across the width of the image. Every cell must be the same width, and the top of the plunger image must be at the same position in every cell. The leftmost frame depicts the fully forward position, and the rightmost frame depicts the fully retracted position. Set the Flat Frames property to the number of cells making up the image.

You can create any number of frames you want. VP will interpolate the frames across the travel distance. If you create an image with 4 frames, VP will display the first image in the upper 25% of the travel range, the second image in the next 25%, and so on.

As an example, suppose we're starting with these images, showing a plunger in various stages of spring compression. (In practice you'd probably want to use more than just 5 images; this is just to keep things simple.)



We need to combine these into a single image file (a PNG file, say). They're already in the right order: the most forward position goes in the leftmost cell, and the most retracted position goes in the rightmost cell. However, the vertical alignment is wrong. In the combined image, we need the **top** of the plunger to be aligned at the same point in every cell. So we need to tweak the images like this:



Now we're ready to combine these into a single file:

To use this animation, import the image file into VP, using the Table | Image Manager dialog. Set the plunger's Image property to the imported image, and set Flat Frames to 5.

(These images, by the way, are all in the release package. The full set has 25 frames for smoother animation.)

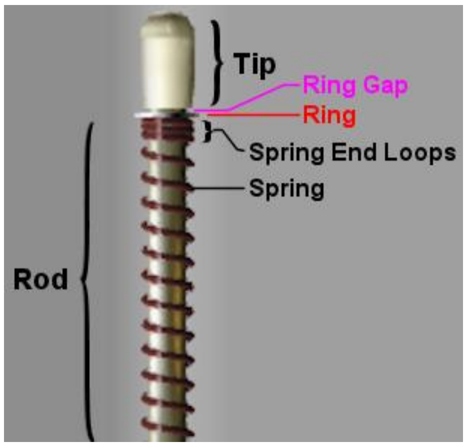
PlungerTypeCustom

The Custom plunger is similar to the Modern plunger from previous versions, but gives you much more control over the exact visual appearance. Like the Modern plunger, the Custom style uses a full 3D model to generate the rendering, so it automatically adjusts to the correct perspective for the table.

The new properties in the Custom Settings box (in the Colors & Formatting panel in the plunger property sheet) let you control the visual appearance of the plunger. The Custom plunger isn't a fully general 3D object; it's still limited to the basic shape and components of a standard pinball plunger. But assuming you don't want something really unusual, this should be flexible enough to draw just about any real pinball plunger. For unique controls that don't fit the standard model, you can always use the Flat style and a fully custom image (although you do lose the 3D-ness if you go that route.)

The Custom style is drawn slightly longer than the Modern style, to make room for the spring. For all plunger types, the Stroke Length property sets the distance that the plunger can travel. The visual length depends on the Stroke Length but is slightly longer; exactly how much longer depends on the plunger type. For most types, it's about 20 table distance units longer. For the Custom style, it's long enough to accommodate the spring in the fully retracted position.

Just so we all agree on terms, here are the fixed components of a plunger in the Custom model:



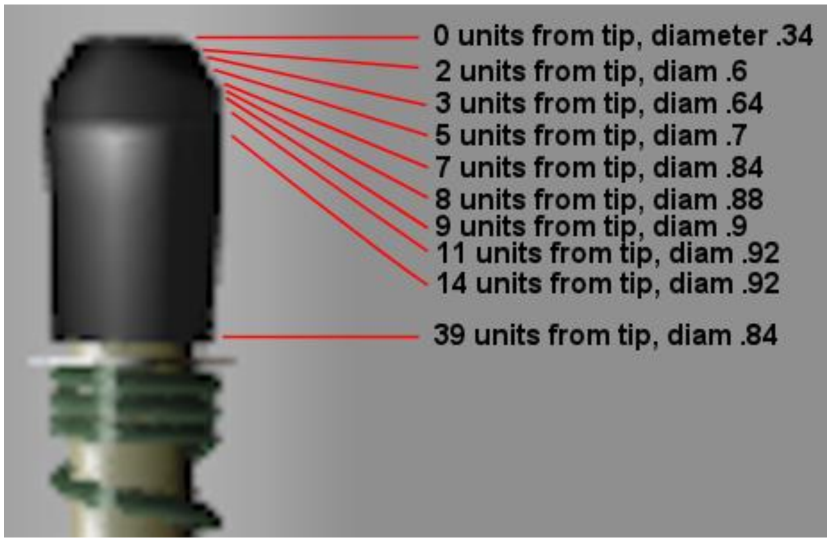
Here are the properties that control the various parts.

Width: This is the base value for all of the "diameter" properties below. The diameter values are multiplied by this value to get the actual size in table distance units. So you can use this to scale the visual size up and down while keeping the same overall shape and proportions.

Tip Shape: This is the most complicated of the custom properties. This is a list of (Distance, Diameter) pairs separated by semicolons. The Distance value in each pair specifies a distance back from the tip of the plunger, in table distance units. The Diameter value gives the diameter of the tip at that point, relative to the Width setting. This will be easiest to explain by way of example. Here's the default Tip Shape that you get when you create a new plunger (in the new version, anyway):

0 .34; 2 .6; 3 .64; 5 .7; 7 .84; 8 .88; 9 .9; 11 .92; 14 .92; 39 .84

And here's how that gets rendered:


0 units from tip, diameter .34
2 units from tip, diam .6
3 units from tip, diam .64
5 units from tip, diam .7
7 units from tip, diam .84
8 units from tip, diam .88
9 units from tip, diam .9
11 units from tip, diam .92
14 units from tip, diam .92
39 units from tip, diam .84

Note that the very top comes out a little flatter than a real pinball plunger, which has more of a dome shape. You could fix this by adding another couple of smaller circles, but I figured this wasn't necessary for the defaults because the perspective settings usually end up hiding this part anyway. Note also that the shading effect isn't coming from any clever 3D processing in VP; it's just from the texture mapped onto the tip.

One important rule about the Tip Shape list: the items have to be in order, from the tip down. This means that each Distance value must be higher than the previous one.

Rod Diameter: Sets the diameter of the rod, relative to the Width setting.

Ring Gap: This is the distance, in table distance units, between the tip and the "ring". (The ring is the part in a real plunger that holds the spring in place – it's a little flat metal disk roughly like a washer.)

Ring Diam: The diameter of the ring, relative to the Width setting.

Ring Width: The thickness of the ring, in table distance units.

Spring Diam: The diameter of the spring coil, relative to the Width setting. This should always be greater than the Rod diameter, so that the spring is rendered fully outside of the rod.

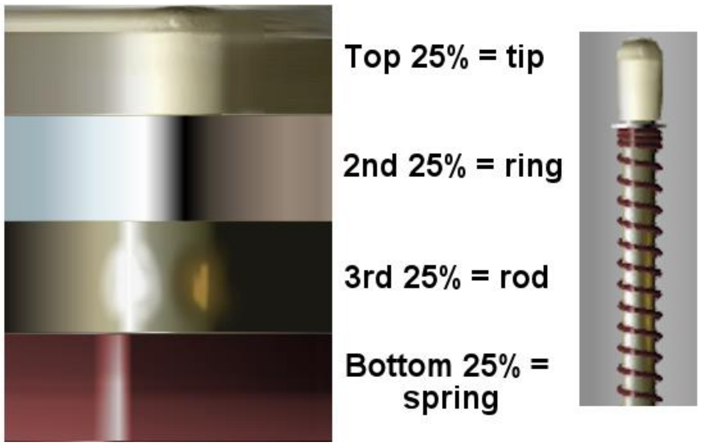
Spring Gauge: The thickness of the spring wire, in table distance units.

Spring Loops: The number of loops the spring makes around the rod within the displayed extent of the rod. This can have a fractional part; e.g., 5.5 gives you 5-and-a-half loops.

End Loops: The number of "end loops" the spring makes. Real springs tend to have a few coils at each end that are tightly wound together, and don't expand with the rest of the spring. This specifies the number of turns to use for those coils at the end. As with Spring Loops, this can have a fractional part. VP starts rendering the spring on the **bottom** side of the plunger, makes the number of turns specified in End Loops, and then continues directly into Spring Loops. If you want the transition from the end loops to the regular loops to start on the bottom of the plunger, use a whole number for End Loops, since that will make the end loops end after a whole number of turns around the rod. If you want the transition to be visible on the top of the plunger, use .5 for the fractional part in End Loops, like 3.5 or 4.5.

Image: For the Custom plunger style, the image isn't just plopped directly on top of the 3D object. Our model has four distinct components, and each component is made of a different material. To achieve a most photo-realistic appearance, we want to map a different texture onto each component. It would be tedious if you had to go through the process of assigning four separate image files to each plunger, so instead, the Custom style uses one image to hold all four textures. It does this by divvying up the one image into four quadrants. Each quadrant is mapped onto one of the components. The quadrants are arranged vertically; for simplicity, each takes up exactly 1/4 of the vertical space of the image. The horizontal extent of the each section is wrapped around the cylinder that forms the corresponding component. The horizontal center of the image is mapped to the middle of the top of the cylinder, and then each side of the image is wrapped around the corresponding side of the cylinder. The outside edges meet at the middle of the bottom, so the seam is conveniently hidden. In the case of the spring, the texture is wrapped around each loop of the spring; for the rod, tip, and ring, the texture is mapped across the vertical extent of that component.

Here's an illustration showing an image file and the rendered plunger it produces. If you study the two images, you should be able to see how details in the texture correspond to details on the rendered plunger using the mapping rules described above.



New physics features

The new plunger includes an overhaul of the plunger physics. The changes should be essentially transparent to table authors, in the sense that you shouldn't have to change any properties to get the effect you were originally aiming for. However, the new physics calculations are different from the old ones, so you might end up wanting to tweak the settings for some tables. If the strength of the plunger is too low or too high, you should adjust the Release Speed property, and possibly the Mech Strength property. In most cases, you can now simply set Release Speed and Mech Strength to the same values to get consistent behavior in both the keyboard and analog plunger device interfaces.

The physics of the plunger is basically a black box, so I'm not going to go into detail about it here. To summarize, though, the changes I made are intended to make the workings of the black box seem more like a real plunger and less like a random number generator. If I was successful, the plunger should now feel natural and consistent and controllable.

There are a few visible changes, though: one new property that gives table designers a new knob for greater control over the plunger physics, one old property that was no longer necessary, and a new bit of automatic behavior for Auto Plunger mode. Details below.

Momentum Xfer property

This is a new property that lets you tweak the strength of the plunger independently of the animation speed. In the past, the strength of the plunger was controlled mainly by the Release Speed property. This is still the case; the basic strength calculation in the new physics is meant to produce roughly the same strength as in past versions, to make sure existing tables work about the same way they used to. But the Release Speed doesn't just go into the strength calculation. It also controls the speed of the on-screen animation for a plunger release. It makes sense for these two properties to be coupled like this, but the coupling creates a practical problem for some table authors: sometimes the value of Release Speed that gives you the release strength you want also happens to make the animation too fast or too slow.

Momentum Xfer is meant to fix this by adding a separate knob that only affects the strength, and has no effect on the animation. This value is simply multiplied into the strength. The default value is 1, since multiplying by 1 just gives you back the same value you started with. If you want to make the plunger twice as strong without changing the animation speed, change Momentum Xfer to 2. It's more likely that you'll want to make the plunger a little weaker; in practice it seems that the right on-screen animation speed produces plungers that are too strong for tables with light-touch skill shots. To make the plunger half as strong, set Momentum Xfer to 0.5.

Auto Plunger and mechanical plungers

The old physics code had some special Auto Plunger support for mechanical plungers that I'm not sure anyone ever used, and I'm not sure it even worked. The intention was that if the Auto Plunger property was on, and you had an analog plunger device installed in your cabinet, you could fire the auto plunger by pulling back and releasing the analog plunger. This would let the analog plunger stand in for a Launch Ball button, for cabinet builders who didn't want to include both physical controls. The reason I'm not sure this code was ever used is that it required some special handling in the table scripts, which I don't think is documented anywhere.

Just in case anyone knew about this feature and was trying to use it, the new physics code carries it forward, and it should work reliably now. What's more, it should work transparently. It's no longer necessary for the script to do anything special. If the Auto Plunger property is turned on, and the user pulls back and releases the mech plunger, the physics code will fire a KeyDown(PlungerKey) event, followed a short time later by a KeyUp(PlungerKey) event. As far as the script is concerned, the user just pressed the Launch Ball button.

Incidentally, when the Auto Plunger property is set, the on-screen plunger animation only shows the firing motion when a launch is triggered; it doesn't track the analog plunger position. This is because an Auto Plunger represents a solenoid-controller kicker rather than a standard plunger. The motion of this kind of kicker is controlled strictly by the ROM software, so VP disconnects it from the analog plunger motion.

Interaction with "ZB Launch Ball": The Zebsboards digital plunger kit has its own Auto Plunger feature built into the plunger device itself, where it sends a Return key press when the plunger is pushed in **and** the "ZB Launch Ball" LedWiz output is turned on. (The "ZB Launch Ball" output is a virtual output, defined in the DOF configurator, that tells external hardware that the currently loaded table has a button-style launcher instead of a traditional plunger.) The Pinscape Controller added a similar feature in its latest software. There seems to be a potential for conflict here, in that a single release gesture on the plunger could have the effect of generating two Launch Ball key presses – one from the hardware plunger device, and another from VP's Auto Plunger.

I don't think this will be a practical problem, though. When the first Launch Ball key press occurs, the table will fire the auto plunger and send the ball flying, so there won't be a ball in the launch position when the second key press happens a short time later. It won't be any different from physically pressing the Launch Ball button twice in a row – the table will just ignore the second key press because there won't be a ball ready for launching.

Even so, there's a simple fix that could be implemented in the controller devices. Whenever the "ZB Launch Ball" signal is on, the controller should simply report a constant 0 value for the analog plunger position. This will prevent VP from detecting a release motion in the analog plunger, since VP just sees no motion. The user can still launch the ball with a plunger pull, since the controller device detects the gesture and sends the Launch Ball signal to VP. I've already updated the Pinscape software to use this approach, and it seems to produce good results.

Break Over Velocity removed

The Break Over Velocity property in past versions was part of an attempt to handle a tricky problem with mechanical (analog) plungers. Basically, real plungers move too fast for the VP simulation to keep up with. Or more specifically, they move too fast for the USB joystick system to keep up with.

The new plunger physics uses a somewhat different approach to address the same problem. I think the new approach will work a lot better – it should produce consistent and reliable results, which the old algorithm didn't really succeed at. A side benefit is that I was able to remove the Break Over Velocity. It's no longer a factor in the physics calculations, so I removed it from the property sheet. I don't think most people ever had a good handle on what this property was supposed to do, which is understandable – it's a pretty obscure internal part of the old physics that ideally would have just been part of the black box, but I think it got exposed as a property because it was too hard to pick a one-size-fits-all setting for it. Fortunately, the new approach doesn't require a parameter like this.

How to use the new plunger to un-hack existing tables

I got started on this project because I was frustrated by the amount of time I was spending trying to fix plungers on tables I download for my cabinet. I have a mechanical plunger on the cabinet, so it's important to me that the tables I install work with it. But I've found that the majority of tables don't. The reason is that most table builders use various scripting tricks to create their on-screen plungers, and these scripts are often written without consideration for a mechanical plunger.

What I wanted to do was come up with a way to get rid of the scripts that don't work with the mechanical plunger device, and replace them with something else that does work. That's why I did all this work on the VP built-in plunger object: *it's* now the something else. To make the VP plunger viable as a universal replacement for the scripted plungers, I had to make it look good enough to replace the nicely animated, photo-realistic plungers you find in the best tables. That meant it had to look as good as the custom-drawn scripted plungers, and it had to have realistic physics.

I think the new upgraded plunger meets these requirements for my purposes. I'm hoping it's even good enough that table builders will find it to be the plunger implementation option of choice from now on. If they do, future tables will just work with mechanical plungers when you first download them.

But right now, there are lots of existing tables that still have these pesky scripts. For those, I've come up with a fairly simple and quick recipe by which you can remove the scripts and replace them with a simple built-in Plunger that looks at least as good as the scripted version and works properly with any mechanical plunger device. The recipe is fairly simple, but not quite as simple as I'd like. There's still some thinking involved; I haven't been able to reduce it to something purely rote and mechanical. You'll have to be able to read table scripts and find some key parts. But once you find them, all you really have to do is delete them; it's not necessary to understand how they were supposed to work.

Here's the recipe:

Open the table in the VP editor.

Edit the table script. Find the section that implements the plunger object; this is usually a series of three or four Sub (subroutine) blocks and maybe a few Dim (variable declaration) statements. They're usually all grouped together and labeled with something like "So and so's plunger code". There's sometimes a second block of code right after labeled "So and so's pinwizard code", or something similar. Delete all of this code.

Find the lines in the KeyUp and KeyDown event handlers that handle the PlungerKey case. These usually set a few variables and start or stop a timer or two. You can delete most of the code here; just keep the parts that say "Plunger.PullBack" and "Plunger.Fire". The object name might be "Plunger1" instead of "Plunger", or something else entirely, like "p1".

Go to the playfield editor window and find the object or objects located where the plunger goes visually. If you see a bunch of objects, you can delete them all. There are often 12 of a kind, sometimes 25; these are often Ramp or Light objects. They often have names like "p1"…"p25" or "r1"..."r12".

Click the Backdrop button in the editor to view the background of the table. In a few rare cases, there's an EM Reel object on the backdrop that handles the plunger animation. If you see nothing attached to the backdrop, you can skip this step. If you find an EM Reel object here, check its assigned image. If the image looks like a bunch of animation cells for a plunger, you're dealing with one of these rare tables. You can just delete the EM Reel object.

If the table already has an ordinary Plunger object located where the plunger goes visually, make sure this object is visible and "Enable Mechanical Plunger" is checked. Use the little dotted guide line to make sure the rest position (about 1/6 of the way back from the tip of the plunger) is aligned with the position on the playfield where the ball will sit when in the launch lane. There's usually a wall with a notch-shaped front edge that holds the ball in place here.

In some cases, there will be a plunger that's located off to the side of the playfield, so that it won't show up visually when the table is running. Table builders sometimes use this trick to get the input data from the Plunger object without letting the object ever touch the ball (thus bypassing the old VP plunger physics that many table builders didn't like very much). If you find such an off-screen plunger, move it into the proper position that you've opened up by removing the old animation objects, and align its rest position line as above, and set its properties as above.

If you can't find a real plunger object anywhere on the table, create one, and position it as above.

Finally, you just need to set the plunger's visual properties to make it look right for the table. Start by setting it to the Custom type. Import one of the plunger texture images from the plunger mod release package (i.e., the same place you found this guide) – ideally, find one that matches the color scheme of the table's original custom-drawn plunger that you're replacing. Run the table and observe the results. At this point, you might need to make any number of visual adjustments: e.g., tweak the position to get it lined up properly, change the length (via the Stroke Length property) or width (via the Width property) to get the dimensions and proportions just right, and adjust the spring parameters to make the spring look right. It usually takes me a few iterations (run, tweak properties, run again) before I'm happy with the results.

You should now have a simulation plunger that works with the mechanical plunger device as well as the keyboard.