
*Revision 1.0.7*

***Disclaimer:*** *RageTools Pro*, hereby also called *the product*, all its components and code are registered trademarks of *Breno Azevedo da Silva* and *Freakow Interactive*, hereby called *Freakow*.

***Suggestions and Support:*** All product and support-related communication should be carried out through e-mail, directly to *contact@freakow.com*. You will find general information in our 'RageTools' Unity forum thread, Asset Store section.

***Credits:***

**Breno Azevedo** – Project lead, design, code, user support and documentation

**Sandro Bihaiko** – Coder, API decoupling, editor routines, optimization

**Rafael Ribeiro** – Deformation code, general code bug-fixing and math support

**Adelson Tavares** – Demo pieces art and coloring

# 1. Intro

## 1.1. What is *RageTools Pro*?

### 1.1.1. RageTools Pro is a set of components aimed to bring ScaleForm™ and AnimeStudio™-like functionality, and more, to the Unity platform. It builds upon the *RageTools* product functionality, which includes SVG vector-art importing and automated, resolution-independent anti-aliasing and contour smoothing. RageTools Pro adds to this arsenal an advanced vector text generator, RageText, one animation and deformation component, RageMagnet, two animation assistance components, RageIK and RageConstraint, and two selection- and control-assistance components, RageHandle and RageIkHandle. Many helper macros are also included to turn repetitive operations into one-click actions.

## 1.2. What are the *RageTools Pro* requirements?

### 1.2.1. Unity 4.5 or later (older versions might work but aren't supported)

### 1.2.2. RageSpline 1.6.6 or later – available at the Unity Asset Store

### 1.2.3. RageTools 1.1.7 version – available at the Unity Asset Store

### 1.2.4. RageTools was tested and works on the Windows and Mac platforms. Please report if you find any problems

## 1.3. *RageTools Standard* functionality

### 1.3.1. This document only covers the *RageTools Pro* extension specific features, Please refer to the RageTools documentation to learn about all the standard RageTools components – RageGroup, SVG-In, Edgetune, Pivotools and CanvasAlign.

# 2. License

## 2.1. Unless otherwise noticed on a side license, this product utilizes the standard Unity Asset Store license.

# 3. Install.

## 3.1. To install the package: 1) import the RageSpline package to your project in the standard way – *Assets -> Import Package -> Custom Package* menu option, if you haven't done it yet. 2) Import the RageTools standard package. 3) Import the RageTools Pro package. After import you can find our tutorial and sample scenes at RageToolsPro>Tutorial and RageToolsPro>Demo folders.

# 4. Components

4.1. **RageMagnet** – RageMagnet is an incredibly flexible deformation tool (pun intended!), useful for both animation and sculpting of a group of vector shapes. One of its key advantages is that it affects all RageSplines of a certain group at once, independent of their scene hierarchy. A key feature of the RageMagnet is that all the base settings (the ones before "Local Settings") affect all RageMagnets that point to the same RageGroup. This makes it easier to update a number of related magnets (e.g. affecting a tail) at once.

4.1.1. Live – This toggle, when active, actually enables the component effect. *Keyboard Shortcut: "V"*

4.1.2. RageGroup – RageGroup to be affected by the RageMagnet. When you apply a RageMagnet to a Game Object, it automatically searches for the first RageGroup it finds, looking *upwards* in the hierarchy. That is, it finds the nearest parent in the hierarchy that has a RageGroup. You can override that with any group you want, by simply dragging a RageGroup-equipped Game Object to that field, or by clicking the search button next to the field and picking another RageGroup in the scene.

- Draft – The RageGroup's Draft option (in the assigned RageGroup's inspector) makes the RageSplines refresh to perform quicker. For Magnets it might work well or not, depending on how much deformation you're applying, it can even provide hard-to-achieve results since it hides self-intersection artifacts. Read the RageTools user guide for further info.

4.1.3. Set Rest Position – Sets the *rest position*, which is the default position where the RageMagnet doesn't deform the shape, of *all* the magnets assigned to this Group. *Keyboard Shortcut: "C"*

4.1.4. Show Weights – Enable this option to show the group's magnets influence with small disc gizmos over the influenced points. The closer the color is to Blue, the lesser the influence. Conversely, a color closer to Red means a higher influence. This option can be overridden by the "Local Weights" option.

4.1.5. Normalize – RageMagnets work independently, meaning it isn't a full skinning solution – at least not yet. Thus, when you're overlapping decaying magnet areas which affect the same group, you will probably get straying vertices during deformation. In some circumstances, especially with parented magnets, the "normalize" option alleviates or correct those problems. To keep 100% clear of those issues, make sure to not use the *Inner Size* (below 100%) or *Vary Strength* options. The provided *Vilas-walk* scene is proof that you don't need these options for even a complex full-character rig if you play smart with your shapes and magnets.

4.1.6. Local Settings

4.1.6.1. Collider Type – RageMagnets' area of influence is defined by Unity's standard colliders. Both sphere and capsule colliders are supported, and one of each is attached to the object automatically by the script as soon as you add it – although only one can be active at a time. This dropdown box allows you to change the collider type to be used by the selected RageMagnet.

4.1.6.1.1. **Attention:** If the colliders "disappear", remember that it's a standard Unity behavior to hide the gizmos shown by any component when you collapse the settings (using the arrow at the inspector's top-left corner) of *any* component of that type.

4.1.6.1.2. For best performance, RageMagnet colliders use rigidbodies set to "Is Kinematic". Disable this checkbox and enable gravity if you want physics.

4.1.6.2. Radius – That's the radius of the collider defined above. This option can be entered directly in the collider value field if you prefer.

4.1.6.3. Height – When you have the capsule collider type selected, that's its height. This option can also be set directly in the collider value field.

4.1.6.4. Center (X,Y,Z) – Mirror fields to the same-named collider settings, provided as a convenience.

4.1.6.5. Inner Size (%) – Sets the percentage of the default (outer) collider radius that'll define the inner collider radius. Whenever you set this percentage to be less than 100%, you get a decaying effect from the inner collider boundaries (with 100% influence) outwards to the outer collider boundaries, beyond which the magnet has no influence. By default that's set to 100%, which equals to "don't decay the effect, keep it at 100% throughout the collider shape". *This is the recommended setting;* generally you can get smooth deformations by simply playing with the affected spline's points tangent handles size.

4.1.6.5.1. **Attention**: The inner colliders are added automatically and it's strongly recommended that you don't destroy or manipulate them directly.

4.1.6.5.2. Vary Strength – This option allows the RageMagnet influence to decay linearly, along the Y axis, not only radially as it's the case with less-than-100% inner sizes. With it it's possible to simulate linear-gradient falloffs, and even mimic muscle bulges. For this option to work, your RageMagnet collider *must be offset in the Y axis*, either positively or negatively.

4.1.6.5.2.1. Max – Maximum influence value, to be in effect closer to the GameObject center – to see it, make sure Unity's pivot/center button is set to "pivot".

4.1.6.5.2.2. Min – Minimum influence value, or how much effect is applied at the opposite end (considering the GameObject center) of the magnet collider.

4.1.6.6. Local Weights – Overrides the "Show Weights" group option on a magnet-by-magnet basis, allowing you to isolate the weight gizmos visibility.

4.1.6.6.1. Gizmo Size – Size of the group's magnets weight gizmos

4.1.6.7. Handles – By default RageMagnet shows its magnet icon, as a quick way for you to select it in the scene view. Sometimes though, as when you're animating a magnet chain using a RageIK controller, it's very useful to be able to hide the underlying RageMagnets to prevent selecting them inadvertently. This option works on a per-magnet basis.

4.2. **RageIk** – "IK" stands for "Inverse Kinematics", and it's an automated way to, given a target position, calculate all the needed rotations of a hierarchical chain so that its end reach or get as close as possible to that target. In layman terms, think about it as the easiest way to pose an arm, leg, or even a tentacle, by moving a single target controller. RageIk work as a perfect complement to RageMagnet, but notice that you can just as easily pose regular spline hierarchies (segmented parts). RageIk supports multiple segments and angle constraints, to assure that each part of your chain will rotate only a max set amount towards each direction.

- **Attention**: RageIk updates work slower in Edit mode, for real-time testing enable Unity's Play mode.

4.2.2. Live – Enables the IK effect. Rotation of controlled game objects start being overridden.

4.2.3. Always –For best efficiency, RageIk is by default only calculated when the controller changes its position. When "Always" is enabled, it continuously computes the IK effect. This is usually needed when the root of the chain will be moved around.

4.2.4. Align End – Rotates the game object at the end (tip) of the IK-chain according to the target controller rotation. Useful to pose, for instance, a hand using the same controller used to pose the arm. To match the Auto-setup *End Offset* option (read below), if the last joint in the list is named "endJoint", the controller rotates the second-to-last joint instead of the last one. Notice that *Align End* respects any *Ik Joint Limiter* settings applied to the controlled joint.

4.2.5. CCW Bias – Most often than not, IK has two or more solutions to reach a given target. Even with IK Joint Limiters in place this might lead to inconsistent animation. This setting works that problem out, by defining the rotation favoring (bias) of the root of the chain. If enabled, the chain will favor counterclockwise (CCW) rotations. Conversely, when disabled it'll favor clockwise rotations.

4.2.6. Controller Handle – This field points to the object Transform to be used as the target controller for the system. If this controller has a RageIkHandle component, it can remotely change some settings of the actual RageIk.

4.2.7. Joints - In this section you can drag-and-drop each object of the hierarchy, sequentially from the root to the child-most, to define the object (joint) chain to be controlled. Drag and drop objects to the "+" field to add a new joint to the end of the list, and click [x] next to an existing joint to remove it.

4.2.8. Auto Setup – This "magic" section allows you to populate all needed fields and automatically create a controller positioned exactly at the end of the hierarchy, or slightly offset from it – the positions where you'll usually want it. To use it, make sure you've added the RageIk component to the topmost game object in the hierarchy you want to control (e.g.: the thigh for a leg setup), drag-and-drop the last object you want to control in the chain (eg.: the foot, for a leg setup) to *End Joint*, then simply click the "Apply Auto-Setup" button. Supposing you want gaps in your setup, where for instance only joints 1, 3 and 5 should be IK controller but you want direct (or "FK") control over joints 2 and 4, you may add all joints normally then remove the unwanted ones by simply clicking the red [x] button next to them.

4.2.8.1. **Important:** Auto-Setup never creates duplicate controllers or components. If it finds previously created ones, it simply overrides its settings – making the function also useful for bulk-editing of the joints settings.

4.2.8.2. End Joint – The last desired object (transform) to be controlled in this chain

4.2.8.3. End Offset – If enabled, Auto Setup will create an "endOffset"-named Game Object as a child of the End Joint, and offset it in the same direction, and by the same length, of the last chain segment. You'll want to use *End Offset* when the last joint must also point to the controller, commonly the case for long chains. For simpler two-joint chains like arms and legs you'll usually disable *End Offset* and enable "Align End".

4.2.8.4. Limiter – If enabled, Auto Setup will add *Rage Ik Joint* components to all joints, except to the End Offset game object, when present. Auto Setup will also auto-align each RageIkJoint's *Rest Angle* so it points straight to the next joint, which is the default procedure.

4.2.8.4.1. Angle Delta –Defines the minimum and maximum angle variation, from the Rest Angle, to be set in the Joint Limiters. Setting this to zero is similar to disabling the components (turning *Limiter* off), but the latter is the preferred (and faster) way.

4.2.9. *Settings*

4.2.9.1. Joint Gizmos – Shows bone-like gizmos between each joint segment

4.2.9.2. Snap – When set to a value below 1, it provides some delay to the joints rotation. In practice the result is similar to that of time-expensive animation overlap tweaks.

4.2.9.3. Joint Size – Size of the Joint Gizmos to be shown for this RageIk.

4.2.9.4. Joint Color – Color of the Joint Gizmos to be shown for this RageIk.

4.2.9.5. Debug Line –Shows a line with the ideal rotation angle to solve each joint's rotation. If the line is displayed in red, it means the ideal angle can't be reached due to a *Joint Limiter*, so the joint is rotated to the nearest angle to that one.

**4.3. RageIk Joint** – This component is complementary to RageIk, and should be attached to game objects in its *Joints* list, to define their default, minimum and maximum rotation angles. It can be assigned automatically by the "Auto Setup" button, as described in the RageIk section of this document. Min and Max Angle can be adjusted interactively in the scene view, by simply dragging around their gizmos. To do that, you'll need to enable the "Draw Gizmos" toggle and set a suitable value in its "Radius" field, so the limiting cone and the max/min gizmos get large enough to be visible. Another interesting feature is that Joint Limiters can be interactively edited *while RageIk is live*, allowing for a very intuitive and productive workflow. Try it!

4.3.1. Live – Enables the limiting (constraining) effect. Rotation of this game object starts being limited.

4.3.2. Rest Angle – The default or "rest" rotation angle of this joint, usually you'll want it pointing to the next joint in the chain.

4.3.3. Min Angle – Minimum Z angle that this game object will be rotated to, when controlled by RageIk.

4.3.4. Max Angle – Maximum Z angle that this game object will be rotated to, when controlled by RageIk.

4.3.5. Draw Gizmos – Enables the display of the rest, min and max angle gizmos. Global setting.

4.3.5.1. Gizmo Radius – Radius of the rotation limit cone, as displayed in the Scene view. Global setting.

**4.4. RageIkHandle** – This is a selection and operation helper to RageIk. It can be assigned automatically by the "Auto Setup" button, as described in the RageIk section of this document, or manually assigned if you prefer.

4.4.1. IK – Enables the IK effect. Rotation of controlled game objects start being overridden.

4.4.2. RageIk – The component to which this RageIkHandle is associated. To populate it manually, either drag the Game Object that has a RageIk component to this field, or click the button right to this field and pick one from the list. On a previously (or automatically) assigned RageIkHandle, click on this field to highlight the associated RageIk Game Object in the hierarchy list – useful to find the component and change the settings not present in its handle.

4.4.2.1. **Attention**: If this field is not assigned, none of the remaining options will show up in the component editor

4.4.3. Align End – Rotates the game object at the end (tip) of the IK-chain according to the target controller rotation. Useful to pose, for instance, a hand using the same controller used to pose the arm.

4.4.3.1. **Attention:** To match with the auto-setup *End Offset* option, if the last joint in the list is named "endJoint", the controller rotates the second-to-last joint instead of the last one.

4.4.4. CCW Bias – As described in the RageIk section, use this toggle to define the the favored sense of rotation in the system. If the bending looks flipped or "breaking bones", simply toggle this option.

4.4.5. Always – This setting makes the IK calculation happen on every frame, necessary when the root of the chain is moved during the animation. Nevertheless, this is more computationally expensive, so keep it disabled when you're sure only the target controller will be moved during the animation.

4.4.6. Show Gizmo – Enables the gizmo icon on the current controller. It makes selection much easier, and it's specifically designed to not overlap with RageMagnet gizmo icons – although it's generally wiser to disable controlled magnet gizmo icons, as described in the "Animation Tips" section. Differently from a regular gizmo, this gizmo changes to a transparent version if the IK is disabled. To disable the gizmos on all controllers at once, use Unity's default method – select the "Gizmos" dropdown in the scene view and turn off the checkbox of "RageIkHandle", at the scripts section.

4.5. **RageHandle** – This simple component allow you to programmatically set the icon gizmo for its Game Object. The Unity editor has a standard way to do that, by simply selecting the red-green-blue box dropdown (top-left corner, right under the "Inspector" label) and picking any gizmo available in the scene. RageHandle has the advantage that you may switch those icons by setting a simple string in the component, disable its drawing programmatically and also disable all RageHandle gizmos at once. To do the global icon disabling, simply select the gizmos dropdown on the Scene view menu bar and unselect the checkbox next to "RageHandle".

4.5.1. Live – This toggle, when active, actually enables the handle icon drawing.

4.5.2. Gizmo File – Enter a string with the name of the gizmo file, without the extension (.png, .jpg, etc). The gizmo file, to be found by Unity, must be placed in the Gizmos folder of your project's Assets folder. A number of icons are provided in the project folder after you import the RageToolsPro package. For usage examples check the included *Vilas-walk* scene, provided in the Demo/Animations folder. In that scene RageHandle was used for the global controller (Vilas) and the Controller/Root controller.

4.6. **RageText** –RageTools Pro's powerful text generation component. RageText formats and displays vector text dynamically on-screen, similar to what you'd get from Flash-dependent products like ScaleForm™. The huge advantage over the typical Bitmap font is found in the super-sharp anti-aliasing no matter what the font size, no additional texture memory usage and easy styling (like fill color, gradient and outline color) without the need to edit the bitmap images in an external image editor. Besides, RageText offers a "Quick Copy" mode which allows it to operate with next-to-zero performance impact, even on older smartphones. RageText works seamlessly with Edgetune, check the section "Using RageText with Edgetune" under "Use Guides".

4.6.1. Live - This toggle, when active, enables the component effect.

4.6.2. Quick Copy – The 'Quick Copy' mode is much faster than the standard one, and the recommended option for mobile devices. In the other hand, Edgetune doesn't work on the RageText when this option is used. So if you use Quick Mode you should duplicate and set, before hand, one different RageFont object per size and either have a fixed anti-aliasing (using RageGroup's *tweak* option) or add EdgeTune to the Font's root and set it to 'start only'.

- **Attention:** Quick Copy doesn't work with Prefab Fonts, only with instantiated ones.

4.6.3. Always – When enabled, RageText will update continuously instead of waiting for the "Text" property to change. Useful when interactively tweaking RageFonts kerning.

4.6.4. Font – Assign a "RageFont" game object or Prefab to this field. These container objects don't require a specific component; simply make sure they use the exact same format as described in section ***"How to create a new RageFont".***

- The three provided RageFont prefabs can be found in the folder RageToolsPro/Prefabs.

- **Important:** If you drop an instantiated (from a Prefab) Ragefont in this field, and the Quick Copy mode is enabled (read below), this instance connection is automatically and irrevocably broken. This might be necessary in most cases due to the recent prefab changes in Unity, to prevent text shapes from disappearing when using Quick Copy.

4.6.5. Buffer – RageText increases and decreases the amount of display characters interactively as needed. Both the initial amount of display characters and the increase/decrease amount are defined by this field. For mobile devices development, make sure you have as many characters defined on *buffer* as you'll ever need for each line, or else you might suffer from hiccups as new characters are instantiated or removed.

4.6.6. Text – The actual text to be displayed goes here. You can set it programmatically using the (unsurprisingly-named) *Text* property.

4.6.7. Alignment – Aligns the RageText to the right, left or center of the text maximum size (initially defined by the "Buffer" settings field), using spaces as fillers when needed. It also aligns the generated text to the RageText *box collider* boundaries, if it finds any. Use this collider to define the 'line size' and/or positioning. Remember you can press and hold Unity's default "shift" keyboard shortcut to show and interactively tweak the box colliders size handles.

4.6.8. Tracking – Set this field to anything other than zero if you want additional letter spacing (tracking) to be applied to the whole line. Basic inter-letter spacing (kerning) is defined per RageChar, by its collider size. Tweak each RageChar collider individually to suit your needs. Check the *RageFont Setup Macro* below for the automated setup process.

4.6.9. Container – You can set an optional "Container" CanvasAlign component here. This container is generally the parent of the RageText element, with a CanvasAlign component and its own *box collider*. The container box collider boundary is usually set to encompass the RageText bounding box. If you

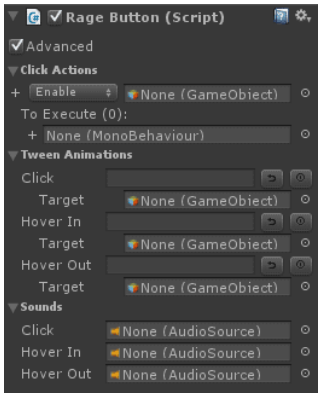
know HTML concepts, think of it as a *padding* area, relative to the screen edges. Check the *tutorial-6-RageText* scene for a working example.

4.6.9.1. **Attention:** Changing the Alignment of the RageText change its container's alignment.

4.6.10. Styling a RageText (changing fill and outline colors, gradients, etc) is very easy. Simply assign the desired style(s) to the *Styles* section of the related RageGroup. Multiple styles with name filters also work. If you're using Quick Copy you will need to style the instanced RageFont characters directly.

4.7. **RageButton**– When setting up a GUI you are usually required to spend precious time coding even the simplest of the interface buttons. RageButton brings the *fun* back (or would it be "for the first time ever") to GUI creation, allowing you to simply create animations visually and hook the behaviors through drag-and-drop and naming match up. RageButton uses standard Unity audio sources for sound effects and trigger colliders (any type) to define the button area that will be clickable. It also relies on the free and powerful HOTween library (included) to create its hover in, hover out and click tween animation without a single line of code, through its visual editor component. You may also enable, disable and instantiate game objects when clicking the button (which could auto-start a particle system or animation, for instance), and hook up more complex behaviors from your own scripts.

➢ A recent HOTween version (1.1.600) was used as the basis to this release, with only a couple cosmetic and usability improvements. Although we only support the connection bridge with this version, the code infra-structure has remained compatible with RageButton's calls for months. As such, chances are high that future versions will keep compatible – and we're continuously in touch with HOTween's creator, Daniele Giardini, talking about future improvements. To completely disable HOTween from your project check the *"Disabling HOTween"* section. For product support and to check for the latest HOTween version and its complete documentation, including a video tutorial on the visual component editor, go to http://www.holoville.com/hotween/




4.7.1. Advanced – Shows all advanced attributes like game object management actions and specific tween animation targets. When the target is not set, all hierarchic children tweens with the same ID string defined in the Click, Hover In and Hover Out fields will be played.

4.7.2. Click Actions – Expand this foldout to feed in what you want to execute and operate on once the button is clicked.

4.7.2.1. Operations List – Only shown when the advanced mode is enabled, in this row you select an operation and define which game objects will be operated on. You can add new rows by simply dragging game objects to the operator game object field (next to the "+" sign). This way you're able to perform multiple actions on the same or different objects.

4.7.2.1.1. Operation – dropdown where you select an operation to be executed on the row's game object once the button is clicked. May be "Enable", "Disable", "Instantiate" or "Delete".

4.7.2.1.2. Delete Button – by clicking the red "x" you remove the entire row

4.7.2.2. To Execute – To fire up any custom script (monobehaviour) when clicking the RageButton drag its host game object to this field. All "*OnClick*" methods in the game object components will be fired – so make sure to name the method exactly like this. You may add more items to the "To Execute" list by simply dragging more objects to the empty field next to the "+" sign. You may delete previously added items by clicking on the red "x" button next to it.

4.7.3. Tween Animations – In this foldout you enter the "IDs" of the HOTweens you want to play when each ragebutton event happens. IDs are regular text strings which you must also enter in the target game

object (s) HOTween ID fields. By default the RageButton's host game object and all its children receive the HOTween play call, if you want to defined an external game object to work as a message receiver (including its children), enable the "Advanced" mode and fill the Target field below each Tween Animation entry type.

4.7.3.1. Tween Animation Events:

4.7.3.1.1. Click – fired when 'mouse 0' was pressed while hovering the RageButton collider

4.7.3.1.2. Hover In – fired once the pointer moves over the RageButton collider area

4.7.3.1.3. Hover Out – fired once a hovering pointer leaves the RageButton collider area

4.7.3.1.4.  Play Reverse – Plays the Hotween animation backwards. Useful, for instance, when the hover-in and hover-out animations are exactly mirrored in time, so you only need to create and tweak one tween instead of two.

4.7.3.1.5.  Reset before Play – When active, resets the HOTween clip before playing it

- **Hint:** You'll usually want this enabled for repetitive tweens, or else they'll only play once. When the same tween is used for hover-in and hover-out (usually with hover-out set to reverse) you'll want this disabled so the animation smoothly transition between states.

4.7.4. Sounds: usually you'll want sound effects played when the RageButton events happen. That's what this foldout works for, by assigning standard Unity AudioSource components. You may only have one AudioSource component per game object in Unity, so when you need different entries simply create multiple game objects and attach one AudioSource component to each. More info in the Unity docs.

4.7.4.1. Click, Hover In and Hover Out: AudioSources to be played when the corresponding event (read previous section) is fired

4.8. **RageSprite** – "Animation is the visual illusion of life provided by a sequence of frames shown in rapid succession." Many techniques are used to create those animations; deformation-based animations for instance can be created with RageMagnet. Cut-out or parts animation can easily be done with RageTools standard features. Traditional animations (also called Switch-type or "cel") on the other hand, use completely different drawings, one per frame. This technique was popularized by Disney and Warner Bros hand-drawn animated cartoons and is commonly used in a number of electronic games, with bitmap-graphics sprites. RageSprite allows you to do the same, yet with the power of vector graphics. By relying on standard animation clips and its Unity API commands, RageSprite is instantly compatible with any state machine code you already have, and visual solutions like Playmaker or uScript – simply use the default animation play commands. RageSprite animations have extremely high playback performance, making it perfectly suitable for mobile games and simulations.

- Follows the component's basic functionalities:
  - Assign and preview your animation frames and its individual data – order and duration
  - Generate Unity-standard animation clips which you can play with any state machine (code or visual)
  - Actually switch the frames in play/run time once those animation clips are generated
    - Hints: Try a Base Frame Delay of 0.1 (100ms) for starters. Preview by clicking on the play button, toggle it off, tweak values, rinse and repeat. When adding multiple files, they will always be ordered alphabetically, so it makes sense to name the frames in ascending numeric order. Check the ragesprite-runner demo scene for a complete setup. When you're done, click "Generate Clip". The clip will be created at the project root and auto-added to your Animation component, make sure it's set to *Play Automatically* and click Play in Unity to see the final result.


4.8.1. New Animation – Create a new animation entry. Multiple animations will be shown in sequence.

4.8.2. Generate Clips – Exports all animation entries as default Unity .anim files, at the project root. *To work, these clips have to be assigned to the animation component at the same Game Object that hosts RageSprite.* This is automatically done when you generate a clip.

4.8.2.1. **(new)** RageSprite now works with both Animation (legacy) and Animator (Mecanim) Unity components. It'll auto-detect the component type attached to the host GameObject and create the proper Animation Clip in your Assets folder. If there's an AnimatorController assigned to your Animator, it'll auto add a state and assign the created clip to it. **ATTENTION:** If no Animation or Animator component is attached, no Clip will be generated.

4.8.3. **Animation Foldout** – Expand the animation entry by clicking the foldout arrow next to its name – initially named "NewAnimation…"; Only one animation entry may be shown (expanded) at once.

4.8.4. Animation Name – Name your RageSprite animation here

4.8.5. Delete Animation – Deletes the RageSprite animation currently shown.

4.8.6. Preview Box – The frames preview outline is shown in this area. Quality varies according to settings.

4.8.7. Play Button – While enable, plays back the animation in loop, as close to real time as possible. As soon as you click Play the preview cache is built, playback smooths out. Click again to stop previewing.

4.8.8. Generate Clip – Similar to the "Generate Clips" button, but exports only this RageSprite animation.

4.8.9. Preview Scrub Slider – Click and drag this control to manually preview the succession of frames.

4.8.10. **Frames** – Expand this foldout to feed and tweak your animation frames values

4.8.11. Base Frame Delay – The default duration, in seconds, for each animation frame

4.8.12. Move Down Button – Moves the frame entry down in the list, making it be displayed later. *Shift+Click to move the frame to the end (bottom) of the list.*

4.8.13. Move Up Button – Moves the frame entry up in the list, making it be displayed earlier. *Shift+Click to move the frame to the beginning (top) of the list.*

4.8.14. Frame Game Object – Host or Parent to the RageSpline component(s) of this frame

4.8.15. Frame Extra Delay – The final frame delay is the sum of the base frame delay and this value

4.8.16. Delete Frame – Removes the frame of the list, re-ordering it and cleaning cache links automatically. *Shift+Click to delete all frames.*

4.8.17. New Frame Game Object – Drag and Drop a frame GO here to add it to the bottom of the list

4.8.18.  Add Children – Very useful to add multiple frames at once instead of dragging them one by one. When this button is enabled, the next Game Object you drag to the *New Frame Game Object* field won't be added; instead, this game object's immediate (1st level) children will be added.

4.8.19. ***Settings:***

- Enable Preview: The inspector preview may slow down the editor refresh. Disable this toggle to stop rendering it, without affecting the actual animation playback in the scene and game view.

- Draft Preview: When ‘Draft Preview’ is enabled only the shapes points are drawn, connected by simple lines. Disabling this toggle will switch to the ‘Quality’ preview mode, which uses anti-aliased bezier lines. Especially with complex drawings it’s best to keep this toggle enabled.

# 5. Macros

5.1. Intro – Macros are one-shot functions that automate global or specific functions that would be otherwise tedious to perform manually. They can be accessed through the Unity component/RageTools/Macros menu, and in some cases require you to have a GameObject selected in the scene to work.

5.2. RageMagnet Hierarchy Live Toggle – Checks the state of the currently selected RageMagnet and sets its inverse on all RageMagnets found in its children hierarchy.

5.3. RageMagnet Hierarchy Set Rest Position – Sets the rest position of all RageMagnets found in the currently selected object's hierarchy.

5.4. RageFont Setup - After importing an SVG file in the format required by RageText to be used as a RageFont, use this macro to automatically format the RageChars (individual letters) and respective elements to their proper naming conventions so they can be properly identified by RageText. For instance, a percentage sign is composed of three elements (separate shapes), two small zeros and one forward slash.

5.4.1. Format Chars – The char elements must be named as a numeric sequence (1,2,3, etc). Also there are some special symbols whose "names" aren't properly exported by SVG. This macro works all this out for you automatically, when the "Format Chars" option is toggled on before hitting "Process".

5.4.2. ***(new)*** Create Space – Adds a space character automatically, based off of the largest character size.

5.4.2.1. Width – Width percentage to be used of the reference character in the RageFont.

5.4.3. Create Colliders – With that option enabled, the macro creates the box colliders and set an appropriate size – ie. Encompassing all the composing elements size – plus the additional kerning size you want as a default for your characters. After using the macro, it's recommended to manually tweak some characters for the best possible results.

5.4.4. Kerning – the amount of Kerning (inter-character spacing)

5.4.5. Collider Z Depth – an optional z-depth amount to be set to all colliders, useful if you want to play with the text character's physics later on.

# 6. Use Guides:

## 6.1. RageMagnet Quick Use Guide

6.1.1. Add the component to a Game Object with a parent. The parent (root) object should contain a RageGroup component.

6.1.2. You can have multiple nested Game Objects, RageMagnet will always search and find the first parent it finds (looking up the hierarchy) to set as its RageGroup. Optionally you can set it by dragging a RageGroup component to the RageGroup field.

6.1.3. Click the 'Set Rest' button to set its rest position and rotation, that is, the default position and rotation where it won't affect any vertices. The Sphere collider radius is the RageMagnet maximum radius of effect

6.1.4. Optionally set the *inner Size* to anything less than 100% to decay the influence (or weight) of the magnet up to its outer limit. The inner radius influence is always 100% (unless you have the 'Vary Strength' option enabled), and the outer radius influence is always 0%.

6.1.5. Please notice that both *Inner Size* and *Vary Strength* are designed to work for independent/isolated magnets, if you have intersecting magnets (generally the case for characters) you shouldn't use those options since they generally lead to unwanted distortions. You can try the "Normalize" option to minimize those distortions. In the future we intend to have a full skinning solution which will properly support the intersection of decaying influences.

6.1.6. Enable the "Live" toggle to activate the component. Move the hosting Game Object around to see its effect. You can "sculpt" a shape by un-living the component temporarily, repositioning it, re-setting its Rest position and "re-living" the component. Rinse and repeat as needed.

6.1.7. You can create add an animation component to the root object and animate the RageMagnets position and settings in the animation editor just like any children Game Object. For best performance move the Magnet hierarchy out of the RageSplines hierarchy, please read the "Animation Tips" section for more information.

## 6.2. How to create a new RageFont from a regular system font:

6.2.1. In the RageTools/Demo/SVGFiles folder you will find *RageFont-Atarified* and *RageFont-Verdana*. These are the original SVGs that were imported as the included RageFonts. Checking them out might be all you need to understand the needed structure, yet the actual process is outlined below. Be aware that right now this is still an involved process; we intend to drastically ease the workflow in the near future.

6.2.2. Inside your Vector Drawing software (we recommend *Illustrator* but the free *Inkscape* will do fine), select the desired font, and use the text tool to create one individual letter, we recommend using font size 100 (any measurement unit). Create one layer per letter of the font that you want to use with RageText. You can have a RageFont with only numbers if you wish, it's much faster to set up and will work fine for scores and other number-only text displays. You can also have fonts with only the upper case characters, with or without special characters. It's all up to you really, just remember that the more characters you use, the more setup work you'll need. E.g.: if you're only using it for scores, you can do just numbers.

6.2.3. It's recommended to align the individual characters vertically to the center of the canvas. Don't align them horizontally (along Y); keep the original horizontal offset of the font letters.

6.2.4. Convert the letters to outlines. Clean up the resulting shapes and get rid of the redundant (too close or overlapping) points. Deleting a point in Inkscape preserves the original shape curvature as much as possible, for Illustrator we strongly recommend a super-handy plugin called PathScribe (formerly "BetterHandles"), and its "Smart Remove Point" functionality. Starting from RageSpline 1.5 you can also achieve a similar result right inside Unity after importing the font SVG.

6.2.5. RageSpline only supports closed shapes, yet since RageTools 1.1.4 you can import compound / hollow shapes (ie. shapes with holes) like the letters "A" and "O" by simply enabling the "Create Holes" SVG functionality. The final result should present a single <path> for each character element. Elements are isolated shapes – for instance, the percent sign is composed of three elements, two "o"s and one "/".

6.2.6. Group each element(s) shapes, and name the group exactly as the character. For the percent sign example your group name would be "%" (without the quotes).

6.2.7. Make sure you don't sub-group those shapes. All the shapes into the group should be at the same level, right under the group.

6.2.8. Finally select all groups and group them once more, or move them to the same layer – groups and layers are exported just the same to SVG. This top layer will become your RageFont root object.

6.2.9. Save to SVG, making sure to select "ISO" as the encoding format. If you use UTF-8 or UTF-16 some special characters won't be recognized by the RageFont setup Macro after import. Save it to your project Assets folder as usual.

6.2.10. Inside Unity, create a new Game Object, add the SVG-In component and assign the exported SVG file. Inside settings, set "offset" to 0 (unless you've exported pre-generated outlines), also select "Outline Behind Fill" if your font has outlines. Click Import.

6.2.11. With the root game object of the font still selected, select the menu item *Component* > *RageTools* > *Macros* > *RageFont – Setup*.

6.2.12. If you set the font size originally to 100, the default kerning setting of 10 should do. Click 'Process' and the font will be ready for use. Drag the root object to your RageText display object and test it out.

6.2.13. To be able to style the font, add a RageGroup component to the root Game Object, optionally assign a RageGroup style. Create a prefab with it for ease of re-use in your project and you're done.

## 6.3. Styling RageFonts for QuickCopy mode:

6.3.1. There's a "Styles" foldout inside RageGroup which allows you to quick and easily style groups of shapes – including the characters of RageFonts. To use it, do as follow:

6.3.2. Instantiate the entire RageFont in the scene, make sure the whole hierarchy is enabled. Update (or add) the RageGroup at the root.

6.3.3. With RageGroup selected, simply open the foldout "Group Style" and assign a pre-created RageSplineStyle (check *RageSpline* documentation for further info).

6.3.4. Click "Apply Style" and watch the characters change. While you're at it, feel free to enable "Group Tweak" and update the antialiasing and density settings of the font, which generally changes according to the size of the font.

6.3.5. In the RageText game object, assign this instatiated and customized version of your RageFont and enable "Quick Copy".

## 6.4. Animation Tips:

### 6.4.1. Setup and Selection

6.4.1.1. Turn off the camera icon to stop it from confusing your magnet selection. Click on the Gizmos pulldown menu, at your scene view top bar, and click over the camera icon so its greyed out.

6.4.1.2. For each IK Chain, select the individual Magnets controlled by it (eg. Arm and forearm), and un-toggle "Show Gizmo" at its local settings foldout. This will prevent you from selecting it inadvertently during animation, which would kick you out of animation mode.

6.4.1.3. When you're on a pose where controllers overlap, it's hard to tell if you got the right magnet selected. In these cases get used to selecting the right one directly in the Hierarchy view – a good reason for properly naming and grouping your magnets beforehand.

6.4.1.4. Sometimes when you want to select a magnet close to the one already selected, the rotation gizmo will get in the way. An easy way to select the other controller is to switch to the transform tool ("W" shortcut), which has a much less obstructive gizmo.

### 6.4.2. Rigging

6.4.2.1. To create an IK controller takes literally two clicks. Select the magnet which is the root of the chain – say, the Thigh - , then (1) add RageIK and (2) click "Auto-Setup". Like magic, all fields are properly setup for you. Controllers will be created with a sensible name right below your hierarchy root – generally where you'll want it, if you imported it using SVG-In. Set it to Live to put it in effect, and if the chain is bending opposite to its intended/natural direction, simply toggle the "Flip Bias" option to get it fixed.

6.4.2.2. Remember the Animation Clip creates keys for anything below its hosting Game Object in the hierarchy. Build your animation structure (rig) carefully so that, for instance, you don't create keys for the actual magnets when you're animating using an IK controller. To prevent that, simply move the magnets out of the actual controller hierarchy (where your animation component lies), and apply RageConstraints to keep these guys "virtually parented" to their original parents. Check the Vilas-Walk demo scene for a working rig sample.

6.4.2.3. To make non-specific controllers (like RageIkHandles and RageMagnets, which got their own gizmos) easy to select and animate, use RageHandle. Simply apply its component to the desired object, then type a string with the name of the gizmo you want to be displayed on that object. A couple gizmos are provided, and you can add your own image files to the 'gizmos' folder, only the file name is needed to fill the "Gizmo File" field, you can omit the extension. Use PNGs for smooth-transparency gizmos.

### 6.4.3. Animation Editor

6.4.3.1. It's a good idea to directly create the animation curves for only the channels you want to animate. Generally you'll want the rotation channels curves created, if you'll use squash-and-stretch animation, key the position channels too. After that, click the "Show:" button at the bottom of the Animation window, that'll switch it to "Show: Animated", and clean up the view from all the unused channels. Clean is always good, aye?

6.4.3.2. Aside from selecting keys and hitting "F" to 'frame' them in view, two nice non-documented shortcuts similar to Maya's are Shift+Alt+RMB+drag and Ctrl+Alt+RMB+drag (Cmd+Alt+RMB+drag on OSX). These – far from obvious – combos let you scale the view in the vertical or horizontal directions respectively. Try it, it's very handy.

6.4.3.3. Make sure to add the Animation Component (found in Unity's Components>Miscellaneous menu) at the root level of your controllers hierarchy. The more components and parameters that it finds below itself, the slower it gets, so if you add it above the actual RageSplines group, performance will generally decay way below the usable level.

6.4.3.4. Due to the above, if you intend to animate the magnets using the Animation Editor, they **should not** stay at the RageSplines/RageGroup hierarchy, it just has to point to the proper RageGroup. One trick you can use is to add RageMagnet to the RageGroup, to get it auto-assigned, then move it out of the hierarchy. The RageGroup reference will be kept.

6.4.3.5. Interactively editing animations is fine and dandy, yet most of the time it'll create extreme euler rotation keyframes which'll give you multiple unwanted rotations. You may try to prevent that by setting the rotation interpolation of the channel to Quaternion (using the right-click menu) but bear in mind that during our tests we've got weird, unwanted keys created with this option active. The most reliable animation method, in our experience, has been to simply create keyframes at the desired time and edit it directly in the animation editor, either through value editing (also needed for copy-paste) or by dragging the keyframe up and down to change its value semi-interactively.

6.4.3.6. Note that you **can** update an animation during play time by editing the channel values or dragging the keyframes in the animation window. This method is useful when you want some specific interaction with a physics-enabled feature, like a joint or collision. Be creative and see what you get ☺

6.4.3.7. Did I mention it's not a good idea to create rotation keyframes interactively with the default Rotate controller? For Translation keys that's fine, but if you want to use the "saner" Euler keyframes (especially desirable if you're going to animate loops due to the Quaternion max-180$^{o}$ rotation restriction), you'd better stick to dragging the rotation keyframes directly in the Animation Editor.

6.4.3.8. Remember you can create a keyframe for only the selected channels, use that to your advantage so you don't, for instance, create a Position keyframe when all you want is a Rotation keyframe.

6.4.3.9. There's a **lot** going on under the hood when you use constraints and IK simultaneously, and depending on your processor speed you might get some redraw and update inconsistencies when you jump around the timeline in the Animation Editor, or after playing the animation. Simply drag the timeline to the immediately previous frame, then back to the one you're viewing or want to edit, so things are properly updated. Other actions like creating a new keyframe (shortcut "K") will also refresh the Animation Editor. These issues have been greatly diminished due to recent improvements done to the system parts inter-communication.

### 6.4.4. Mobile Devices Performance Tips

6.4.4.1. Deformation looks cool and organic but it is CPU-intensive – and the more points the shape has, the worse. For mobile devices development, due to its limited performance, it's wise to favor movable parts (e.g.: arm, forearm, etc.) and/or *RageSprite* animations over magnets. On the other hand, the simplified 2D IK algorithm we use is very fast and shouldn't affect performance significantly, so feel free to use it to easily pose a character's separate parts.

6.4.4.2. Sparse and focused use of RageMagnet can still be suitable and advantageous for mobile games. For instance, a deforming ball bouncing on the scenery, a cape or a boss creature's tentacle. The fewer control points (density) the RageSplines shape has, the better.

### 6.5. Using RageText with Edgetune

6.5.1. RageText was designed with Edgetune in mind, so that you can scale your text elements to any size you want while keeping perfect anti-aliasing and contour smoothness – a huge part of wanting to use vectors instead of the age-old bitmaps, right? In a nutshell, if the Edgetune component (required by RageText) is live, it works in tandem with RageText, simple as that. Follows some usage tips:

6.5.2. Select your RageFont and type some text as usual, preferably showing some rounder characters like "O" and "@". Disable Edgetune by unselecting the "live" toggle. That'll allow you to tweak the visual appearance of the characters with RageGroup.

6.5.3. Within RageGroup, enable the "Group Tweak" setting. Interactively edit the density and anti-aliasing until it looks good on your Game View. Doing it with the "proportional" option enabled it's generally sufficient. That's an artistic decision, it's up to you how crisp or smooth you want your anti-aliasing to look. You might want to scale up your Game View some before carrying out this step, to see the letters contours in more detail.

6.5.4. Disable Group Tweak. Back to Edgetune, click on "Snapshot" to sample the current Game View size, then "Initialize" to sample the object height on-screen and current camera z distance from object (for perspective cameras). Enable "Live", and scale the RageText game object up and down.

6.5.5. If the text gets too blocky while scaling down, you might want to reduce the "Density Factor" inside Edgetune's 'settings' foldout. With the default 'Verdana' font size we've got good results with a Density Factor of 0.5.

6.5.6. If the text contours get faceted while scaling up, sensibly increase the Auto Density 'max' value. That's how much subdivided the text can get as you scale up. Please notice that a high value will slow down processing sensibly as the shapes size increase on-screen.

6.5.7. Although the "Optimize" option in RageSpline can sensibly reduce the total polygon count of your RageText, it presents outline width consistency problems, quite noticeable in letters like "W". If you 'expand' your outline into curves inside an external editor before importing a new RageSpline, that won't be a problem and it's generally a good idea – except that you won't be able to edit the outline width after importing the font, of course. *Optimize* is also strongly **not** recommended if you're going to deform the RageText (or anything else) with Magnets, and **never** use it with the Draft option of RageGroup.

## 7. Disabling HOTween

7.1. RageTools Pro comes bundled with the free and highly efficient "HOTween" animation tweening package. It's required by RageButton, but if you don't need it or any tweening in your project and wants to clear up your folder, you can remove HOTween entirely. Simply select your scene's RageCamera and select the "Toggle HOTween" right-click context menu option. It automatically moves the _ThirdParty/HOTween folder out of 'Assets' and into the project's root, just as the dependent RageButton scripts. Toggling it again reverses the operation. After disabling it feel free to completely delete the moved folder and files from the project root.

## 8. Flash Publishing *(deprecated in latest versions)*

8.1. RageTools Pro 1.0.5 is compatible with Unity's Adobe Flash Publishing mode. For it to work, first you must disable Farseer (check RageSpline docs) and HOTween (read above) before switching to Flash on Unity's Build settings. Most editor operations should be carried out with the build / deploy option set to PC/Mac, only switching to Flash before deploying the project.

## 9. Conclusion

RageTools Pro has been on my mind for almost a year now, since the first time I envisioned the RageTools product, and being able to bring it to the amazing Unity community is nothing short of a dream come true. As a matter of fact, most of its additional feature set was initially designed for a single 'RageTools' package, but reality constraints and the community urgency for a simpler "RageTools" turned it into an extension pack for RageTools itself. And wow, how long it took, how many tears and sweat drops. With all said and done, I can barely believe I'll finally be able to reply those "When will RageTools Pro be released?" e-mails and PMs with a sound "It's live!" ☺

Never-ending thanks to my Freakow mates, Sandro and Rafael. Sandro Bihaiko is a senior coder able to push any complex piece of code over that extra quality mileage it needs to succeed, and his mentorship was absolutely necessary to get us out of some dark and obscure corners along the way. Rafael Ribeiro code support and inventive math skills were so important that, without him, RageMagnet simply wouldn't do half of what it does now. Finally, I'm sure I talk in the name of all of us from Freakow when I thank our families for the continued support and understanding of the interminable late night Monday and Wednesday meetings (just to cite the "regular" days).

We're also eager to listen to your opinions and suggestions about it, also don't think twice before sending an e-mail telling us about anything cool you create with it. Last but not least, thank *you* for buying and supporting RageTools Pro, count on us to fix any problems you find with the product. Enjoy RageTools Pro!