2022
4
ISSN 1433-2620 > 26. Jahrgang >> www.digitalproduction.com
Publiziert von BuschGlatz Germany GmbH

Deutschland € 17,90
Österreich € 19,–
Schweiz sfr 23,–

DIGITAL PRODUCTION

DIGITAL PRODUCTION

MAGAZIN FÜR DIGITALE MEDIENPRODUKTION

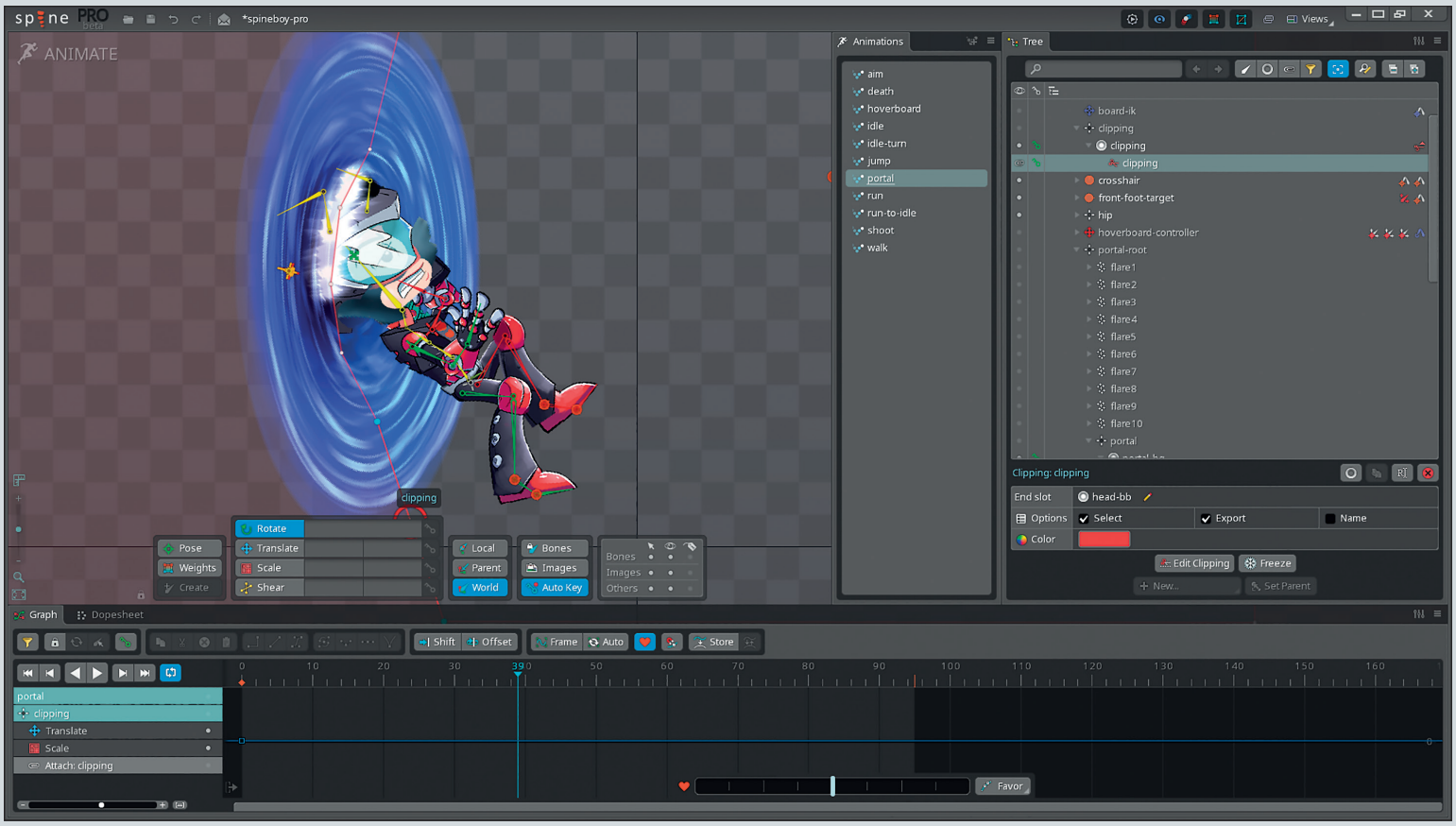
JULI | AUGUST 04:2022

Hardware!
Sehen, hören, tippen, selber machen!

Projekte
Lightyear, Life is Great, Blender VHS FX Workshop

Tools
Renderman, Omniverse, Flair, Flame und mehr

Praxis
Roundtripping, Helium, Zynaptiq, XML und, und, und



Spine hält auch einige Spezialfunktionen bereit, z.B. ein Clipping-Tool, mit dem das Verschwinden oder Erscheinen von Charakteren wie hier durch ein Portal ermöglicht werden.

Smooth Moves mit Spine

Spine hat sich als die erste Wahl etabliert, wenn es um 2D-Game-Character-Animationen via Unity, Unreal oder diverse andere Entwicklerplattformen wie Cocos2D und XNA geht. Aber auch abseits dieser Hauptanwendungen findet Spine häufig Verwendung in anderen Sphären. Wer in den Untiefen des Webs schon auf Abwege geraten ist, wo dubiose Casino-Websites aufpoppen, wird hier und da Bekanntschaft mit dem unverwechselbaren Charme von Spine-Animationen gemacht haben.

von Cornel Hillmann

Häufig sind es exotische Anime-Schönheiten, die mit wallend gleitendem Haar und Diamantglitzer um den Augenaufschlag den Besucher mit sanfter Lockbewegung um die Kreditkarte bitten, oder es sind bildschirmfüllende Schlachtschiffe, die im wogenden Meer abrauchen, um den einen oder anderen Web-Zocker anzuködern. Sprich: Spine ist Chef in Sachen komplexe Image-Animationen, die fast auf jeder Plattform, inklusive Html5 via Canvas oder WebGL, hochperformant und butterweich laufen.

Der 2D-Spezialist

Esoteric Softwares Animations-Tool Spine ging ursprünglich aus einem Kickstarter in 2013 hervor. Der Wunsch von Entwickler:innen und Artists war, endlich ein seriöses Tool zu haben, das skelettales 2D in Games vereinfacht und die gestalterischen Möglichkeiten voll ausschöpft. Herausgebildet hat sich daraus eine Art 2D-Realtime-Animationsstandard, der äußerst vielseitig ist, was Plattformen und Anwendungen angeht. In erster Linie sind dies natürlich skelettale Animationen für Character-orientierte 2D-Spiele, häufig für mobile Endgeräte via Unity oder Unreal. Oft sind es aber auch Html5-Spiele, webGL-Anwendungen wie animierte Websites, Online-Ads oder auf der anderen Seite manchmal sogar auch animierte TV-Spots mit mehreren Charakteren und Dialogen im Broadcast-Bereich. Ebenfalls schwer im Kommen sind Digital-Art-Installationen mit komplexen Bone-Konstrukten und subtilen Idle Loops, die dem bei Medienfunktionären so beliebten Begriff „Bewegtbild" eigentlich am nächsten kommen.

Was herausragt an Spine, sind das exzellente User Interface, die filigranen Skinning Tools, um präzise Vertex-Gewichtungen zur Bitmap-Deformierung zu gestalten, die übersichtlichen Animationswerkzeuge mit Dopesheet und Graph sowie die vielseitige Exportunterstützung von Game Runtimes, um eine tiefe Integration in die Game-Mechanik zu ermöglichen, wie z.B. die Bone-Orientierung mit Ausrichtung der Maus zu verknüpfen oder einen Partikelemitter dem Bone eines Gewehrlaufs folgen zu lassen. Besonders schön: Spine-Animationen laufen in der Regel butterweich, da sie sich immer an der gehosteten Framerate orientieren. Die ist bei Games in der Regel hoch, während die Dateigrößen durch den effizienten Texture-Atlas entsprechend sehr klein sind, da ja die Animationen via Bone-Rig in Echtzeit entstehen und nicht wie bei Sprite Sheets als volle Einzelbilder abgerufen werden müssen.

Natürlich gibt es im Bereich 2D-Animation jede Menge Konkurrenz. Neben Engine-internen Lösungen wie Unity 2D Animation, Open-Source-Kandidaten wie Dragon Bones, Feature-reichen Projekten wie Creature 2d, oder aber auch video-/filmorientierte Lösungen wie Moho in einem ähnlichen Preissegment. Trotz dem vielfältigen Umfeld hat Spine eine Sonderrolle, denn es trifft den Sweet Spot, den die Branche benötigt: Artist-friendly, breite Runtime-Unterstützung, präzise Tools und verlässliche Entwickler:innen mit überzeugender Roadmap. Nicht umsonst sind gute Spine Artists sehr gefragt, denn Spine-Talent is rar und es gibt nur wenige Artists,

die die Möglichkeiten von Spine ganz auszuschöpfen vermögen. Die wirklich herausragenden Spine-Zauberer sitzen häufig in Asien bei den großen Mobile-Game-Entwicklern.

Samtweiche, vorsichtig gewichtete, subtile Bewegungen aus Illustrationen, häufig im Anime-Bereich, herauszuholen ist Kernkompetenz bei Spine. Weiterer Bonus: Einmal gekauft sind alle zukünftigen Updates im Preis enthalten.

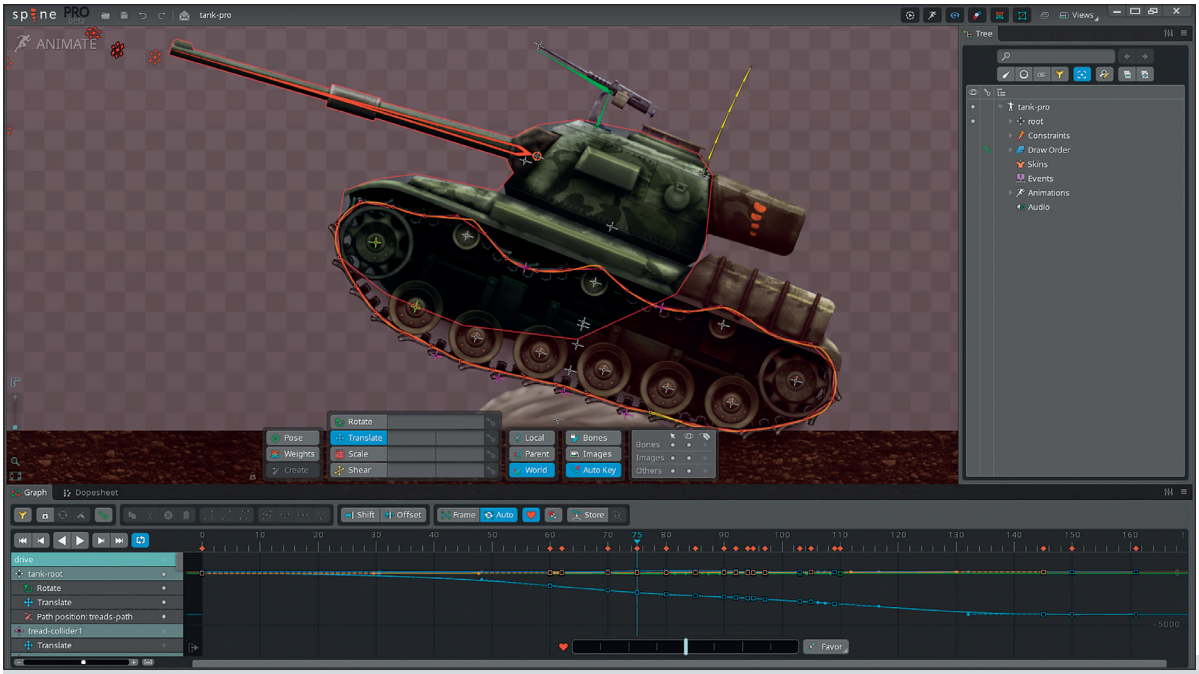
Der wesentliche Vorteil einer Spine Runtime gegenüber Sprite Sheets, die ja auch weiterhin häufig für 2D-Games verwendet werden: Neben der Tatsache, dass die Framerate nach oben offen ist, und dem Fakt, dass die Dateien sehr klein sind, gibt es die Möglichkeit, sehr viel fließender zwischen den verschiedenen Animationsloops zu überblenden. Das Resultat sind elegante Bewegungen mit flüssigen Übergängen. Im Vergleich dazu wirken auf Sprite Sheet basierende Animationen holprig und altbacken, das kann natürlich zuweilen als kultige Retro-Optik gewünscht sein.

Favorit aus der Trickkiste: 2.5D, auch bekannt als Fake 3D

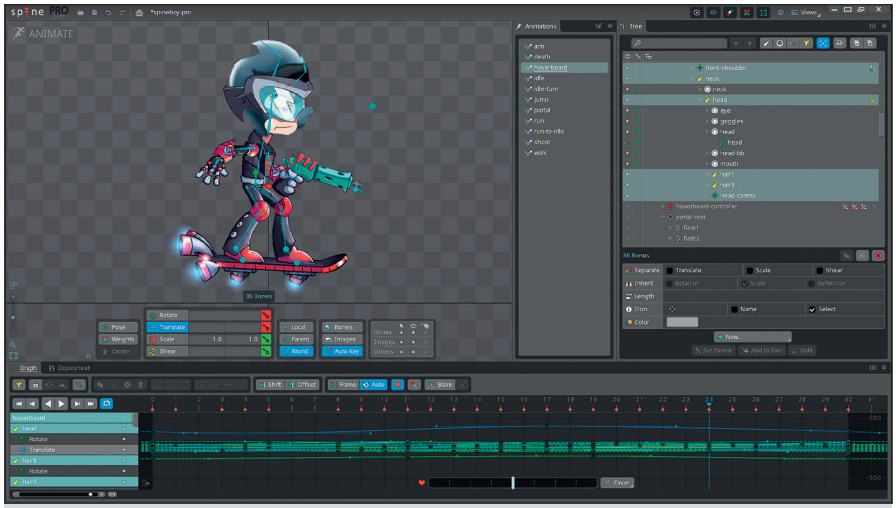
Bei animierten 2D-Illustrationen sind 3D-Effekte wie Head/Body Turns Teil des Standardrepertoires und in der Regel arbeitsintensiv, aber kein Problem, solange es sich um Vektorgrafik bzw. gleichförmige Farbflächen handelt, wenn man dem Rigging-Prozess mit seinen strengen Regeln folgt. Texturierte Flächen und Verläufe sind dabei notorisch schwer zu deformieren, ohne dass hässliche Texturverzerrungen auftreten. In diesem Bereich hat Spine die Nase vorn, denn mit den Spine Tools lassen sich Image-Segmente in Meshes verwandeln, die wiederum präzise mit Vertex Weights, Transform Constraints und Free Form Deformation (FFD) gesteuert werden können. Als Rigging Tools stehen umfangreiche IK-Möglichkeiten zur Verfügung, die zusammen mit dem Path-Werkzeug erstaunlich unorthodox eingesetzt werden können. Komplexe Abläufe wie ein Ballon, der sich aufbläst und dann mit einer Gondel abhebt, lassen sich über diese Methoden aufbauen. Spine erlaubt es, recht anspruchsvolle Rigs zu bauen, z.B. ein Mesh via Bone Chain zu deformieren, während diese wiederum einem Pfad folgt, der seinerseits durch Control Bones animiert wird, wie z.B. bei Tentakeln, langen Haaren und Ähnlichem. Lösungen für Dialoge gibt es über das externe Tool Rhubarb Lip Sync. Um einmal gesetzte Rigs mit verschiedenen Charaktervarianten zu laden, gibt es in Spine die sogenannte Skins-Funktion, über die sich die unterschiedlichen Designs unkompliziert austauschen lassen. Das ist ein Beispiel, an dem man sieht, wie das Tool mit den professionellen Bedürfnissen von Entwickler:innen und Game Artists gewachsen ist. Was Spine besonders benutzerfreundlich macht, ist die klare und eindeutige Benutzerführung mit präzisen und intuitiven Werkzeugen, die sich einfach angenehm anfühlen.

Favorit ist aber auch das Offset Tool in der Timeline, mit dem sich Keyframes innerhalb eines Loops versetzen lassen, was dann durch Verzögerung die Illusion von physikalisch trägem Verhalten imitiert. Offset ist wohl die am häufigsten zum Einsatz kommende Technik, um wallendes Haar, flatternde Mäntel und Derartiges zu animieren. Häufig sind es in Games ja gerade die ganz subtilen Idle-Animationsdetails, die eine vereinnahmende Atmosphäre zaubern, wenn alles geschmeidig wippt und minimal, aber organisch lebendig wirkt. Unter dem Hashtag #madewithspine lassen sich viele knuffige Werke auf Twitter und Instagram zur Inspiration bestaunen, die diese Prinzipien opulent veranschaulichen. Esoteric Software hat zudem Spine mit Demoprojekten (**esotericsoftware.com/spine-demos**) und Tutorials gut dokumentiert. Anhand der Beispiele lassen sich insbesondere die komplexeren Features mit Constraints und Pfaden anschaulich austesten. Bei der Einbindung in Unreal-Game-Mechaniken ist die Arbeit mit Spine-Animationen erfreulich geradlinig: Animationen, die vor dem Export in Spine via Animation Preview mit verschiedenen Tracks zur Überblendung geprüft wurden, können nun ebenfalls direkt in Unreal als unterschiedliche Tracks angesprochen werden und, zum Beispiel im Fall eines Wimpernschlags, dynamisch und additiv hinzugefügt werden. Genauso lassen sich nun die vor dem Export als Event Track angelegten Schritte abgreifen, indem sie via Event Dispatcher die entsprechenden Sound-Dateien ansprechen.

Im Spine-Animation-Preview-Fenster können unterschiedlich gewichtete Tracks miteinander kombiniert werden, damit man vorab testen kann, wie die Überblendungen in der Game Engine aussehen werden.



Spine bietet umfangreiche Rigging-Tools, um zum Beispiel auch mechanisches Verhalten zu imitieren. In diesem Fall die Ketten eines Panzers auf unebener Oberfläche via Bone Chains, Pfaden und Constraints.



Der Spine Animation Graph bietet Tools wie zum Beispiel das Offset-Tool.

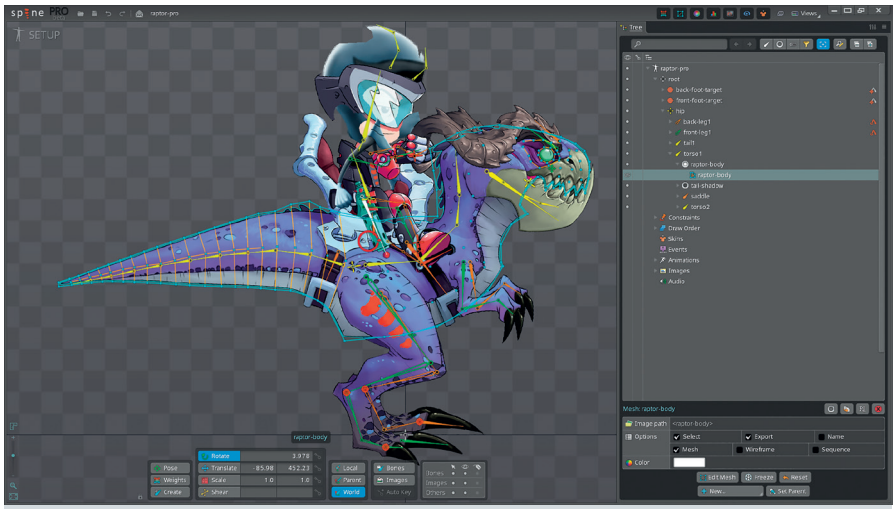
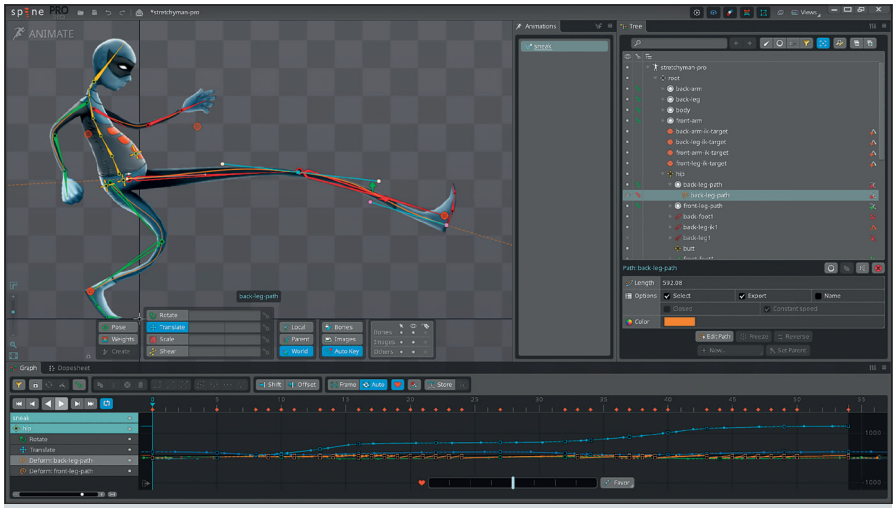


Image-Bereiche können als Mesh definiert und mit Free Form Deformation (FFD) modifiziert werden. Bei Export wird dann automatisch ein platzsparender Image-Atlas erstellt.



Pfade in Verbindung mit Bone Chains und Constraints eröffnen interessante Rigging-Möglichkeiten für dehnbare Gliedmaßen.

Beispielprojekt mit Ziel Unreal

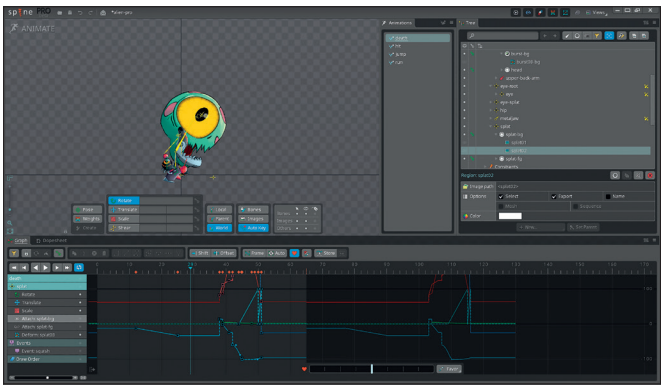
Für ein Beispielprojekt wurde als Zielplattform Unreal 4.27.3 ausgewählt. Die erste Frage, die eventuell in diesem Zusammenhang aufpoppen könnte, ist: Kann das eventuell Overkill sein, eine kleine 2D-Anwendung mit der Echtzeit-3D-Plattform Unreal zu bearbeiten? Ist das nicht wie mit Interkontinentalraketen auf Kleinwildjagd zu gehen? Die Antwort darauf kann nur ein getrostes Nein sein, denn zwar beherrschen Unity und Konsorten den 2D-Game-Entwicklungsbereich, trotzdem kann es gute Gründe geben, es trotzdem mit Unreal zu machen, z.B. wegen der Effizienz des Blueprint-Visual-Skriptingsystems, das die Entwicklungsarbeit erleichtert, oder wenn, wie in diesem Beispiel ‚ein 2D-Charakter mit Unreal Features und 3D-Environments kombiniert werden soll.

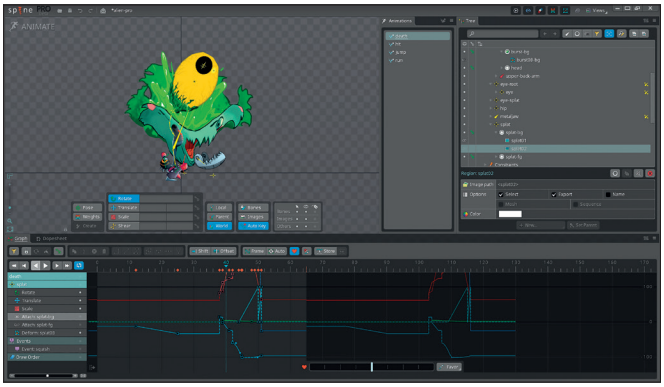
Unreal hat, nebenbei bemerkt, auch sein eigenes 2D-Tool mit dem Namen Paper 2D, was jedoch stiefmütterlich vor sich hin vegetiert, aber es gibt auch im Unreal Marketplace einige Game Kits, wie die Pixel 2D Engine for Unreal, die um einiges besser ausgestattet sind, jedoch in diesem Fall auf Sprite-Basis funktioniert. Davon abgesehen kann es auch andere Gründe geben, 2D-Spine-Animationen in ein Unreal-Projekt zu integrieren, zum Beispiel um Image-Effekte steuern zu können, die in bestimmten Fällen anders nicht machbar sind, beispielsweise fürs UI oder weil Zielplattformen oder andere Gründe es nicht zulassen. Ein Beispiel wären Regentropfen am Fenster, die auf Events reagieren. Spine bietet zum Glück volle Unterstützung für Unreal UMG (Unreal Motion Graphics) Widgets, sodass sich hier tolle Möglichkeiten für UI-Designer:innen bieten, um komplexe Interaktionen zu gestalten.

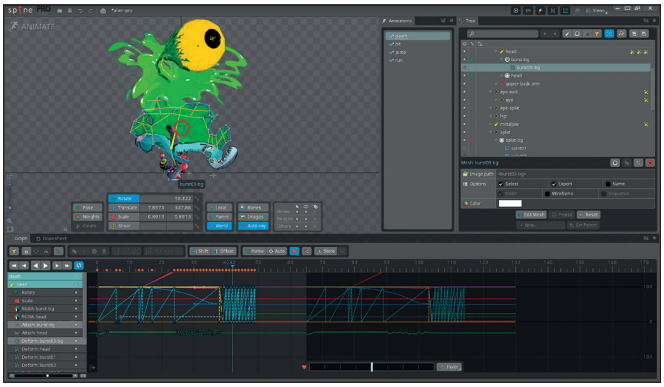
Die Hauptkompetenz von Spine ist jedoch Character Animation: In unserem Beispiel dreht es sich um den Oberkörper eines Orakels, der an verschiedenen Stellen in einem Dungeon erscheinen soll, um den User:innen rätselhafte Botschaften mit auf den Weg zu geben.

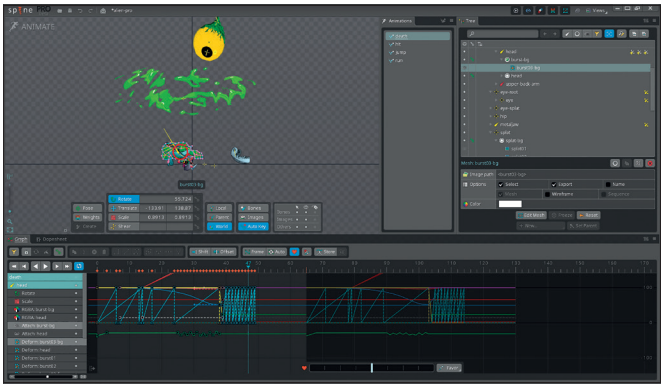
Vorbereitung

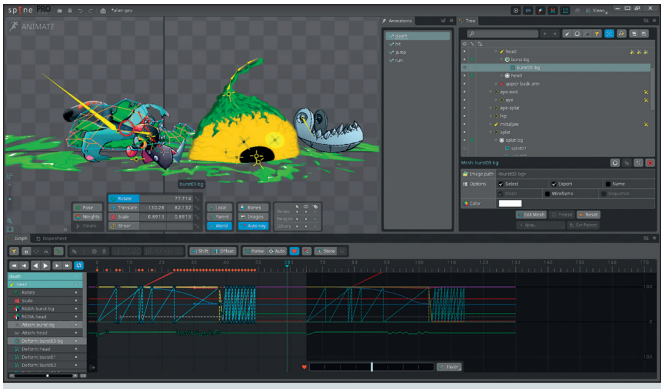
Um das Spine-Plug-in in Unreal nutzen zu können, muss das Projekt ein C++-Projekt sein, was wiederum voraussetzt, dass Visual Studio mit der Unreal-Integration auf dem Rechner läuft. Nachdem die Spine-Plug-ins in die dafür vorgesehenen UE4-Projektordner gelegt wurden, kann es losgehen. Für Blueprint-Puristen, die sich nun Sorgen machen, kann getrost Entwarnung gegeben werden: Trotz C++-Projekt kann genauso wie vorher mit Blueprints gearbeitet werden, ohne sich irgendwelchen Code anschauen zu müssen. Das Spine-Plug-in verlangt einfach



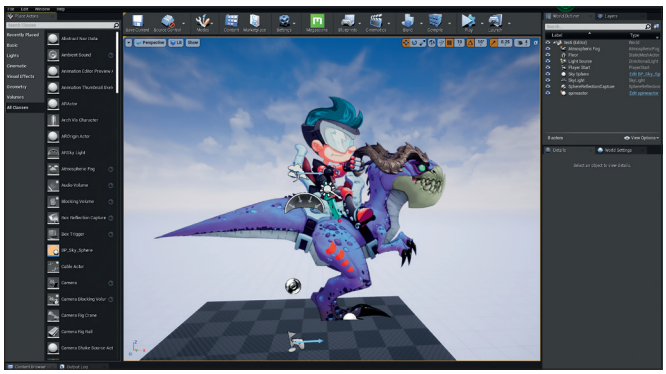


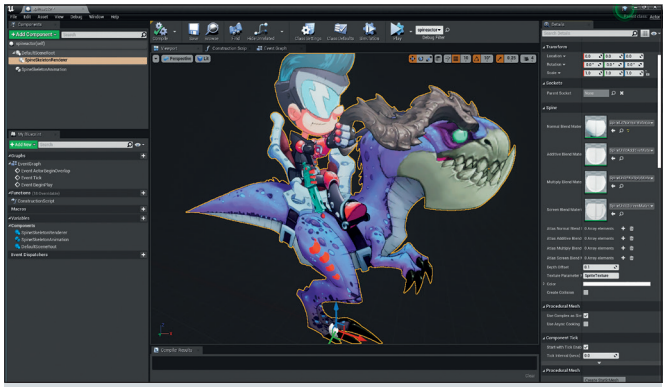




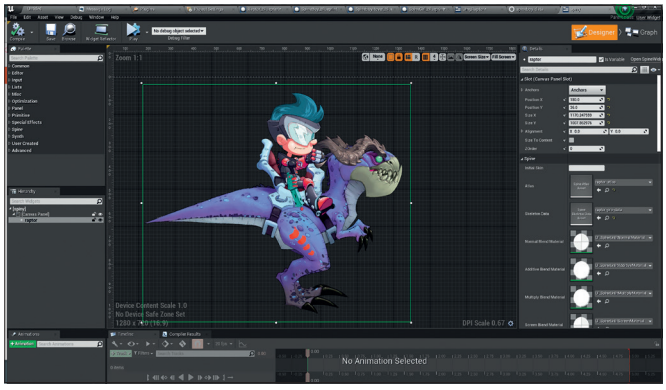


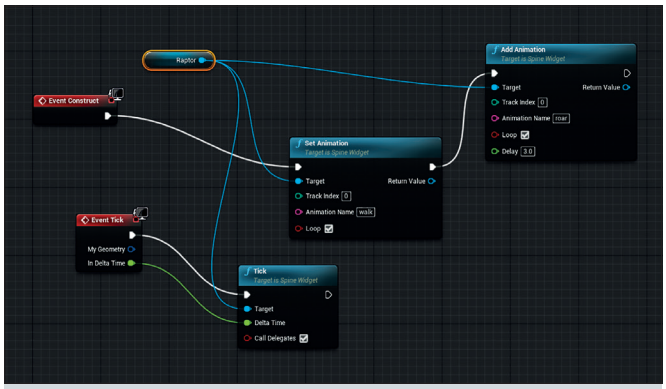
Für ungewöhnliche Animationsabfolgen steht auch immer die klassische Frame-by-Frame-Methode bereit, bei der die Image Sprites für den entsprechenden Teil geswitcht werden.



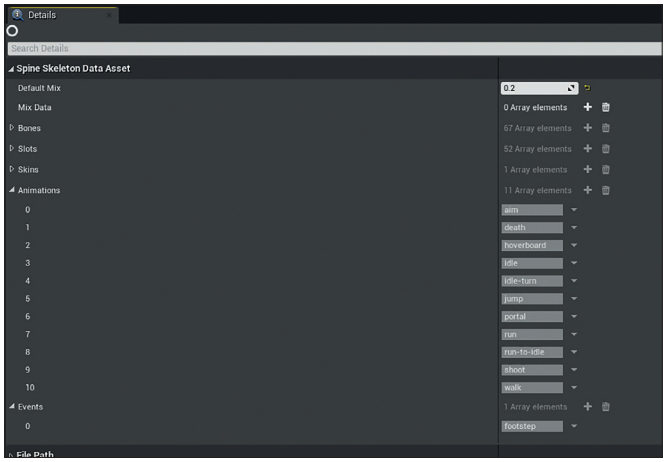


Mit dem Spine-Plug-in lässt sich die Spine-Raptor-Demo problemlos in Unreal importieren. In einer Unreal Actor Class benötigt es dazu den Spine Skeleton Animation Component und den Spine Skeleton Renderer Component, mit dem es auch möglich ist, die Base-Materialien von unlit zu lit zu ändern. Im Actor Event Graph ruft man die Animationen dann via dem Set Animation und Add Animation Node auf.

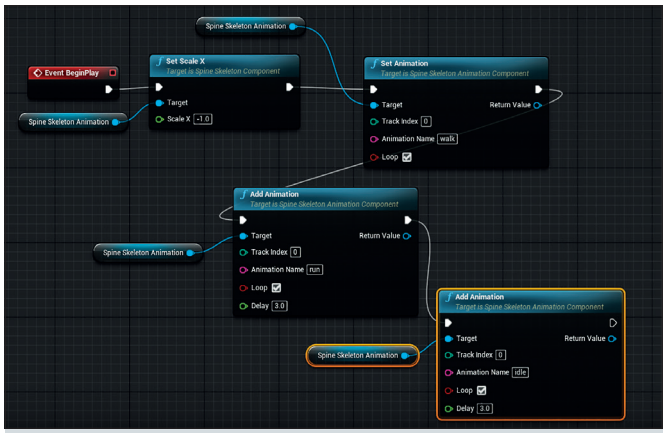




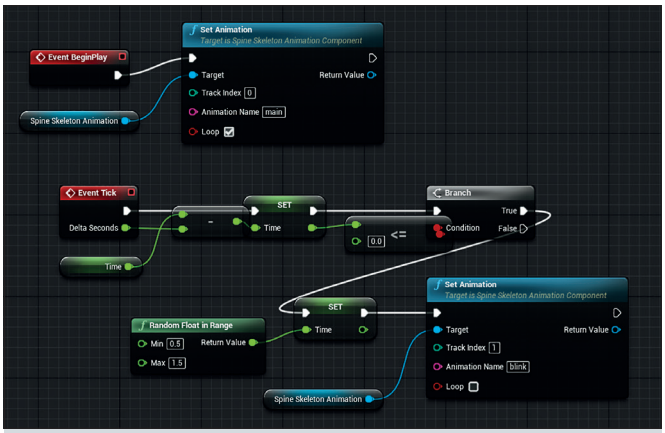
Alternativ lässt sich eine Spine-Animation auch in einem Unreal Widget positionieren. In diesem Fall wird die Animation zunächst über ein Event Construct sowie via Event Tick aufgerufen, um dann als Set Animation und Add Animation Nodes hinzugefügt zu werden.



Zugriff auf sämtliche Spine Features, die im Spine Skeleton Data Asset angezeigt werden, ist in Unreal via Blueprints möglich.



Ein typischer Ablauf einer Unreal-Spine-Animation: Nach Event Begin Play wird der Charakter auf der Achse geflippt, bewegt sich dann via Walk-Animation und dann im Anschluss mit jeweils 3 Sekunden Verzögerung in der Run und danach Idle-Bewegung.



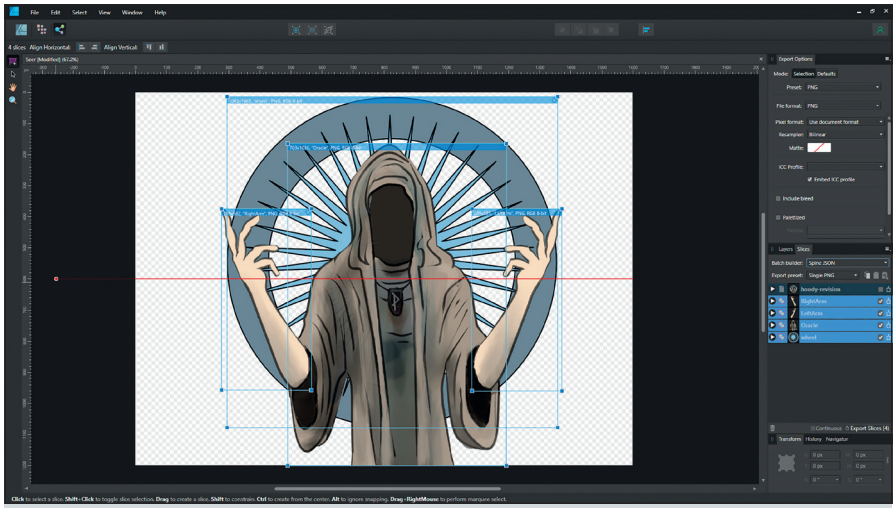
Die Vorteile von Spine zeigen sich in der Überblendung von Animationen, wie hier die Main (Idle) Animation auf Track 0, die endlos geloopt dann via Tick mit einer Blink-Animation auf Track 1 additiv überlagert wird, in dem ein zufälliger Moment in einer Zeitspanne errechnet wurde.

Die zu animierende Illustration entstand in Rebelle 5.

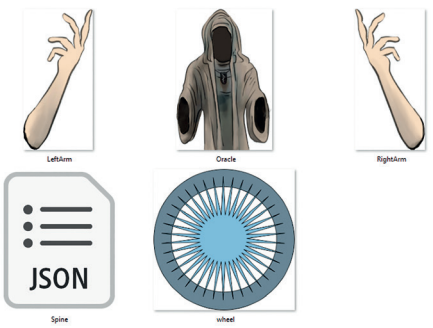
ein C++-Projekt, damit ist das aber auch schon alles in der Richtung. Ist die Spine-Integration einmal eingerichtet, lassen sich Spine-Dateien problemlos öffnen und via Component in einem Actor oder direkt in einem UE4-Widget positionieren und via Blueprint ansprechen.

Art Workflow via Rebelle und Affinity Designer

In der Regel nimmt ein Spine-Charakter seinen Ursprung in einem Illustrationsprogramm. In diesem Fall ist es eine Illustration aus Escape Motions Rebelle 5, die in Affinity Designer fertiggestellt wurde. Glücklicherweise hat Affinity Designer die Spine-Integration bzw. Spine-Export-Features schon per Default mit eingebaut. Wie bei 2D-Anima-

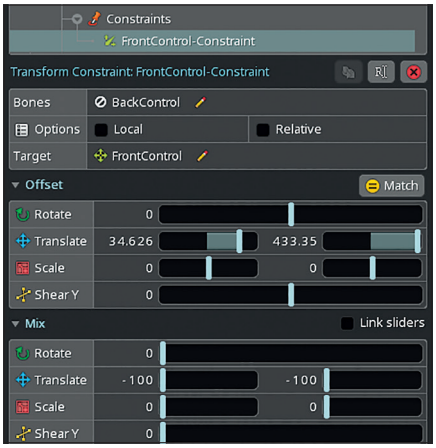


In Affinity Designer wird die Illustration ergänzt und als Slices via Batch Builder Preset Spine JSON exportiert.



Die exportierten Bildelemente und die JSON-Datei im Zielordner

tionen üblich, sind die Layer mit besonderer Berücksichtigung der Deformationsbereiche getrennt. In dem vorliegenden Beispiel sind es lediglich vier Ebenen, da die Bewegung nicht mehr erfordert. In Affinity kann man jetzt einfach über die Export Persona alle Layer auswählen, per Rechtsklickmenü die Slice-Funktion aktivieren und daraufhin via dem Export Batch Builder Preset Spine JSON exportieren. Beim Export erstellt Affinity die erforderlichen Daten, d.h. die Image-Segmente plus die Haupt-JSON-Datei mit den genauen Angaben über die Position der Layer. Öffnet man nun ein neues Spine-Projekt, reicht es einfach, mit Drag-and-drop die JSON-Datei in das Fenster zu ziehen und voilà, alle Image-Teile setzen sich automatisch an die richtige Stelle. Man muss dazusagen, dass der Affinity-Exporter wirklich nur die Basisfunktionen liefert, was in der Regel vollkommen ausreicht. Wer jedoch mit größeren Produktionen zu tun hat, die ein paar Extratricks benötigen, sollte sich das Spine-Photoshop-Plug-in anschauen. Dieses bietet unter anderem Tags und Padding-Optionen und lässt sich als Plug-in im Photoshop-Scripts-Ordner installieren. Das Plug-in funktioniert glücklicherweise auch für die Pre-Subscription-Version CS2, die ja noch sehr häufig verwendet wird. Die aktuelle Liste der Applikation mit Spine-Plug-in-Support wird zudem auch auf Github aktualisiert: **github.com/EsotericSoftware/spine-scripts**.



Die Bone-Hierarchie mit Scene Root, zusätzlichen Ärmel-Bones und Front Control Constraint für den Kopf. Der Front Control Constraint ist ein Transform Constraint, ausgehend von dem Back Control Bone mit dem Ziel Front Control Bone und einem Translate-Wert von -100.

In Spine, das User:innen klassisch zwischen Setup und Animationsfenster hin- und herspringen lässt, geht es nun an den ebenfalls typischen Workflow von Skeleton Setup zu Keyframing, bis das fertige Resultat schließlich als Unreal-kompatibles Format exportiert werden kann. Optional sind auch eine Reihe anderer Exportformate möglich, zum Beispiel PNG-Sequenz oder GIF, die sich gut als Preview eignen.

Feature Highlights

Neben den intuitiven Tools, um Bones aufzuziehen und IK einzurichten, sind es wie erwähnt insbesondere die hervorragenden Wichtungswerkzeuge, die es erlauben, jeden einzelnen Mesh-Vertex mit verschiedenen Bone-Einflüssen zu gewichten, wie man es aus 3D-Anwendungen kennt. In der Regel lohnt es sich, manuelle FFD Edges aufzuziehen, nachdem das Mesh automatisch generiert wurde. Die gelb hervorgehobenen Linien werden in der Regel entlang von Image Features gesetzt, um Verzerrungszonen zu bestimmen. Diese lassen sich dann auch direkt im Animationseditor kontrollieren, sodass Vertex-Level-Animationen problemlos möglich sind und daher fast unendliche Optimierungsmöglichkeiten für das Endresultat eröffnen.

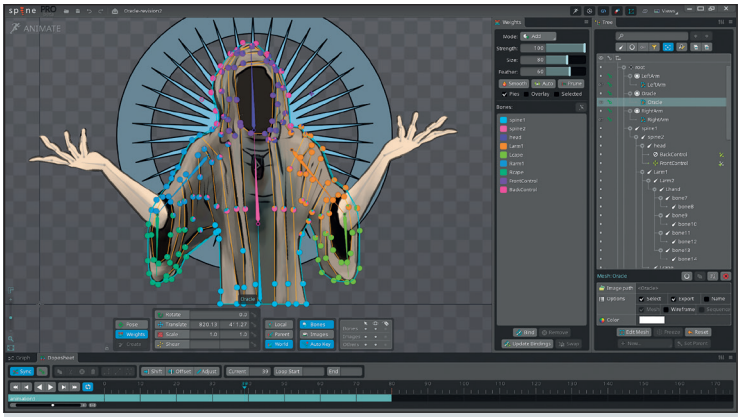
In der vorliegenden Kutte sind die Ärmel jeweils mit einem zusätzlichen Bone versehen, damit sie nachschwingen können. Daher muss dieser Bereich besonders sorgfältig gewichtet werden, um realistisch zu wirken. Unerwünschte Verzerrungen können problemlos mit dem Weight Painting Tool eliminiert werden.

Besonders hervorzuheben ist die vielseitig einsetzbare Transform-Constraint-Funktion. Häufig, wie auch im vorliegenden Fall, werden dabei zwei Headbone Controller unter einem Head Bone Parent mit negativem Wert verknüpft. Das führt dazu, dass die Achsenbewegung des ersten Controllers im zweiten Controller entgegengesetzt ausgeführt wird, was nun dazu benutzt werden kann, eine perspektivische Bewegung vorzutäuschen. Mit diesem einfachen Trick lassen sich nun 3D-Bewegungen konstruieren, aber auch andere Effekte wie Background-Layer-Parallaxen, indem die Image-Mesh-Vertex-Wichtungen an der Bewegungsachse orientiert sind.

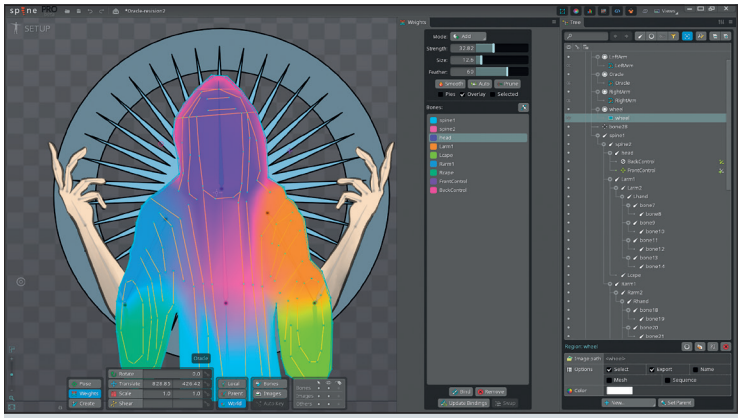
Im vorliegenden Fall soll lediglich eine leichte Kopfbewegung in 3D animiert werden. Daher kontrolliert der Head Controller die Vertexinformationen im Zentrum der Kapuze, während der Reverse Backhead Controller die Punkte am äußeren Rand bewegt und in der Mitte der Head Parent Bone dominiert. Durch diese dreifache Staffelung mit vorsichtig gewichteten Übergängen ist nun eine perspektivische Kopfbewegung möglich und kann nun mit Keyframes animiert werden.

In der Animationsansicht lassen sich Audio- oder auch Event-Tracks einrichten, die dann später in der Engine abgegriffen werden können. Auch ist es möglich, einen Animation Blending Preview in Spine zu machen. Denn letztendlich ist es ja maßgeblich für die fertiggestellte Animation, wie gut sich die einzelnen Loops in der Engine überblenden lassen. Falls etwas nicht ganz rund aussieht, fixt man es lieber gleich direkt in Spine, nachdem man es über die Blending-Track-Preview-Optionen durchgetestet hat.

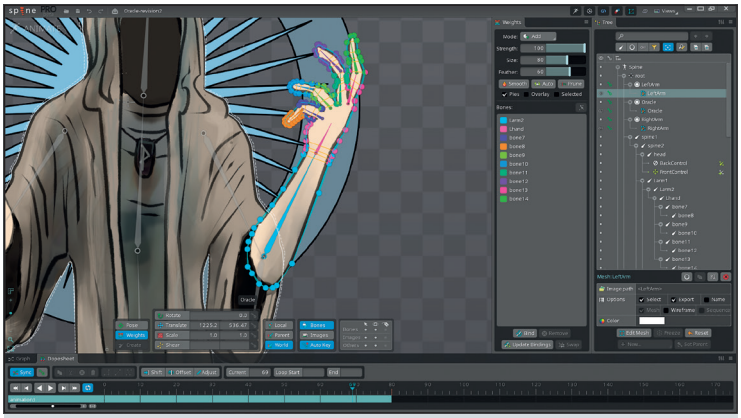
Im Edit Mesh Mode können Deformationsbereiche entlang von Image Features definiert werden, indem Free Form Deformation (FFD) Edges markiert werden. Diese stehen dann zur Vertex-Animation in der Timeline zur Verfügung und eröffnen zusätzliche Möglichkeiten.



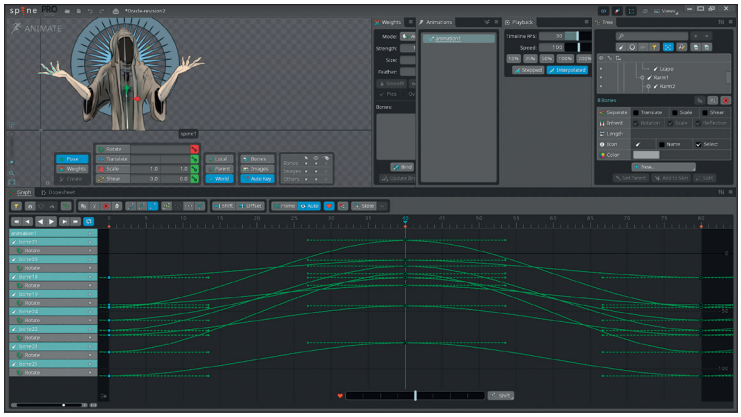
Die übersichtliche Weights-Anzeige im Pies-Modus, wobei der farbcodierte Bone-Einfluss auf den jeweiligen Vertex in einem Mini-Pie-Chart angezeigt wird.



Alternativ lässt sich der Bone-Einfluss auch als durchgängiger Overlay anzeigen, das ermöglicht einen guten Überblick über die Wichtungsbereiche.



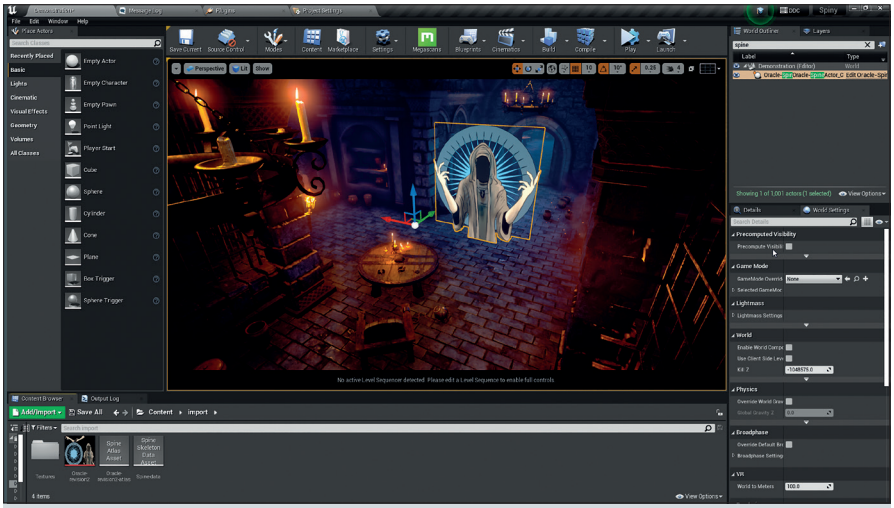
Die Finger-Bone-Gewichtung lässt sich problemlos nachjustieren, um unerwünschte Verzerrungen zu vermeiden.



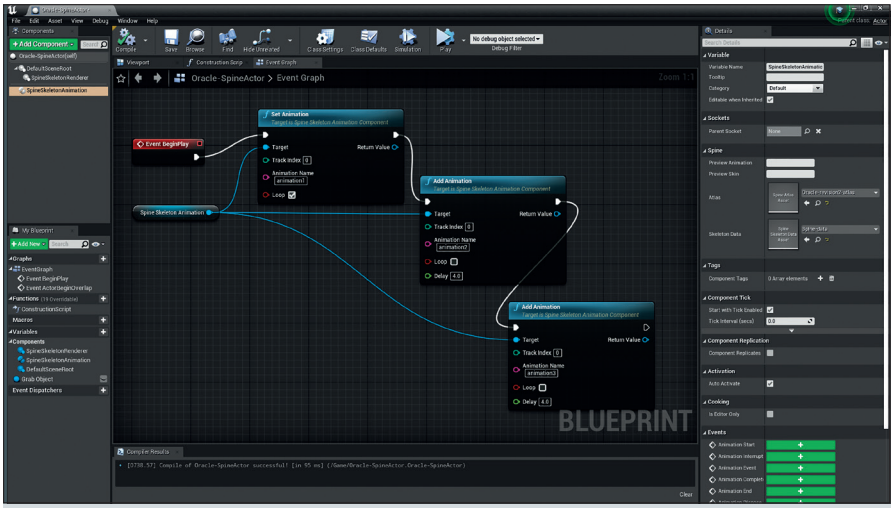
Die klassischen Keyframes Tools mit Dope Sheet, Graph und Curve Tools erlauben schnelles Arbeiten und Optimieren der Loops.

Showtime in Unreal

In Unreal importiert, lassen sich Spine-Assets via dem Spine Actor Component laden. Ein Preview-Feld in den Spine Skeleton Animation Component Details erlaubt es, Animationen zur Preview auszuwählen. Zu den verfügbaren Spine Actor Components zählen auch der Skeleton Driver Component und der Skeleton Follower Component, über welche sich Bone-Informationen direkt ansprechen lassen, zum Beispiel um Scene Assets einem bestimmten Bone folgen zu lassen. Via Blueprints lassen sich nun im Unreal Editor Skins und auch Attachments laden. Alle Spine Features sind laut Esoteric Software in Unreal via Blueprint freigelegt. Animationen lassen sich problemlos in Tracks laden und entladen, Geschwindigkeit ist kontrollierbar, ebenso sind Events über den Animation State abgreifbar. Skeleton-Elemente sind via Blueprint ansprechbar, zum Beispiel über einen Unreal Event Dispatcher. Die mitgelieferten Spine-Materialien, auf denen die Grafik läuft, sind per Grundeinstellung als Unlit definiert, denn in der Regel möchte man Spine-Animationen von der Beeinflussung des Umgebungslichts freihalten. Falls man jedoch die gesetzten Licht- und Schatteneinstellungen im Material einfangen möchte, stellt man es auf die Lit-Materialien von Spine um, die ebenfalls bereitliegen. Jedoch ist die Arbeit mit dem



Die in Unreal importierten Spine-Dateien werden als Components in einer neuen Actor Class angesprochen. Der neue Actor lässt sich dann frei im 3D-Environment platzieren.



Der Spine Actor mit seinen beiden Haupt-Components, Spine Skeleton Renderer und Spine Skeleton Animation, sowie dem Event Graph, um die Animationen zu triggern.

Spine Actor Instances können nun an die gewünschten Bereiche im 3D-Environment positioniert werden, um dort aufgerufen zu werden, wenn der/die User:in sie aktiviert.

Lit-Material von Spine mit Vorsicht zu genießen, da sich gerne mal Normal-Fehler beim Rendern einschleichen, z.B. wenn bestimmte Mesh Triangles unglücklich gesetzt sind.

Im vorliegenden Beispielprojekt soll nun der gestikulierende Finsterling an verschiedenen Positionen in dem Dungeon-Environment eingeblendet werden, woraufhin jeweils verschiedene Kombinationen der Gestik-Animationen gemixt werden. Wir testen dies mit Event Begin Play, was dann später jeweils auf Custom Event Trigger umgestellt wird.

Fazit

Insgesamt macht die Spine-Unreal-Integration einen sehr runden Eindruck und erstellte Animationen laufen wie erhofft geschmeidig, in hoher Qualität und sind voll via Blueprint ansprechbar. Bleibt zu hoffen, dass Esoteric Software in Zukunft noch mit zusätzlichen Unreal Features nachlegt. Findige User:innen haben bereits den Feature-Umfang auf eigene Faust erweitert, mit zum Beispiel Physics, Ragdoll Simulation, Root Motion, Terrain IK und Sequencer-Support. Offizieller Sequencer-Support alleine wäre schon ein Gamechanger, denn damit würde die Unreal-Spine-Integration mit einem Schlag sofort auch eine interessante Ergänzung im Virtual-Studio-Bereich mit allen Möglichkeiten, was Broadcast und TV-Produktionen angeht. Unreal wäre in diesem Sinne eine perfekte Ergänzung, da es ja direkt in Spine keine Kamera gibt. Aber das Wichtigste hatte nun einmal Vorrang, und das ist die Unreal-5-Unterstützung. UE5-Support gibt es nun endlich seit Mitte Mai für Spine 4.0 und alle nachfolgenden Beta Runtimes. **› ei**

Cornel Hillmann ist CG-Artist und XR-Designer und seit über 20 Jahren im Bereich Media & Entertainment, Visualisierung und Design tätig. Er arbeitete unter anderem mit Marken wie Panasonic, Jaguar und Razor, führte 3D-Design-Kurse an der Limkokwing University und leitet Masterclasses für Immersive Media Postproduction, Advanced 3D-, VR- und Media-Design. Außerdem ist er Autor der Bücher „Unreal for Mobile and Standalone VR“ und „UX for XR“ des New Yorker Apress Verlages.

Spine Professional

Preis: 349 US-Dollar

▷ esotericsoftware.com