Bachelor's thesis

Information Technology

Game Technology

2016

Jenni Lehtonen

FROM 2D-SPRITE TO SKELETAL ANIMATIONS

– Boosting the performance of a mobile application

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Game Technology

2016 | 37

Principal Lecturer: Mika Luimula, Adj.Prof.

Jenni Lehtonen

FROM 2D-SPRITE TO SKELETAL ANIMATIONS

This thesis focuses on 2D animation in video games and programs for creating the animations. The purpose was to find the most efficient way to make 2D animations for games, and the best program for making them.

At first, the thesis presents different 2D animation methods which have been used in games during their history. For this purpose, the animation methods and a number of 2D games from different decades were researched.

The thesis then presents different 2D animation programs and their functionalities. The programs were selected based on personal experience, animation specialist's recommendations, and customer's preference. Game engines as 2D animation platforms were ruled out and only programs specially made for 2D animation were compared.

The programs are compared in order to determine which program best suits for 2D game animations. The comparison is based on license cost and different tools and functionalities that speed up the animator's work, make importing to game engines easier, and help the communication between artists and programmers. Two animation specialists were interviewed in order to gain an even better understanding of the needs for 2D animation.

Based on the comparison, Spine 2D was selected for closer inspection and few case study animations were made for the Turku Castle application, a mobile application being developed by Turku Game Lab. It is part of Fast Wow Effects Boosting SME Business by Tekes and is done in co-operation with Turku Muesum Centre. Spine 2D's most useful tools are presented and viewed from a game developer's perspective.

Finally, the thesis describes the process of animating with Spine. The animations had previously been made as sprite sheets and were then recreated as skeletal animations in Spine. In conclusion, the differences of 2D animation programs and reasons to move from Sprite animation to skeletal animation in 2D games are reflected upon.

KEYWORDS:

Digital games, Flash, image deformation, mobile applications, game graphics, animation, 2D

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ
TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Peliteknologia

2016 | 37

Yliopettaja: Mika Luimula, dosentti

Jenni Lehtonen

2D-SPRITE-ANIMAATIOISTA LUURANKOANIMAATIOIHIN

Tämän tutkimuksen tavoitteena oli löytää paras 2D-animointitapa pelianimaatioiden luontiin mobiilialustalle sekä selvittää, mikä 2D-animaatio-ohjelma sopii parhaiten 2D-pelianimaatioiden tekoon. Parhaaksi todetulla ohjelmalla toteutettiin animaatioita Turku Game Labissa kehitteillä olevaan Turun Linna -mobiilisovellukseen. Sovellus on osa Tekesin Wow-hanketta ja toteutetaan yhteistössä Turun museokeskuksen kanssa.

Tutkimus toteutettiin kuvailevana tutkimuksena, joka perehtyy ensin pelianimaatioissa yleisesti käytettyihin 2D-animaatiotekniikoihin. Saadun tiedon pohjalta 2D-animaatio-ohjelmien tarjoamat toiminnot asetettiin tärkeysjärjestykseen pelianimaatiota ajatellen. Mobiililaitteiden asettamat rajoitukset otettiin myös huomioon. Vertailussa olleet ohjelmat valikoitiuvat animaatioasiantuntijoiden suosituksiin sekä asiakkaan toiveisiin perustuen. Pelimoottorit 2D-animaatioalustoina päätettiin jättää vertailun ulkopuolelle ja keskittyä vain 2D-animointia varten tehtyihin ohjelmiin.

Ohjelmien vertailuperusteina toimivat lisenssien hinnat ja kunkin ohjelman tarjoamat työkalut animaattorin työn nopeuttamiseksi, pelimoottoriin siirtämisen helpottamiseksi sekä animaattorin ja ohjelmoijan välisen kommunikaation parantamiseksi. Kahta animaatioasiantuntijaa haastateltiin, jotta voitiin muodostaa parempi käsitys pelianimaation ja etenkin animaattoreiden tarpeista.

Vertailun tuloksena Spine 2D valikoitui parhaaksi 2D-animointiohjelmaksi. Tulokseen eniten vaikuttaneet ominaisuudet olivat helppo käytettävyys, laaja valikoima työkaluja sekä alhainen lisenssihinta. Spine 2D -ohjelmalla toteutettiin luurankoanimaatioita Turun Linna -applikaatioon ja työn tuloksena mobiilisovelluksen kuormittavuus pieneni huomattavasti, kun sprite-animaatiot korvattiin luurankoanimaatioilla. Animaatioista tuli myös huomattavasti sulavampia.

Tutkimuksen lopuksi pohditaan 2D-ohjelmien eroja sekä luurankoanimaation etuja sprite-animaatioihin verrattuna 2D-mobiilipelikehityksen näkökulmasta. Parhaan animointitavan valinta on hyvin tapakohtaista, luurankoanimaatioilla saadaan aikaan sulavampia ja tiedostokooltaan pienempiä animaatioita, kun taas sprite-animaatioilla on helpompi saavuttaa luonnollisemman näköistä animaatiota. Tämän tutkimuksen tulokset pätevät parhaiten juuri mobiilisovelluksiin, ja lisätutkimusta tarvitaan, jos halutaan saada kaikkia pelialustoja koskevia tuloksia.

ASIASANAT:

Digitaaliset pelit, Flash, kuva manipulaatio, mobiilisovellukset, peligrafiikka, animaatio, kaksiulotteisuus.

CONTENT

LIST OF ABBREVIATIONS AND TERMS **6**
1 INTRODUCTION **8**
2 2D ANIMATION IN GAMES **10**
2.1 Rotoscoping 10
2.2 Cutout animation 11
2.3 Digital image deformation 12
2.4 Sprite sheet animation 12
2.5 Skeletal Animation 13
2.6 Keyframe animation 14
3 PROGRAMS FOR 2D GAME ANIMATION **16**
3.1 Spine 16
3.2 Toon Boom and Harmony 16
3.3 Adobe Flash and After Effects 17
3.4 Spriter 17
4 COMPARISON OF ANIMATION PROGRAMS FOR GA-ME ANIMATION **19**
5 SPINE UNDER CLOSE INSPECTION **23**
5.1 Skins 23
5.2 Events 24
5.3 Weights 24
5.4 Inverse Kinematics 25
6 CONVERTING SPRITE ANIMATIONS TO SPINE ANIMATIONS **27**
6.1 Required elements 27
6.2 Importing elements to spine 28
6.3 Swapping images versus animating with mesh tool 29
6.4 Fine tuning the animation 30
6.5 Exporting Unity compatible animation files 31
7 CONLUSIONS **33**
REFERENCES **36**

APPENDICES

Appendix 1. Interview questions and answers

PICTURES

Picture 1. Patent drawing for the original rotoscope (US Patent n° 1,242,674)............ 11

Picture 2. Example of a sprite sheet used on Debate Club, a game by a group of students from Turku University of Applied Sciences. Sprite sheet by Lehtonen, J and Mattila, N. 13

Picture 3. Example image of a 2D character with a skeleton from official Spine tutorials. 14

Picture 4. Example of an image map Spine creates automatically for the animation. Original painting: Juhana III, owned by Turku Museum Centre, painter unknown. Photograph: Martti Puhakka / Turku Museum Centre. 28

Picture 5. Texture Packer Settings window from Spine, with Unity compatible export settings. 32

TABLES

Table 1. Comparing the prices and features of different animation programs............. 20

LIST OF ABBREVIATIONS AND TERMS

2D	2 dimensional computer graphics.
3D	3 dimensional computer graphics.
SME	Small and medium-sized enterprises.
AAA	Classification for High quality games with high development budget and high expectations for success after release.
Bones	In 2D and 3D animation a skeleton can be built to control the animation, the skeleton consists of connected bones which can be moved to create different animations.
CC	CC stands for Creative Cloud and is used for all the Adobe software available through Adobe Creative Cloud.
Game engine	A software framework designed to make developing games more efficient by offering many core functions already programmed in the engine.
Inverse kinematics	A method used to make animating certain movements more efficient, used especially when animating with bones.
Keyframe	An image that marks either the start or an end of a smooth transition to animation.
Mesh	In 3D animation, the term mesh means the model created and animated by the artist.
Runtime	When talking about animation, runtimes are used by game engines to implement aminations and other functions made in 2D animation programs.
Skinning	Skinning is a term used for the process of wrapping a 3D mesh around the skeleton or binding 2D image to the bones.
Sprite	Sprite is one image needed for a game animation, sprite sheet consists of all the sprites needed for one or more game animations.

Weights	Weights in animation refer to the deformation of a 2D image or 3D object based on set values used to replicate the effects of gravity on the animation.

1 INTRODUCTION

During recent years, the game industry has grown rapidly. Games are accepted by ever increasing number of people and with the wide variety of tools for making games, more and more people want to test their own skills as game developers. With many cheap or free game engines on the market, making games is now easier than ever before. (Mao etc. 2010)

The quality of graphics might not be the most important part of a game, but it has always affected the popularity of a game. From early 2D games with massive character sprites to today's AAA quality 3D games, visually pleasing graphics have always attracted the audience (Brown 2014). With the rise of mobile devices, 2D games have gained some popularity again in this highly competitive industry. The second chapter of this work explores the different 2D animation methods that have been used for game development during the years.

Smooth gameplay and short loading times are two of the most important factors for any games success. The memory usage of a game, effects both of these and graphics are what usually take up most memory for games. (Pranatio & Kosala 2012.) Therefore, it is crucial to know what methods and tools to use for most efficient memory usage, without having to sacrifice the quality of graphics. What limited 2D games in the past were the size restrictions for graphics set by the limited memory of computers, and that is exactly what limits the mobile game development today. To release some of the memory space used by traditional sprite sheets, programs have been developed to decrease the size needed for graphic files without having to cut down the quality of graphics or animation.

Today the problem is not so much finding tools for creating dynamic 2D animation, but choosing from the vast number of programs on the market. In the third chapter, some of the most popular software for 2D animation are introduced and the fourth chapter compares the software to determine which of them best suits for 2D game animation. Two animation specialists from Turku Game Lab, Markus Norrgran, and Simo Ruotsalainen were interviewed for professional opinions. As a result of the comparison, Spine 2D was selected for completing few case study animations. Chapters five and six introduce some of Spines most unique tools and describe the experience and process of animating with Spine for the very first time.

The work done for this thesis was ordered by Turku Game Lab from Turku University of Applied Sciences and was done for a mobile application made for Turku Castle. The application is part of Fast Wow Effects Boosting SME Business by Tekes and is being implemented in co-operation with Turku Museum Centre. During the time of working on this thesis, the title of the application was “Turku Castle in Your Hand”. With it, the visitors can gain more knowledge of the castle and interact with some of the items found from there. The animations done in this thesis were made for some of the exhibited paintings in the castle museum. When the user looks at the paintings through the application, the animations are triggered using augmented reality technology and the paintings seem to come to life on the screen.

In conclusions, results are presented and reflected on from game developer’s perspective. Reflection is based on personal experience, research material, and specialists’ opinions.

2 2D ANIMATION IN GAMES

Animation has always been an important part of video games, the basic idea in almost every video game is that there is something moving on the screen and that movement creates the game. This movement requires some kind of animation, be it a white line moving on the screen like in early 70's Pong or complex fighting animations like in Street Fighter II (Pong 1972, Street Fighter 1991). The simple animations in games can be achieved by just moving around an image across the screen, but more complex animations, like walking usually requires at least one sprite sheet.

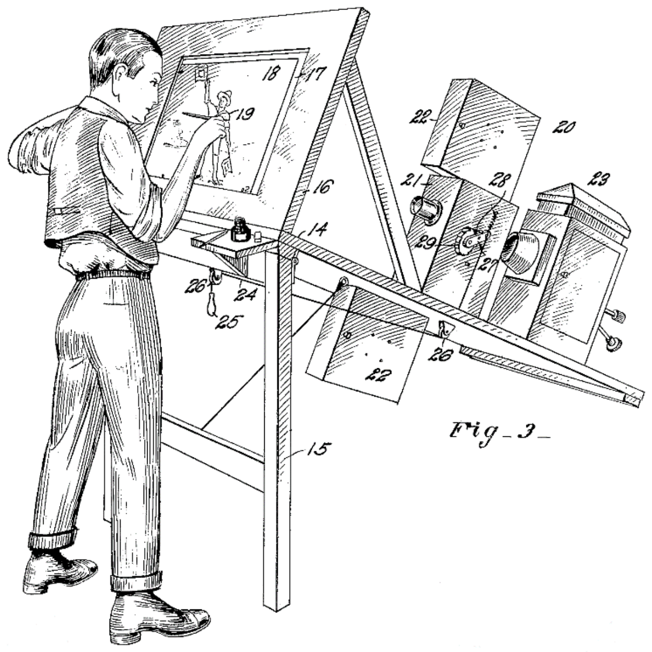
As 3D graphics started to develop further, 2D games were largely overshadowed by them and it is not until the recent years that 2D games have started to gain some ground in the mainstream games again. With successful indie games like Limbo and Super Meat Boy (Limbo 2010, McMillen 2010) and mobile game titles like Angry Birds, 2D games have gained some popularity again.

Making an animation for a video game is a very different process than making an animated movie. In movies the character's actions and camera directions are decided in advance and characters are animated accordingly. In video games the character is controlled by the player and animation needs to be prepared to accommodate each decision the player makes. The amount of choices the player has can be limited by the games functionality, but in most games many different movement animations are needed and they need to look good when blended together in a game engine. (Walther-Franks & Malaka 2014.)

A variety of animation techniques has been used for 2D game animation in the past, and sometimes even 3D graphics are used to create better looking 2D games. In this chapter, these techniques are briefly explained

2.1 Rotoscoping

Rotoscoping is an old animation technique, patented by Fleischer Studios in 1917 and used for animating real life like movement for cartoon characters, like Popeye the Sailor. This technique used a machine called Rotoscope which consisted of a camera behind a desk, projecting real life film footage onto a frosted glass. The projected images would be traced onto a paper by animators to create lifelike movement for the animations. This technique was used especially for studying the movement of the human body. (Bratt 2011, 1.)



Picture 1. Patent drawing for the original rotoscope (US Patent n° 1,242,674).

In game animation, rotoscoping has been used to get that life like movement for the characters. For example, all the original sprites for the game Prince of Persia were created by tracing from real life video (Mechner 1989, Brown 2014). With modern technology, it is possible to create 3D models of the desired objects or characters and then trace the models by drawing 2D images on top of them. This uses the same method as rotoscoping but is done digitally without the use of an actual rotoscope. Same can be done with any digital images, for example, digital photos or still images from a movie.

2.2 Cutout animation

Cutout animation is a technique that was first used and patented by Quirino Cristiani (Bendazzi, G, 1996). The animation consists of cut out images that are used individually or build to a puppet. These images are then moved to create movement. A good example of this animation technique is the VT series South

Park (Waytt 2010, 48-49).

Many animation software uses this same principle when animating a character. The character is cut into pieces which are then assembled in the animation program and moved to create an animation. In most programs this is used together with keyframe animation; the puppet is set only on few key positions and the program will create the movement between them to fasten the process. for more lifelike movement, the keys in between need to be altered as well.

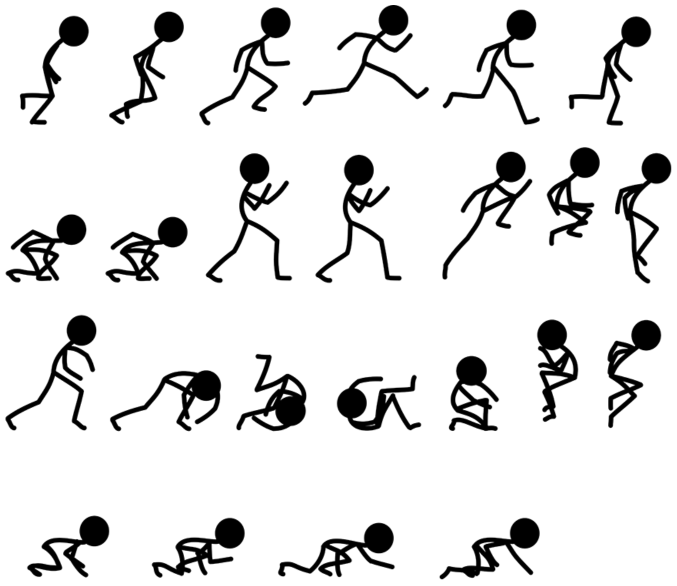
2.3 Digital image deformation

Digital image deformation is a method to manipulate 2D or 3D images to deform them from their original form. Most 2D image deformation methods seem to be based on Ruppert's algorithm for creating a triangulated mesh on top of a 2D image (Ruppert 1995). Some methods create an interactive mesh on top of the image automatically and the user can then set handle points to move certain parts of the mesh to deform the image. To give the used more power over the deformations, weights can be added to different parts of the mesh to affect the way they are deformed (Igarashi etc. 2005).

One way to use this method for character animation is to separate the character's limbs onto different layers and deform them separately. This way deforming one limb won't affect the rest of the image and images can cross over each other without problems. The use of layers for deformation often gives more natural results than trying to deform everything the same layer (Mota etc. 2011). Image deformation is often used together with some other animation methods to achieve more lifelike movements and avoid having to draw new frames for each slight change in the image. For example, slight changes in facial expression can be achieved with image deformation (Bui etc. 2009).

2.4 Sprite sheet animation

All the techniques above can be used to create sprite sheets for game animation. Sprite sheets consist of a set of images that are required to make the desired movement for characters or background items. The game is programmed to cut the sheet into separate images and then assign an order in which it will display the images to create the animation seen on the screen. In early 90’s sprite sheet animation was in its prime as the developing technology allowed bigger and more complex sprite sheets than ever before. (Brown 2014.)



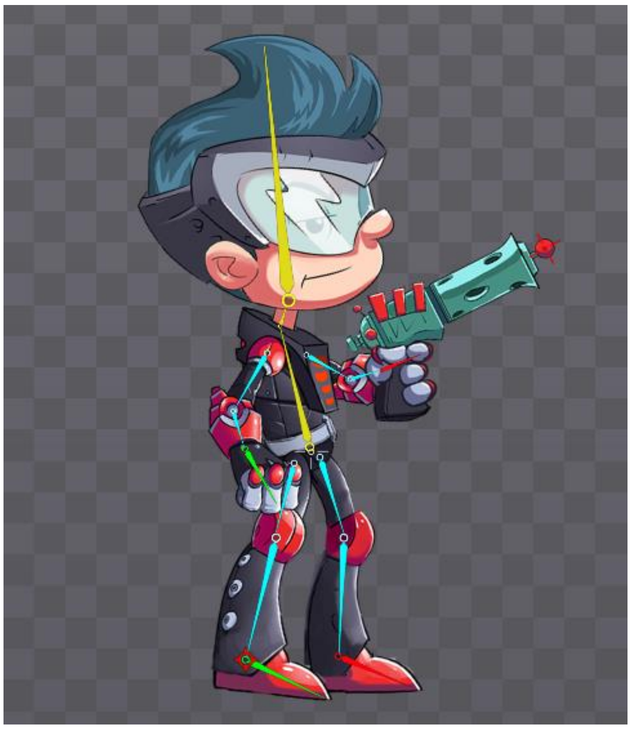
Picture 2. Example of a sprite sheet used on Debate Club, a game by a group of students from Turku University of Applied Sciences. Sprite sheet by Lehtonen, J and Mattila, N.

Today the use of sprite sheets is quite rare. Using these huge images takes up a lot of disc space and slows down the performance of the game. It is also very difficult to achieve a smooth looking animation with sprites unless you want to end up with hundreds of frames for one animation. However, sprite sheet animation can be used to recreate nostalgic 90’s feels for modern games.

2.5 Skeletal Animation

Skeletal animation is common when animating 3D models. First, a mesh is created for the 3D object and then a skeleton is built inside it to roughly match the mesh. The mesh is then wrapped around the skeleton by a method called skinning. Once this is done the skeleton can be set at different positions and the mesh will follow, setting the object on to the wanted positions. (Motwani, R. etc. 2008.)

To use a skeleton for animating in 2D, the image needs to be cut into pieces like in the cutout animation method. Instead of moving each picture individually, a simple stick figure like skeleton is created for the image and the separate images are bound on to different bones on the skeleton. After that, the skeleton can be animated and the images will follow much like in 3D animation.



Picture 3. Example image of a 2D character with a skeleton from official Spine tutorials.

2.6 Keyframe animation

Keyframe animation refers to both 2D and 3D animation created by an animation program and can be considered as the main concept of computer animation. (Izani etc. 2003.)

In 2D, this often resembles cutout animation as the image is cut into pieces and then animated. The animation relies on keys on a timeline, set to the start and end of the desired movement. The program creates all the frames required in between the keys by moving the images from first set place to the next to create a smooth animation.

When exported, the programs usually have the option to create a sprite sheet, some also offer the choice to export the files as a numerical data for skeletal animation. When executed, this creates a map of the used images and a code file which tells the game how to use the images to make the animations. The use of an image map saves a lot of space when compared to the traditional sprite sheets and is often used when animating for modern 2D games.

3 PROGRAMS FOR 2D GAME ANIMATION

There is a variety of programs to choose from for 2D animation but unfortunately, the scope of this work is not big enough to present all of them. The programs included to this work were selected based on the Turku Game Lab's preference, personal experience, and recommendations from specialists.

3.1 Spine

Spine is a 2D animation software produced by Esoteric Software and has been available for animators since 2013. Spine Officially supports 19 game toolkits, including Unity, Flash, and cocos2D. The software offers generic runtimes for a number of programming languages. This gives the user the possibility to extend one of these runtimes to support the engine they are using if it is not on the official support list. A variety of third-party runtimes is already available through the official Spine forums. (Spine a)

The software offers many animation tools and features, including bone based animation, free-form image deformation, weighted mesh animation and inverse kinematics to mention few (Spine a). As an addition to the animation tools, the program offers a range of possibilities for the animator to add events like sound effects and overlapping animations which are very easy for the programmers to use in the actual game.

3.2 Toon Boom and Harmony

Toon Boom Animation Inc. is a software company specializing in animation and storyboard software. On October 16, 2015, all their software under the name of “Toon Boom” were discontinued and replaced by software called Harmony, formerly known as Toon Boom Harmony. In Toon Boom's Harmony, the user can draw and animate the character, everything in one program. They also offer some game animation-related features, like adding sound to your character animation and exporting to Unity Game engine. (Toon Boom a.)

Toon Boom software has been used in a list of animated movies and TV series by many big animation companies such as Walt Disney Studios and Universal Animation Studios to name a few. (Toon Boom c.)

3.3 Adobe Flash and After Effects

Adobe Flash and Adobe After Effects are both programs created and offered by Adobe Systems Incorporated. The name of Adobe Flash was changed to Adobe Animate CC on February 08. 2016. (Lee 2015) Prior to this work only the old version of Adobe Flash CC for students was considered and tried, so any possible new features in adobe Animate CC have not been tested.

Adobe Flash works mostly with vector graphics, resulting in clean and smooth images. Flash is often used on web-based animations and browser games. The earlier version of flash CC did not have the bone tool included which greatly decreased the possibilities for game animation with adobe flash. However, in later update, the bone tool was brought back and is included in Adobe Animate CC. The software does not have any easy if any way to include functions like sounds to the animations that could be used by the programmer when needed, but some things can be set for the animations as they are. The animation can be exported in few different forms, but exporting for game engines is not mentioned, the compatibility of the files should be checked before starting the work.

After Effects is a program mainly for creating or editing videos. The program offers tools for character animation as well and with puppet warp, the images can be distorted for the animation. After Effect does not offer any drawing tools and therefore works best when used together with Adobe Illustrator and Photoshop.

3.4 Spriter

Spriter was created by Brash Monkey LLC and had a successful kick starter campaign in 2012. Spriter is a very similar program to Spine, and offers many of the same features including bone based animation, and inverse kinematics, free-form image deformation is under development. On their official forums, it is mentioned to support three common programming languages and exporting is possible straight for Unity and Construct 2 game engines. The program also offers some features for adding sounds and other functions for your animation, much like Spines event function. (Brash monkey.)

4 COMPARISON OF ANIMATION PROGRAMS FOR GAME ANIMATION

The scope of this thesis was too small to gain hands-on experience with so many new animation programs, so some of the research is based on each programs homepages, game developer forums and interviews with animation specialists from Turku Game Lab, Markus Norrgran, and Simo Ruotsalainen.

Even though Toon Boom Harmony seems to be the obvious choice when it comes to animated movies, it is not ideal for game animations. The program is not made for game animations, so the support for game animation is not their priority. They do offer a way of adding sound to your animation, but overall it's not as convenient and easy to use for both animators and coders as in Spine or Spriter. Exporting animations to game engines is supported, but only Unity is mentioned as a supported engine for Harmony (Toon Boom b).

From Adobe flash CC, the bone tool for animation was removed, which made animating with the program very difficult and time-consuming. Without that one tool, animating with Flash was nearly as frustrating and slow as creating each sprite separately on Adobe Photoshop or Illustrator. For this reason, Adobe flash was ruled out as a possible animation program for this work. In later updates, the bone tool was brought back to Adobe Flash Professional CC and is included in Adobe Animate. Even with the tool Adobe flash does not offer any straight way for exporting to any game engines and there is no way for any interaction with anything else but animation.

Like Toon Boom, After Effects is meant for video animation rather than game animation and does not have any tools for interacting with the game code. After Effects offers to export in various video file formats and still images, but no straightforward method to export for any game engine. All adobe programs have been designed to work together and the work done in one program can usually be imported straight to one of the other programs. The programs are used by many professionals for different purposes, but at least by now, adobe does not have any program that would work really well for game animations. The animations can be done professionally, but bringing them to the game engines can be difficult and time-consuming.

Toon Boom Harmony, Adobe Flash and - After Effects are all tools made for professionals and they come with a price, as can be seen in the table below. All offer monthly subscriptions which will need to be taken for one year at a time. One-time payment for the license is also possible, but then we are talking about thousands of dollars for each program and at least for small indie game companies, that is a lot of money just for animation. As an addition to that, according to animation expert Norrgran, you need more than just Flash or After Effects from the Adobe software to be able to animate professionally for games, so more than one license would need to be purchased.

Table 1. Comparing the prices and features of different animation programs

	Spine	**Spriter**	**Adobe Animate (former Flash CC) and After Effects**	**Toon Boom**
Price	Essential: $69 Professional: $299	Limited version: Free Pro: $59	$19,99/month (Animate only) $49,99/month (includes all adobe software)	Essentials: $15/month Advanced: $38/month Premium: $73/month
Skeletal animation	Yes	Yes	Animate: Yes After Effects: no	Yes

Table 1. Comparing the prices and features of different animation programs (continue).

	Spine	Spriter	Adobe Animate (former Flash CC) and After Effects	Toon Boom
Exporting for game engines	Exports to 18 different engines	Exports to Unity 3D and Construct 2	No specified platforms, file format compatibility check required	Exports to Unity
Image deformation	Essentials: no Professional: Yes	under development	Animate: no After Effects: yes	Yes
Inverse kinematics	Essentials: no Professional: Full support	Free version: Basics Pro: Full support	no	Only in premium
Drawing tools	No	No	Some basic tools in Animate	Full support for drawing in the program

Spine offers a trial version for the users, but saving and exporting are both enabled in this version. In order to use the software for actual game animations, the user will have to purchase either the limited Essentials license or the full Professional license for the software. The license will only have to be paid once per user. Businesses with $500,00 USD or more annual revenue are required to purchase their Enterprise license. (Spine b.)

Spriter is the cheapest of these 4 options and that is often mentioned on developer forums when recommending Spriter. Animation specialist Norrgran however, would choose both Spine and the Adobe software package over Spriter.

According to him, Spriter is most like Spine, but many of the functions are laborious and time-consuming, slowing down the workflow critically.

The cheap price also comes with much longer waiting periods for new features. For example, Spriter team announced that they started to work on a feature for deforming images in the program in 2014, and on the feature list on their official website, it is still listed as a “coming soon” feature for the Spriter Pro. Spine already has this feature, and working with it is relatively fast and easy, and it is one of the key features for making character movement look smooth and natural.

5 SPINE UNDER CLOSE INSPECTION

Spine 2D animation program (Spine in short) was originally considered as the animation program for this work by the Turku Game Lab. They had very positive experiences with Spine before and many specialists had recommended the program for game and mobile application animations. However, a comprehensive research of other possibilities for 2D animation program was required before the final decision was made. When comparing all the different options for animations, Spine seems to be the most appealing choice. Spine is not the cheapest option out there, but it is affordable for almost anyone and a lot cheaper than most programs used by professionals. Despite the lower price, Spine comes with the whole package, including everything needed for professional 2D game animations. While Spine does not offer any drawing tools, it has a wide variety of tools and methods for image deformation, allowing the animator to work efficiently with less amount of images. Spine also offers easy ways for the animator and programmer to work together seamlessly, giving the animator more control over other elements that are closely connected to the animations. This program was also recommended by animations specialist Norrgran as the obvious choice and specialist Ruotsalainen had just started to learn how to use Spine. This chapter covers some of Spines important tools for game development which were not needed in the case study animations.

5.1 Skins

These work little differently than the skinning in 3D animation. Spines skins are basically sets of images that can be used to replace each other, for example, a male and a female character. With the use of skins, both characters can be set to use the same bones and animations without having to recreate everything for the second character. To use skins, a “skin placeholder” needs to be set as an attachment under a slot on the spine tree. The placeholders are then replaced with assigned images depending on which skin is active. If some image needs to stay the same for both characters, it can simply be left without a skin placeholder, and the same image will be shown regardless of the skin in use. Alternatively, skins can be used for an avatar system that requires many different body and clothing choices to be mixed and matched. (Spine c.)

Skins are a great way to speed up game development when similar characters can just be set to use the same animations. The animating process is usually very time-consuming but being able to reuse the same animation for multiple characters will greatly improve the artist’s effectiveness. (Song etc. 2013.)

5.2 Events

This is one of the main features that makes Spine stand out when considering game animations. With Spines events, different things can be triggered through the animation. The events by themselves don’t do anything, but they can be set to trigger whatever the developers have set up in their own code through the runtime. The events can also be set to have an integer, float, and string values easily through Spine.

Events are a great way for the animator to effect on something else than just the animation alone. They can make sure sound effects play exactly at the right moment in the animation cycle with all the other possible effects needed for each specific animation. Without a feature like this, many things are easily lost or misinterpreted between animator and programmer when putting the game together. Spine gives more control for the animators to get things to look exactly like they want, and relieves programmers from some of the stress to try and understand how the animator meant some things to happen.

5.3 Weights

Weights in Spine allows the user to bind image mesh to one or more bones. When bound together, the image will deform along the bones when moved. For example, when animating a long tail which consists of one image, more bones can be created for animating it and bound together, the tail image will bend when the bones are moved. Weights can also be adjusted to affect more or less on a specific area of the image to get better deformation results. To speed up the process of tweaking the animation, weights can be adjusted in both setup and animation mode. (Spine d.)

Weights make is possible to easily create animations which look like the gravity is affecting the animated character or object. Assigning more weight to areas that would be heavy and less to anything light, gives more natural flow for the animation. According to animation specialist Ruotsalainen, getting the flow into the animation is very important but usually difficult to achieve with skeleton animations. Animating with skeleton often makes the movement too perfect and because the real movement is not perfect, the animation looks strange. Adding imperfections and effects of gravity to the movements makes them a lot more believable. Spines weights tool is great for achieving this without having to manually edit or draw each frame for the perfect animation. The quality of the animation is usually what gets cut out first when trying to boost games performance. Too many separate images or huge sprite sheets take up a lot of space and processing them is slow. With spine, it is easy to keep the image files small for better performance while still achieving good flow for the animation.

5.4 Inverse Kinematics

Inverse kinematics is a method originally used to better control the movements of robots. From there this method has been adapted to skeleton based computer animation. The usual way to animate with bones is called “forward kinematics”. With this method, objects can be controlled only by rotating the bones between the root and the object that needs to be moved. When animating a hand, for example, the upper arm would need to be moved first and after that the lower arm to get the hand to the desired position. For some movements, this works well, but for others, the bones would need to be tweaked constantly. If for example the upper body wanted to be moved while the hand stays still, it would require the animator to constantly tweak the arm bones to get the hand to stay still. With inverse kinematics, this process is inverted and the hand would stay still while the other bones are moved. The formula used for inverse kinematics calculates the position of the bones and joints between the object and the root. (Welman 1989, Spine e.)

Spine offers animators the possibility to use both, forward - and inverse kinematics for their animations. Additionally, Spine enables a smooth transition between these two, giving the possibility to determine how much each kinematic effects the movement at each moment. This saves the animator from a lot of unnecessary bone tweaking and makes it possible to get smooth and accurate animations efficiently.

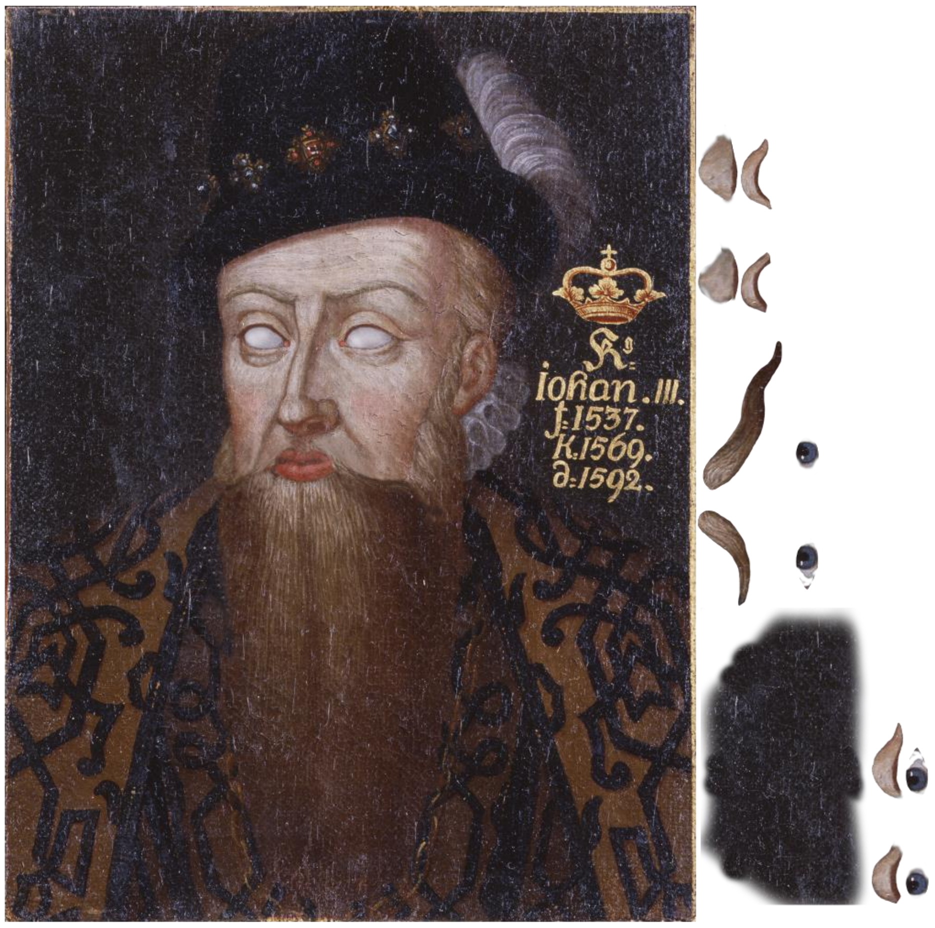
6 CONVERTING SPRITE ANIMATIONS TO SPINE ANIMATIONS

Before this work, the paintings in Turku castle were photographed, the photos broken into pieces for animation and sprite sheets were made for animating them. Using these sprites as a reference, the paintings were animated in Spine, in order to achieve smoother animations with smaller file sizes. The animations were made for an augmented reality application for mobile devices, so small file sizes were really important. In this part of the thesis, the work process of transforming sprite sheet animations to spine animations is described.

6.1 Required elements

The animation process in spine resembles the cutout animation technique. In preparation for this, the animated image needs to be cut into pieces. Each part that is supposed to move needs to be made into a separate image. In image 6 all the separate elements used in one of the animated paintings are shown as an image map created in Spine. The original photos of the paintings were edited in Adobe Photoshop to cut out the eyes and mustaches for animating. The painting itself was edited to be the background for the animation and areas like the eyes were left blank so they would look natural then the cutout images are animated on top of them.

The paintings were edited already for making the sprite sheets used as a reference for animations in this work. A full access was given to all the files used for making the sprite sheets to speed up the work process with Spine.



Picture 4. Example of an image map Spine creates automatically for the animation. Original painting: Juhana III, owned by Turku Museum Centre, painter unknown. Photograph: Martti Puhakka / Turku Museum Centre.

6.2 Importing elements to spine

During the course of this project, the images needed for each animation were saved to separate PNG files in Adobe Photoshop and then imported to Spine one by one. Because of this, the images had to be moved to their right places in Spine to assemble the animation. The first painting had already been saved as PNG files before it was handed out to be animated. That was a very good way to get started with Spine, giving the opportunity to concentrate learning the basics of Spine without having to worry about the way the images were saved. The other paintings were given as PSD files, with everything separated on different layers. The files had lots of extra layers and images that were not used in the spine animation. That and the fact that the first painting had been saved as separate PNG files, is why I decided to save all the needed images separately. At the time, this seemed like the best approach.

More efficient way to import images would have been to save a Photoshop file including only the required images on different layers and then export it with the LayerstoPNG tool provided by Spine. This would have placed all the separate images in the right places automatically and saved time from saving each image separately when working with Spine.

6.3 Swapping images versus animating with mesh tool

For the first animated painting, the movement included the person with head movements and expressions. For the sprites, this was executed as a separate, slightly tilted image of the head which could be put on top to hide the original head. Showing and hiding this separate head gave the illusion of the head shaking, but with only two frames the movement looked very stiff.

In Spine, the movement was first created by using this extra image. For different results, the same movement was also done by using the mesh tool on the background image alone. For a small movement like shaking one's head, the mesh is an excellent tool, by the right use image deformation, the movement can be animated to look very real.

For best results, a new mesh was created around the whole head. An important thing was to create enough vertices along the edges of the mesh and inside it to cut the area into small enough triangles for better control when animating. Once the mesh was in place, the vertices inside of it were moved in animation mode and locked on the timeline to create the effect of the head turning. It took several attempts to get the movement look natural enough.

Most vertices were placed to the face area to gain control over the expression as well as creating an illusion of the face shaking. These vertices were used to animate the face to frown before and while it is shaking to make it look more alive.

In this case, it was very important to pay attention to the background around the face because the mesh was created on top of the whole background image to animate only a part of it. Moving certain vertices moved parts of the background as well and this created some unwanted movement onto the image. To avoid this, the first vertices inside the mesh were placed very close to the ones on the edges. This created small triangles around the image and if the vertices creating those triangles were left untouched, the background remained in place. This was very time consuming when done for the first time and the mash had to be edited several times before all the unwanted movement was removed from the background.

When comparing the animation made with the mesh and the one with swapping the images, the mesh animation is noticeably smoother and lighter for the program. The image swap was a lot faster to execute, but the results are not nearly as good. And to change the expression with image swapping would have required a lot more image editing and more than one additional image.

The mesh tool was used in the other paintings as well, for smooth movement and to eliminate some of the images used to create the animations sprites. It is well worth it to learn to use the mesh tool properly, it can save a lot of time for the artist when each little change does not have to be drawn but can instead be edited from the basic images.

6.4 Fine tuning the animation

Once the base animation is done, it is important to play it through many times and try to look for mistakes or unnatural movement. All the big movements should be done first and after that add some more keyframes in between to make the movement more natural. The timing of certain animations is also very important. For one of the paintings, it took a few reanimating attempts, a lot of replaying and small editing to get the eyebrow movement to look just right.

It is very important to plan the whole animation from start to finish at first so it is easy to set keyframes for unchanged images where needed and then modify it accordingly in between these frames. During this work, the animating was first started without proper planning and new ideas were executed on the way. More than once this led to having to delete several keyframes because end frame should have been set first to fit the rest of the frames nicely between the start and finish of the movement.

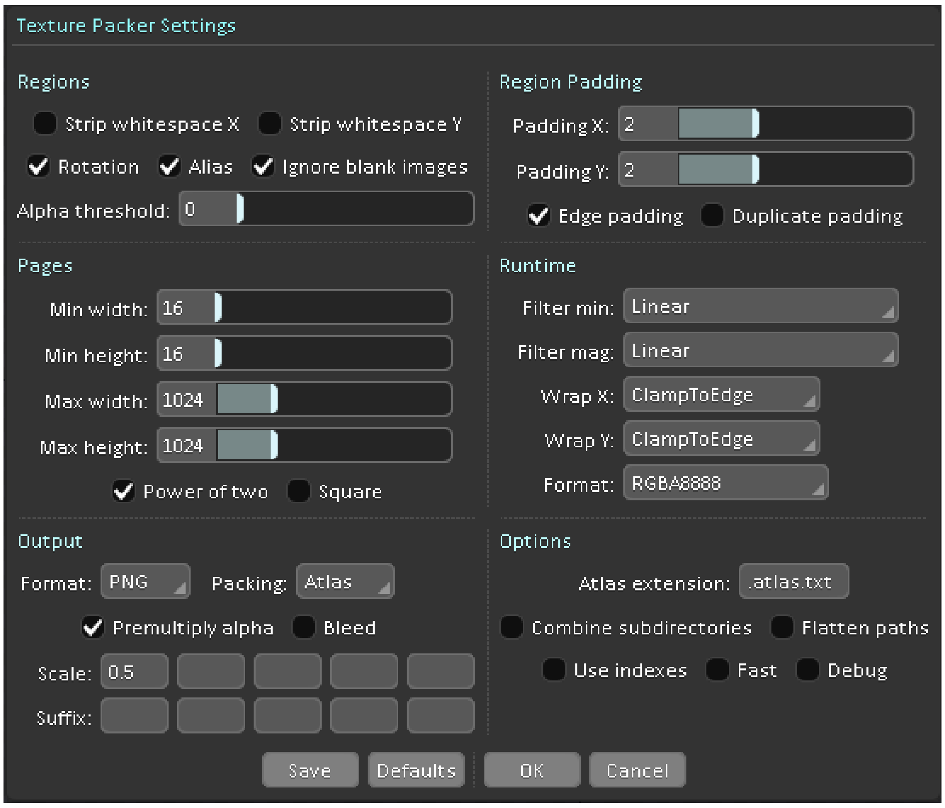
6.5 Exporting Unity compatible animation files

Before Exporting anything, the runtime for Unity was downloaded from Spines homepage and installed to the Unity project. Once installed, the skeletal animation data created in Spine could be used and manipulated in Unity without having to use any other plugins.

When exporting for Unity, the JSON was selected as a file format. The most important thing for a good workflow was to set the output directory to be the right folder in the Unity project. In a tutorial from Spines official website, it is recommended to create a folder for sprites under the project’s root folder and then under that create a separate folder for each character and other animated objects. This way the unity project stays clean and every animation is easy to find even in a big game project.

An atlas was created for unity and the atlas settings were checked to make sure everything was saved right for Unity to easily understand the files. On picture five the Unity-specific atlas settings used in this work can be seen. The most important thing was to change the extension from the default “.atlas” to “.atlas.txt” so Unity can read it as a text file. Because the animations had been created using a lot of meshes, “strip whitespace x” and “– y” were unchecked. The format was set to PNG and “premultiply alpha” was checked to get as accurate images as possible.

Once the settings are set for a project, they will be saved and don’t need to be changed again unless something is not working as it’s supposed to. If the animations or images used are changed in Spine after the project has been exported to Unity, the project can easily be exported again with the exact same setting and the changes will become visible in Unity.



Texture Packer Settings

Regions
- Strip whitespace X
- Strip whitespace Y
- ✓ Rotation
- ✓ Alias
- ✓ Ignore blank images

Alpha threshold: 0

Region Padding

Padding X: 2

Padding Y: 2

- ✓ Edge padding
- Duplicate padding

Pages

Min width: 16

Min height: 16

Max width: 1024

Max height: 1024

- ✓ Power of two
- Square

Runtime

Filter min: Linear

Filter mag: Linear

Wrap X: ClampToEdge

Wrap Y: ClampToEdge

Format: RGBA8888

Output

Format: PNG Packing: Atlas

- ✓ Premultiply alpha
- Bleed

Scale: 0.5

Suffix:

Options

Atlas extension: .atlas.txt

- Combine subdirectories
- Flatten paths
- Use indexes
- Fast
- Debug

Picture 5. Texture Packer Settings window from Spine, with Unity compatible export settings.

The rest of the settings were left as they were in default. Only the scale was set to 0.5 because the image files were fairly big and needed to be scaled down for better performance for the mobile application. The same could have been achieved by scaling down the images before starting the animation process. In this case, the animations were fairly simple and consisted of only a few images so working with big files was not an issue. With very complex animations consisting of many big images, it would be smarter to define the desired resolution and maximum diameters for the graphic files before the animation process. This way the animation process will be as quick and smooth as possible. Any graphics based program will eventually slow down if it is filled with too many big images, even the most powerful computers will eventually reach their limits.

7 CONLUSIONS

This thesis concentrates on many aspects of 2D animation in video games by exploring the different animations techniques used and comparing some of the top animation programs out there. The purpose was to find the best way to do 2D animations for games and the program that best delivers animation for the game developer's needs.

Making successful game animations is very different from making a successful short film animation. Game animations need to be dynamic and flexible for smooth gameplay. Often animations need to be blended together or animation loop needs to be cut from the middle in order for a new animation to take place. (Walther-Franks & Malaka 2014.) For example, the character might be running when suddenly the player wants to strike an enemy. These two animations need to be made so, that the strike animation looks natural no matter what part of the running animation is playing at that exact moment. Overlapping animations are also common in games, for example, the character needs to blink their eyes no matter if they are standing, walking or running. Some animations need to be triggered only by certain things happening in the code, for example, the character dying.

What's even more important for games than smooth animations themselves, is that the animations can easily be imported to different game engines. With many great animation programs, problems start to arise when the developers try to import the animations to their games. The files are not working as they are supposed to or the engine does not even understand any of the file formats that can be exported from the animation software. In these situations, developers might need to compromise to import the animations as sprite sheets.

Sprite sheets might have been the big thing for 2D games in the 90's, but today professional game developers rarely want to sacrifice disc space for them. Sprite sheets can quickly grow to enormous sizes making the game huge in file size and slow the performance with too many big images to process constantly. For a powerful PC, this might not be such a big problem, but today's 2D games are often developed for mobile devices. No one wants to download a game that takes half of the disc space on their mobile device and on top of that does not run smoothly. To solve this issue with limited space, developers have come up with a way to export numerical data for skeletal animation. With this method, the quality of graphics can be brought up without having to cut down on animation quality nor the games performance.

Some of the biggest miscommunications in game development happen between the artist and the programmer. It can be difficult for the programmers to get the artist to understand what is needed, or the artist can’t explain to the programmers how exactly they envisioned something to be in the game. Especially animators have to often compromise the quality of their work if they do not know how to work with the animation in the game engine or if the programmers don’t want the animator to interfere with the code. When using sprite sheets, the animator might have meant for some frames to be used more than once during the animation cycle but the programmers just cut the sheets and use all frames once in order. The animator might also have meant some frames to be shown for longer or shorter time to get the animation to look smoother, but the programmer just makes them all last the same amount of time because it is fast and easy to do that way. This way the animation still gets done and is usually with acceptable quality, but it is not as the animator intended it to be.

A good program for game animation should be capable of all of the above; smooth, flexible animation, exporting the right file formats while keeping these files as small as possible without decreasing the quality and making the communication between the artist and the programmer somehow easier. Ideally, the same program would offer all the tools needed for drawing the images to be animated so the artist could concentrate on mastering only that one program.

When comparing different animation programs, Toon Boom Harmony was clearly the top choice when thinking about animation alone. However, when trying to import Toon Boom animations to a game engine, they just don’t work well with the code, if at all. In addition, Toon Boom does not offer tools to help the communication between the artist and the programmer.

Spine and Spriter seem to be the two programs that were both designed to provide the tools needed in game development. When comparing these two, Spriter’s lower price is clearly affecting the quality of the program and its development. With a little higher price, the full version of Spine offers great image deformation tools and exporting to several popular game engines, both features that are lacking from Spriter. Spine is definitely the professional choice of these two, but for someone with a really low budget, Spriter’s free or full version is a better choice than Spine Essential. However, if planning to become more professional with game animation in the future, Spine essential is a good investment to be upgraded to the full version later.

When inspecting the animation done for this work, the animation made in Spine is definitely smoother than the animation achieved by sprite sheets. The amination files are also much smaller in size, boosting the applications performance noticeably.

REFERENCES

Bendazzi, G. 1996. Quirino Cristiani, The Untold Story of Argentina's Pioneer Animator. Animation World Magazine. Volume 1, No.4. Referenced 25.4.2016. http://www.awn.com/mag/issue1.4/articles/bendazzi1.4.html

Brash monkey. About us. Referenced 20.3.2016. https://brashmonkey.com/about-us/.

Bratt, B. 2011. Rotoscoping: Techniques and tools for the Aspiring Artist. Amsterdam: Elsevier Inc.

Brown, S. 2014. Sprite Supreme: A Brief History of Graphics, Part Two. Referenced 1.4.2016. https://www.youtube.com/watch?v=a1yBP5t-fSA

Bui, L. H. and Bui, T. D. 2009. "Fast and Realistic 2D Facial Animation Based on Image Warping" in International Conference on Knowledge and Systems Engineering. Vietnam. Hanoi. pp. 81-86.

McMillen, E. & Refenes, T. Super Meat Boy. 2010. Team Meat. Referenced 11.4.1016. https://en.wikipedia.org/wiki/Super_Meat_Boy.

Igarashi, T.; Moscovich, T. and J. F. Hughes. 2005 "As-rigid-as-possible shape manipulation" ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH. Vol 24, issue 3. New York. pp. 1134–1141.

Izani, M. ; Aishah ; Eshaq, A. R. and Norzaiha. 2003. "Analysis of the keyframe animation and motion capture case studies" in Student Conference on Research and Development. SCORED. pp 177-182.

Lee, R. 2015. Welcome Adobe Animate CC, a new era for Flash Professional. Referenced 17.4.2016. http://blogs.adobe.com/animate/welcome-adobe-animate-cc-a-new-era-for-flash-professional/.

Limbo. 2010. Playdead. Referenced 20.4.2016. http://playdead.com/limbo/

Mao, C.; Yi, Z.; JianGang; O. and Guo-tao, H. 2010. "Game Design and Development Based on Logical Animation Platform". in International conference on Computational and Information Sciences. ICCIS. pp. 573-576.

Mechner, J. 1989. Prince of Persia. Brøderbund. Referenced 17.4.2016. https://en.wikipedia.org/wiki/Prince_of_Persia.

Mota, T. Esperança, C. and Oliveira, A. 2011. "2D Shape Deformation Based on Positional Constraints and Layer Manipulation" in Games and Digital Entertainment. SBGAMES. Brazil. pp. 1-10.

Motwani, R.; Ambardekar, A.; Motwani, M.; Harris; F. C. 2008. "Robust Watermarking of 3D skinning mesh animations" in 2008 IEEE international conference on Acoustics, Speech and Signal Processing. Las Vegas. pp. 1752-1756.

Pong, 1972, Atari Inc. Referenced 20.4.2016. http://www.pong-story.com/

Pranatio, G. and Kosala, R. 2012. "A Comparative Study of Skeletal and Keyframe

Animations in a Multiplayer Online Game” in 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies. ACT. pp. 143 – 145

Ruppert, J. 1995. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. Journal of Algorithms. Vol.18, Issue 3. pp. 548–585.

Sanchez-Crespo, D. 2003. Core Techniques and Algorithms in Game Programming. New Riders Publications. Edition 3.

Song, Z.; Yu, J.; Zhou, C; Tao, D. and Xie, Y. 2013. “Skeleton correspondence construction and its applications in animation style reusing” in Neurocomputing vol. 120. p. 461-468.

Spine a. What is Spine?. http://esotericsoftware.com/spine-in-depth#What-is-Spine. Referenced 17.4.2016

Spine b. Purchase spine. https://esotericsoftware.com/spine-purchase. Referenced 17.4.2016

Spine c. Skins. http://esotericsoftware.com/spine-skins. Referenced 17.4.2016

Spine d. Weights. http://esotericsoftware.com/spine-weights. Referenced 17.4.2016

Spine e. Inverse Kinematics. http://esotericsoftware.com/spine-ik-constraints. Referenced 17.4.2016

Street Fighter II. 1991. Capcom. Referenced 11.4.2016
http://streetfighter.wikia.com/wiki/Street_Fighter_II

Toon Boom a. Harmony. https://www.toonboom.com/products/harmony. Referenced 17.4.2016

Toon Boom b. Export to game engine. Referenced 17.4.2016.
https://www.toonboom.com > Harmony > features > export to game engine

Toon Boom c. Toon Boom Customer Productions. Referenced 20.4.2016.
https://www.toonboom.com/company/customer-productions

Walther-Franks, B. and Malaka, R. 2014. “An interaction approach to computer animation” in Entertainment computing. Vol. 5, Issue 4. pp. 271 – 283

Waytt, A. 2010. The Complete Digital Animation Course. London: Thames & Hudson Ltd.

Welman,C. 1989. “Inverse kinematics and geometric constraints for articulated figure manipulation” in B. Sc. Simon Fraser University. Referenced 17.4.2016.
http://graphics.ucsd.edu/courses/cse169_w04/welman.pdf

Interview questions and ansvers

questions

Question 1: How long have you been animating with computers, and when did you start to do game animations?

Question 2: Which animation programs have you tried?

Question 3: Which of them would you recommend for game animations?

Question 4: What makes this program better than the others when considering game animations?

Question 5: If the program you chose was not available, could you work on game animations with one of the other programs you have tried?

Question 6: Other comments?

Answers, Markus Norrgran

14.3.2016

Answer 1:

Been animating for the past 3 years and before that I animated 2 years on the side of other things. The focus has been game animation the whole time.

Answer 2:

Spine, Spriter, Adobe Flash, Adobe animate, Toon Boom Harmony, Adobe After Effects

Answer 3:

Spine, clearly

Answer 4:

When you take an animation from Spine to a game engine, you can work directly with the programmer. For example the animator can set sounds to the right places on the animations. Small side animations, like occasional blink of an eye, can be easily called on top of the main animation without having to make a new big animation. Transitions on the animations, like character turning, are very easy to make smoothly. The program has been designed for both, animators and programmers.

Answer 5:

If I have to choose, it would be the adobe programs.

Many of the useful tools on Spine are not available on the other programs, including the adobe software. Of all the mentioned program, Spriter is most like Spine, but many of the functions are laborious and time consuming, slowing down the workflow critically.

Answer 6:

About Spine:

Speeds up working

Animator gets more authority, no compromising between animators and programmers.

Answers, Simo Ruotsalainen

4.4.2016

Answer 1:

Been doing project based 2D animation since 2003. Animation for games since 2011, also project based.

Answer 2:

Toonz, after Effects, Blender, Spine

Answer 3:

Blender for 3D. Spine is simple for 2D.

Answer 4:

“Squash and strech” is easy to do with it Spine, it eases communication between the artist and programmer, it’s fairly cheap and is approeachable. Weights are a bonus but lively animations are still more difficult to do than with traditional animation.

Question 5 was skipped due to the flow of the conversation.

Answer 6:

Cutout and skeletal animation is stiff and difficult to get to “flow”.

It is important for the animator to know what the animation needs to achieve, in order to do their job as well as they can.