UNIVERSITY OF PAVIA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING

MASTER'S DEGREE IN COMPUTER SCIENCE AND MULTIMEDIA

MASTER THESIS

TITLE in English

3D computer graphics for digital humanities: experiencing Pavia cultural heritage through the digital eye

TITLE in Italian

Computer grafica 3D per digital humanities: vivere il patrimonio culturale pavese attraverso l'occhio digitale

Candidate: Alessio Gullotti

Supervisor: Prof. Virginio Cantoni

A.Y.2020/2021

Index

Introduction

Understanding the fundamentals of 3d in blender

Project presentation

Abstract

This study aims to digitally reconstruct the Renaissance history of the city of Pavia, investigating the creation process as well as the media which will be used to deliver the virtual experience.

It focuses on the recording of a virtual tour of the city in the second half of the 1500s and the search for an interactive method to visualize the changes that have occurred during the century.

Especially in a city like Pavia, where some monuments have been partially or completely destroyed, modeling and rendering with the aid of reference images are effective tools to reconstruct what no longer exists, creating the possibility of producing and distributing humanistic knowledge through immersive experiences.

INTRODUCTION

The term *graphic* generically indicates the product of design oriented towards visual communication.

Computer graphics is a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. It is the science and art of communicating visually via a computer's display and its interaction devices.

It is a cross-disciplinary field in which physics, mathematics, human perception, human-computer interaction, engineering, graphic design, and art all play important roles.

Physics is used to model light and to simulate phenomena, mathematics to describe shape. Human perceptual abilities determine the correct amount of detail in models - complexity is useless if it will be not noticed.

Graphic design and art combine with human-computer interaction to make the computer-to-human direction of communication most effective.

We could state that computer graphics refers to taking a model of the objects in a scene (a geometric description of the things in the scene and a description of how they reflect light) and a model of the light emitted into the scene (a mathematical description of the sources of light energy, the directions of radiation, the distribution of light wavelengths, etc.), and then producing a representation of a particular view of the scene (the light arriving at some imaginary eye or camera in the scene).

Usually abbreviated as CG, it includes user interface design, sprite graphics, rendering, ray tracing, geometry processing, computer animation, vector graphics, 3D modeling, shaders, GPU (graphics processing units) design, implicit surfaces, visualization, image processing, computational photography, scientific visualization, computational geometry and computer vision, motion capture, motion tracking, particles system, physics system, among others.

Although originally it was a subject treated almost exclusively by researchers in universities, very soon arose the curiosity to see what could create such an instrument in the hands of artists. Many of the so-called innovators were housed in universities and research labs and were working to solve fundamental problems of showing data on computer through "pictures" and founding a way to interact with them.

So, it was created as a visualization tool for scientist and engineers in government and corporate research centers such as Bell Labs and Boeing in the 1950s. It is at Boeing, that William Fetter coined the phrase "*Computer Graphics*" to describe what he was doing at the time (Fetter has said that the terms were given to him by Verne Hudson of the Wichita Division of Boeing).

The term encompasses two-dimensional graphics and image processing, but often refers to the study of three-dimensional computer graphics. The basic process in 3D computer graphics is 3D modeling (or three-dimensional modeling), the process of developing a mathematical coordinate-based representation

of any three-dimensional surface of an object (living or inanimate). It is possible to create any kind of object, either real or imaginary, capturing size, shape, and shading. The product is called 3D model and the creator may be referred to as a 3D artist or a 3D modeler.

A 3D model can also be displayed as a 2-dimensional image or used in a computer simulation of physical phenomena. To create a 2-dimensional image, the model should be rendered, this process requires to set up the scene with lights and a camera and generate an image from a mathematical description of a 3D scene, interpreted by algorithms that define the color of each point of the digital image. It can be physically created using a 3D printing device that form 2D layers of the model with three-dimensional material, one layer at a time. In many fields such as manufacturing, 3D modeling is merely a stage in the entire development process, often used to create rapid and valid prototype before producing any physical product.

The possibilities provided by 3D software seems endless and are improving each day passing by. As already said, there's not only the possibility to recreate objects shapes through simple modeling techniques (move, rotate, extrude, inset faces, edges or vertices) and even the same properties of the surface, including how interacts with the light, smoothness or roughness, transparency among others, through the process of shading, you can also use an even more manual approach sculpting and painting the meshes or a more computerized approach, called procedural, that allows to exploit a set of rules to create large and complex environment or textures with specific patterns, in addition with the benefit of infinite resolution.

Rocks, plants, tree trunks are examples of very complex objects with high levels of details combined with the inherent randomness of nature that makes them more difficult to reproduce them with high fidelity. However, scanner can collect data (shape and appearance) of any kind of object to construct digital 3D models. 3D scanning can be done using laser scanner or photogrammetry.

Another example of technique that collects data, specifically motion data, is motion tracking. In the simplest form possible can be imagined as following a point (usually a group of pixels) moving on a screen.

In general, it may refer to:

1. Motion capture: the process of recording the movement of objects or people
2. Match moving: a cinematic technique that allows the insertion of computer graphics into live-action footage with correct position, scale, orientation, and motion relative to the objects in the shot
3. Video tracking: the process of locating a moving object over time using a camera

Motion can be approached in multiple ways.

Highly chaotic systems, natural phenomena, or processes caused by chemical reactions, such as sparks, falling leaves, rock falls, clouds, fog, snow, dust, meteor tails, stars and galaxies, or abstract visual effects like glowing trails, magic spells would need a huge effort by an animator to recreate them by hand.

Therefore, math and physics are tools widely used in 3D animation. Implementing real world phenomena is difficult and it always requires approximation. 3D software provides algorithms for creating simulation through particle systems and physics engine. The idea of particles system (as the name suggests) is to think about object as particles; fluids would be difficult to animate using traditional technique, instead using millions of particles (small enough to make impossible to distinguish them) we can recreate an approximation of real-world motion using physics law

They are also useful for creating hair and furry animals, where a vast number of little strands are needed and interact with external forces being able to be attached to the skin.

Physics engines include the possibility to simulate the motion of solid objects, create forces to influence a simulation. Object can be used as brushes or canvas if dynamic paint is used, creating vertex colors, image sequences or displacement. This makes many effects possible like, for example footsteps in the snow, raindrops that make the ground wet, paint that sticks to walls, or objects that gradually freeze.

Soft body simulation is used for simulating soft deformable objects. It was designed primarily for adding secondary motion to animation, like jiggle for body parts of a moving character.

Figure 1.1 A coefficient establishes the amount of deformation of a soft body

It also works for simulating more general soft objects that bend, deform, and react to forces like gravity and wind, or collide with other objects.

Detecting collision is also a complex subject, especially with soft bodies that deform and particles system.

Cloth simulation is one of the hardest aspects of computer graphics, it is a deceptively simple real-world item, but it has extraordinarily complex internal and environmental interactions.

Cloth is modeled to simulate real world objects such as fabrics, flags, banners. And yet cloth can also be used to model 3D objects such as teddy bears, pillows, balloons, or balls.

Figure 1.2 Cloth interacts with and is affected by other moving objects, the wind, and other forces, as well as a general aerodynamic model.

Fluid physics are used to simulate physical properties of liquids especially water. Gas or smoke simulations are a subset of the fluids system and can be used for simulating collections of airborne solids, liquid particulates and gases, such as those that make up smoke.

Figure 1.3 Fluid simulations are usually generated using an element as a source and a domain inside of which the simulations are contained

Much of the current research in graphics is in methods for creating geometric models, methods for representing surface reflectance (and subsurface reflectance, and reflectance of participating media such as fog and smoke, etc.), the animation of scenes by physical laws and by approximations of those laws, the control of animation, interaction with virtual objects, the invention of non-photorealistic representations, and, in recent years, an increasing integration of techniques from computer vision. As a result, the fields of computer graphics and computer vision are growing increasingly closer to each other.

While graphics has had an enormous impact on the entertainment industry and digital art, its influence in other areas—science, engineering (including computer-aided design and manufacturing), medicine, desktop publishing, website design, communication, information handling, and analysis are just a few examples—continues to grow daily.

1.1 History of 3D modeling

Paradoxically, the history of 3D modeling began way before the first PC appeared. It all started with the mathematical ideas that are behind 3D visualization.

In fact, some of the basic ideas came from Euclid, sometimes referred to as the “founder of geometry”, who lived in the 3rd century BC. Then, Rene Descartes in the 1600s gave the world analytic geometry, also known as coordinate geometry, which allowed to accurately track distances and locations. Later in the mid 18th century, English mathematician James Joseph Sylvester invented matrix mathematics.

In the 1950s, computers were developed and put to many mathematical application, mainly military and scientific, but not much else.

The progress in the history of 3D modeling came when the first commercially available CAD or Computer Aided Design systems started coming out in the 1960s.

The first software capable of revolutionizing how computer can be utilized was Sketchpad. Up until that time, buttons and switches were mainly used to provide input for computers. In 1963, Ivan Sutherland, American electrical engineer and computer scientist, was working on his PhD in EE in the Lincoln Labs on their TX-2 computer. He presented Sketchpad, also known as “Robot Draftsman”, described in a paper, "Sketchpad: A Man-machine Graphical Communications System", which used the light pen to create engineering drawings directly on the CRT. Highly precise drawings could be created, manipulated, duplicated, and stored. The TX-2 included a nine-inch CRT and a light pen which first gave Sutherland his idea. He wanted to be able to draw on the computer and the software he invented made this possible. It is considered the ancestor of modern computer-aided design (CAD) programs as well as a breakthrough in the development of computer graphics in general.

But it wasn't the only one, Tom Diamond patented an approach to handwriting recognition in 1957 that utilized an innovative tablet that detected the regions of interaction, giving rise to the graphics tablet. One of the most innovative input approaches at the time was manifested by the Rand Tablet. It consisted of a matrix of crossed conductors. The circuitry of the tablet used to switch techniques to apply pulses to the conductors in sequence, thus coding their individual locations. General Motors and IBM formally disclosed at the 1964 Fall Joint Computer Conference the DAC-1 system, introducing transformations on geometric objects for display, including rotation and and zoom, and a no-display (later called "clipping") function. It used a light pencil, instead of the commonly used light gun or light pen.

Figure 1.4 From left to right: Sketchpad, Rand tablet, Dac-1 system. The first attempts to change the interaction with machines.

If the sixties started improving the human-computer interaction to achieve the possibility of using the screen as a sheet of paper, the seventies made progress in the visualization of 3d objects. Since the objects were represent as lines on a screen (often referred as wireframe), researchers started studying solutions to represent the surface of objects. One of the first effective technique that sped up processing by simplifying the original algorithms for rendering and delivered better visual results in light, reflection and shading was Gouraud shading (or Smooth shading), a per-vertex color computation technique published in 1971. It uses interpolation to produce continuous shading of surfaces represented by polygon meshes. It is most often used on triangle meshes by computing the lighting at the corners of each triangle and linearly interpolating the resulting colors for each pixel covered by the triangle, achieving continuous lighting.

Computer graphics pioneer Bui Tuong Phong picked up where Henri Gouraud left and invented a per-fragment color computation, also called normal-vector interpolation shading or Phong interpolation. It interpolates surface normals across rasterized polygons and computes pixel colors based on the interpolated normals and a reflection model. Phong shading may also refer to the specific combination of Phong interpolation and the Phong reflection model.

Phong reflection model describes the way a surface reflects light as a combination of the diffuse reflection of rough surfaces with the specular reflection of shiny surfaces. It also includes an ambient term to account for the small amount of light that is scattered about the entire scene. Obviously, it achieves a greater level of photorealism in the materials with an increasing cost in rendering time.

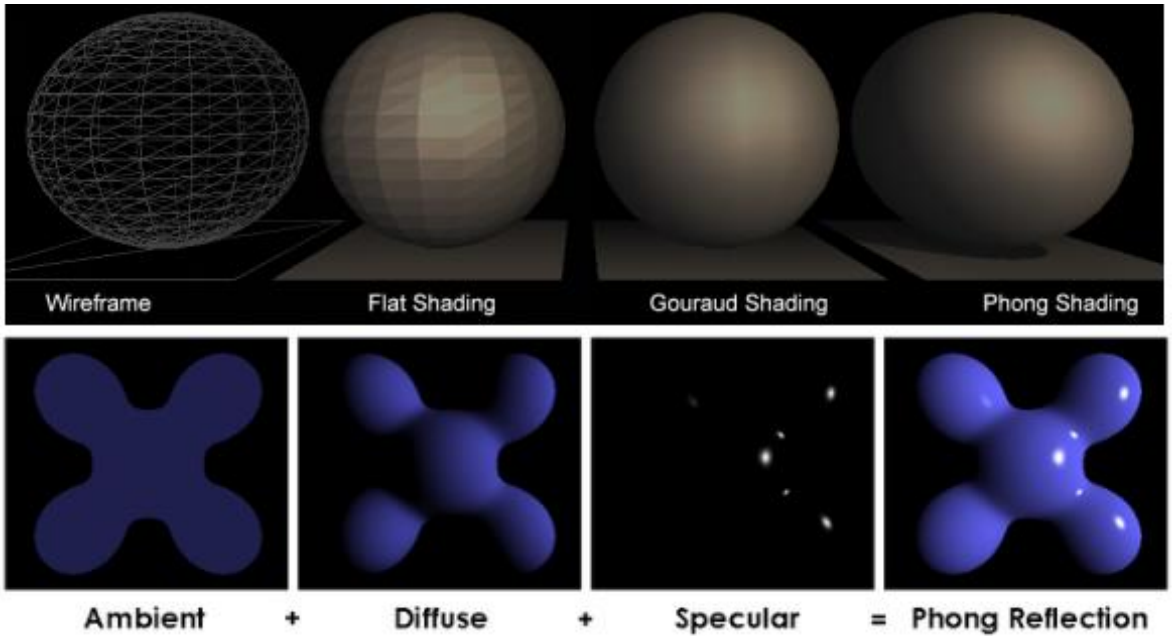


Figure 1.5 Comparison between wireframe, flat shading and smooth shading. Phong considers the reflection as combination of ambient and diffuse light and specular reflection.

There are different rendering techniques to capture the final look of a 3d model in an image, but all of them are based on two approaches: ray casting (rasterization) and ray tracing.

The first ray casting algorithm was presented by Arthur Appel in 1968. The basic idea of ray casting is to start the rays from the eye, one per pixel, and find the closest object that blocks the path (it is necessary to think of an image as a grid, in which each square corresponds to a pixel). The hit object is what the eye sees through that pixel. Using the properties of the material and the effects of light in the scene, this algorithm can determine the color of the object.

Turner Whitted presented an algorithm for ray tracing in 1979. Previous algorithms launched the beam from the eye towards the scene, but the rays were no longer tracked. Whitted considered that when a ray hits a surface, it can generate up to three new ray types: reflection, refraction and shadow. A reflected ray continues in the direction of the mirror reflection on a shiny surface. At this point it interacts with other objects in the scene; the first object that hits will be the one seen in the reflection present on the original object. The refracted ray travels through the transparent material in a similar way, with the addition that it can enter or exit a material. To avoid tracing all the rays present in a scene, a shadow ray is used to test whether the surface is visible to a light. A ray hits a surface somewhere, if this point "sees" the light, a ray (from the point of view of the computer a segment) is followed to the source. If an opaque object is encountered on the way, the surface is in shadow, and that source does not contribute to the color calculation.

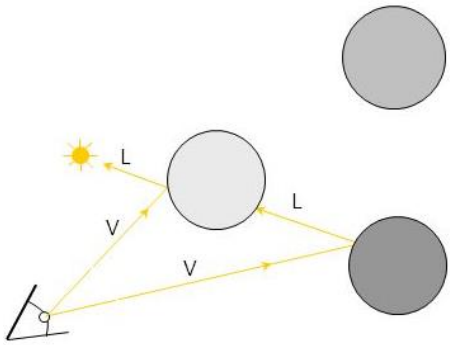


Figure 1.6 Ray casting, Arthur Appel 1968.

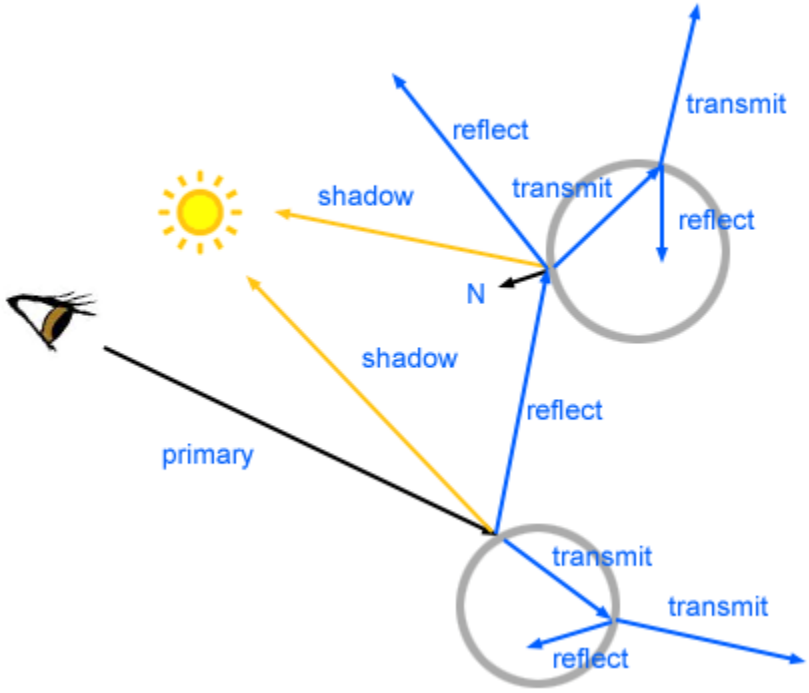


Figure 1.7 Ray tracing, Turner Whitted 1979.

It seems incredible, but in this decade the University of Utah was the place to go if you need an important breakthrough in this field. After Phong and Gouraud, another important moment was in 1975 when Martin Newell developed the Utah teapot model, or the Newell teapot, which is a 3D model of an ordinary Melitta-brand teapot. It was based on a mathematical model, and he found its shape ideal for testing because of its structure, variety of surfaces it possessed and the item's ability to cast shadows on itself. It has become the symbol for 3D computer graphics and an in-joke within the computer graphics community. Using a teapot model is considered the 3D equivalent of a "Hello, World!" program, a way to create an easy 3D scene with a somewhat complex model acting as the basic geometry for a scene

with a light setup. The original, physical teapot resides in the Computer History Museum in Mountain View, California.

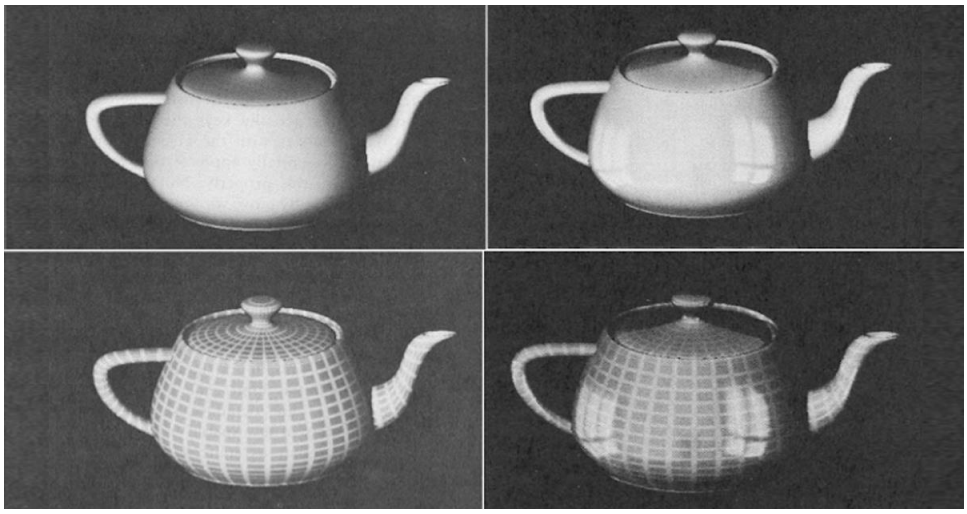


Figure 1.8 Newell teapot by Martin Newell

In the same period, Edward Catmull made two new fundamental computer-graphics discoveries: texture mapping and bicubic patches. Texture mapping exploits the UV mapping process to project a 2D image onto a 3D surface. He invented algorithms for spatial anti-aliasing and refining subdivision surfaces. He also independently discovered Z-buffering, which had been described, 8 months before, by Wolfgang Straßer in his PhD thesis "Schnelle Kurven-und Flächendarstellung auf graphischen Sichtgeräten".

Finally, in 1978, James F. Blinn devised new methods to represent how objects and light interact in a three-dimensional virtual world, like environment mapping and bump mapping. Bump mapping is a texture mapping technique in computer graphics for simulating bumps and wrinkles on the surface of an object. This is achieved by perturbing the surface normals of the object and using the perturbed normal during lighting calculations. The result is an apparently bumpy surface rather than a smooth surface although the surface of the underlying object is not changed.

Blinn became widely known for his work as a computer graphics expert at NASA's Jet Propulsion Laboratory (JPL), particularly the animations for the Voyager project, and the research of the Blinn–Phong shading model.

He is credited with formulating Blinn's Law, which asserts that rendering time tends to remain constant, even as computers get faster.

Only in 1996 a new technique, called normal mapping, used for faking the lighting of bumps and dents was introduced by Krishnamurthy and Levoy. It is used to add details without using more polygons. A common use of this technique is to greatly enhance the appearance and details of a low polygon model by generating a normal map from a high polygon model or height map. Normal maps are commonly stored as regular RGB images where the RGB components correspond to the X, Y, and Z coordinates, respectively, of the surface normal.

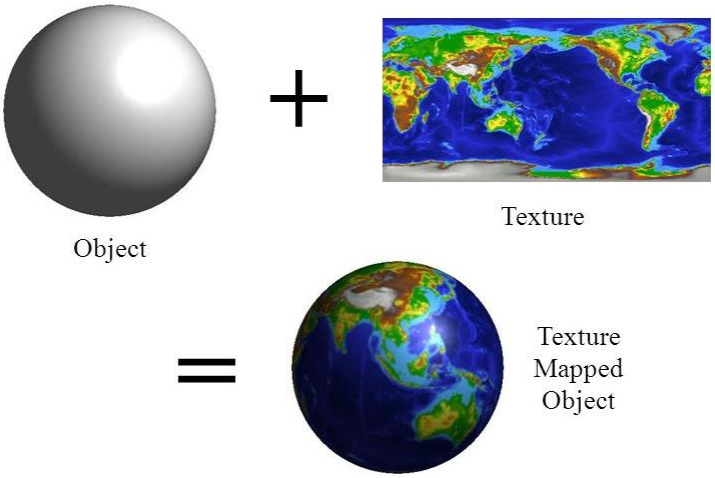


Figure 1.9 Image of an atlas applied to a sphere.

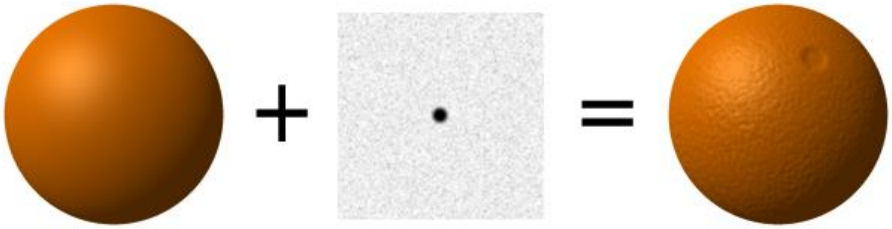


Figure 1.10 Bump map adds details preserving the original mesh.

The eighties are the period where computer graphics garnered most of his popularity with his modernization and commercialization. Since the home computer proliferated, a subject which had previously been an academics-only discipline was adopted by a much larger audience, and the number of computer graphics developers increased significantly. Proprietary animation software was developed by and for commercial firms using it for 3D animations in sequences for movies and commercials as well as for TV spots and music videos. The 1980s is also called the golden era of videogames; millions-selling systems from Atari, Nintendo and Sega, among other companies, exposed computer graphics for the first time to a new, young, and impressionable audience – as did MS-DOS-based personal computers, Apple IIs, Macs, and Amigas, all of which also allowed users to program their own games if skilled enough. For the arcades, advances were made in commercial, real-time 3D graphics.

This decade is known for the rise of pseudo 3D graphics, the developers used physical projections which gave the illusion of 3D presence. First person view (FPV) shooters and slashers were dynamically re-developing the advantages of 3D technology. The racing videogames developers started to use a rear-positioned camera view. In other situation, such as when the player must fight with enemies around him, three dimensions were necessary to make it more playable. Wolfenstein 3D was a pioneer in this category. The creators of the first 3D shooter game succeeded to implement texture mapping to wrap different objects.

The eighties generated an army of enthusiast young people amazed by the power of computer graphics inspiring them to create their own games and companies.

The last decade of the XX century gave birth to the "the world's first GPU", or at least this was the slogan used by Nvidia when marketed the GeForce 256 in 1999. The seminal GeForce 256 was the first home video card billed as a graphics processing unit or GPU, which in its own words contained "integrated transform, lighting, triangle setup/clipping, and rendering engines". In the 1970s, the term originally stood for graphics processor unit and described a programmable processing unit independently working from the CPU and responsible for graphics manipulation and output. Later, in 1994, Sony used the term (now standing for graphics processing unit) in reference to the PlayStation console's Toshiba-designed Sony GPU in 1994.

By the end of the decade, computers adopted common frameworks for graphics processing such as DirectX and OpenGL. AMD also became a leading developer of graphics boards in this decade, creating a "duopoly" in the field which exists this day.

With the introduction of the Nvidia GeForce 8 series, and then new generic stream processing unit GPUs became a more generalized computing device.

GPGPU for General Purpose Computing on GPU, a subfield of research, which perform computation in applications traditionally handled by the central processing unit (CPU), were used to pass data to algorithms as texture maps and execute algorithms by drawing a triangle or quad with an appropriate pixel shader.

Computer graphics become more detailed and realistic each passing day, due to more powerful graphics hardware.

A separate part of the history of 3D modeling is the invention of stereolithography (SLA or SL; also known as stereolithography apparatus or resin printing) is a form of 3D printing technology used for creating models, prototypes, patterns, and production parts in a layer-by-layer fashion using photochemical processes by which light causes chemical monomers and oligomers to cross-link together to form polymers.

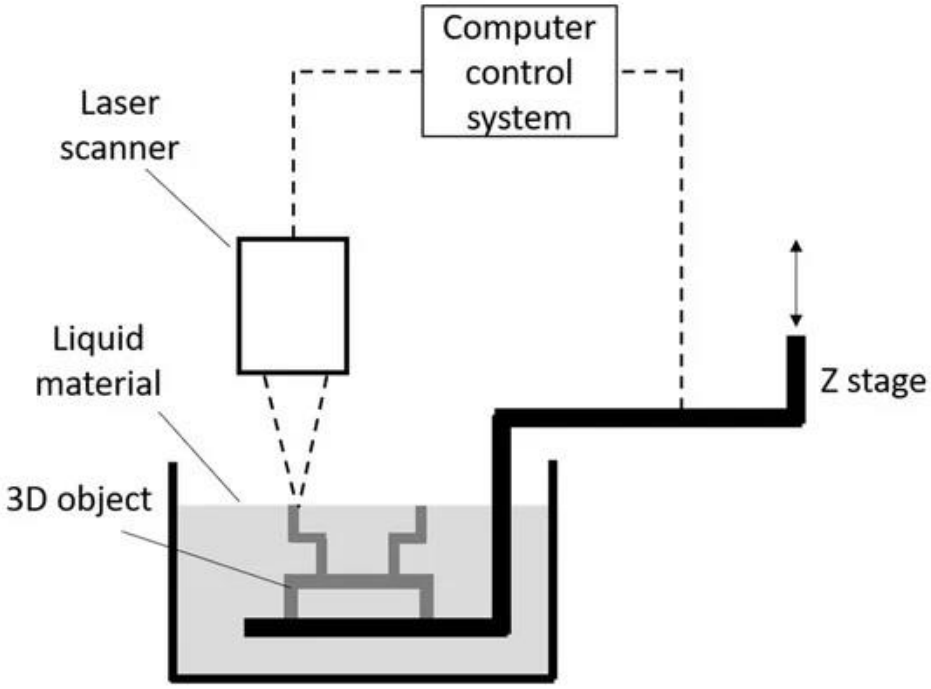


Figure 1.11 3D printer injects liquid on a surface creating layer by layer the 3d object

Stereolithography is recognized as the first commercial rapid prototyping device for what is commonly known today as 3D printing. 3D printing has been identified as a 'disruptive technology', an innovation that has displaced established technologies and created new industries.

1.2 CGI & Art

A long time ago a man, named Alfred Hitchcock, hired another man, John Whitney, to made computer animated opening sequence for his movie Vertigo (1958). He used a strong moving spiral element which correspond to the staircase that triggers Scottie's Vertigo (Figure 1.6, left). Whitney rigged up an anti-aircraft targeting computer called The M5 gun director to a platform. It was composed by 11.000 components and five soldier were required to make it work. Then he placed cells on that platform and used a pendulum to achieve the needed endless rotation.

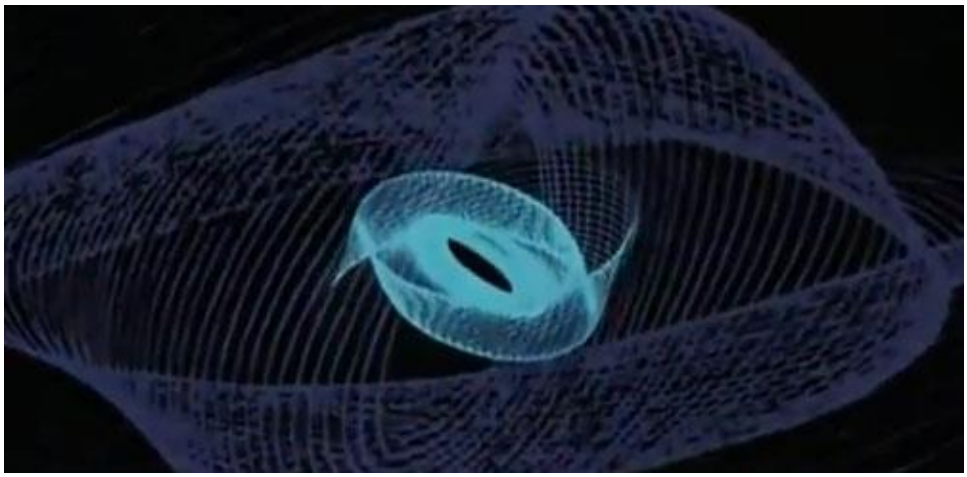


Figure 1.12 Spiral sequence from "Vertigo", A. Hitchcock.

The early computer graphics used configurations of lines representing three-dimensional objects moving to construct film sequences "frame-by-frame".

The first computer animation at the Bell Laboratories was a simple box with edge lines gyrating around a sphere. The sphere should outline the earth and the box should represent a satellite gyrating around the earth.

Another example is a film that presented a four-dimensional hypercube as a rotating "cube within a cube" (1965). A. Michael Noll realized this film using a program of the Bell Laboratories. It exposed one object in slightly displaced perspectives.

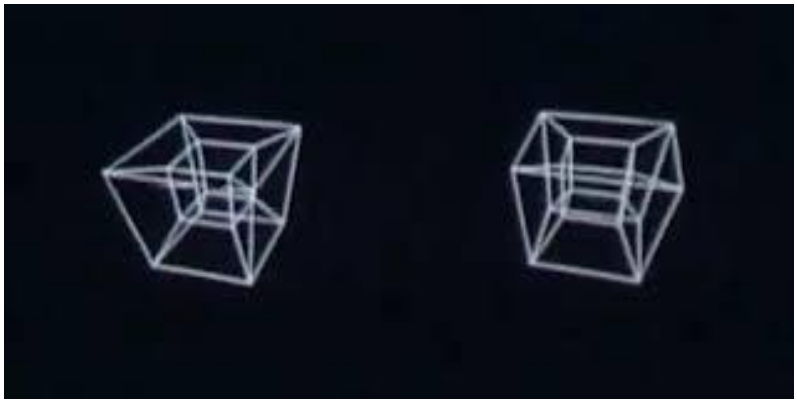


Figure 1.13 Cube within cube from "Hypercube", Michael Noll.

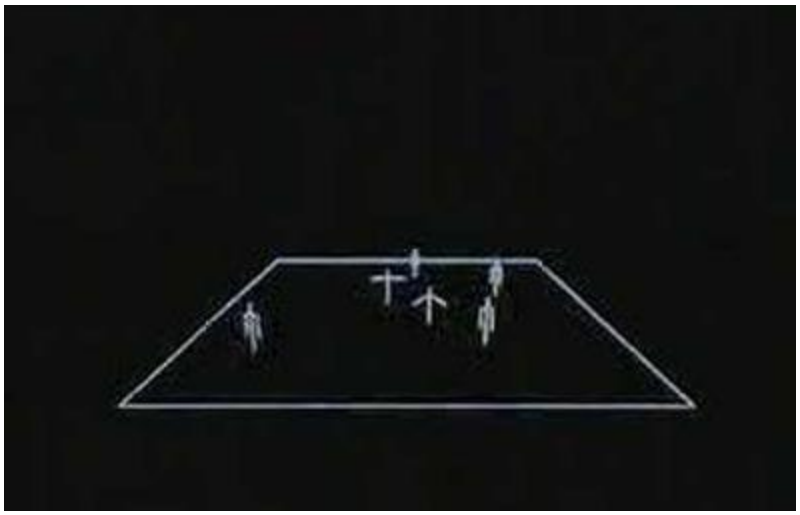


Figure 1.14 Sticks figures dancing from "A computer-generated ballet", Michael Noll.

During these years, universities and labs try to create the very first computer animation languages. In 1965 Ken Knowlton developed at the Bell Labs Beflix, a program capable of drawing basic shapes, lines and letters. Bell Labs was interested in what this technology could do in the hands of an artist.

Using this program, Noll was able to render stick-figures against a black background, or "stage", and program them to "dance". The animation was called "A Computer-Generated Ballet".

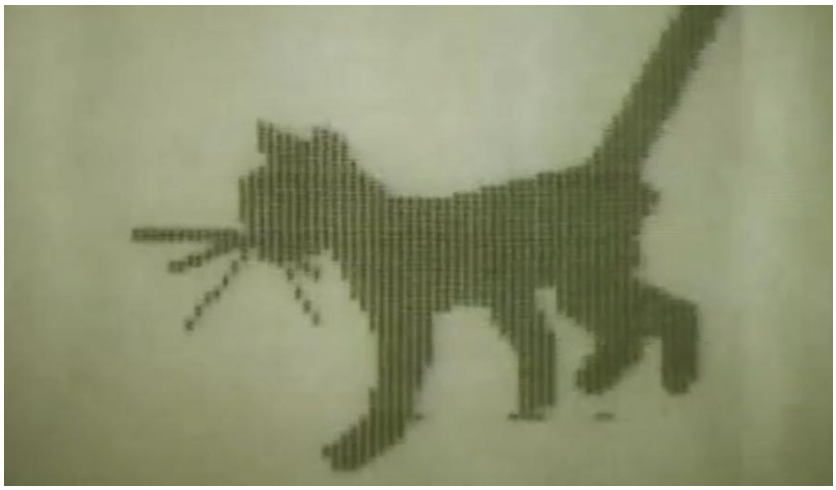


Figure 1.15 "Kitty", Nikolai Konstantinov.

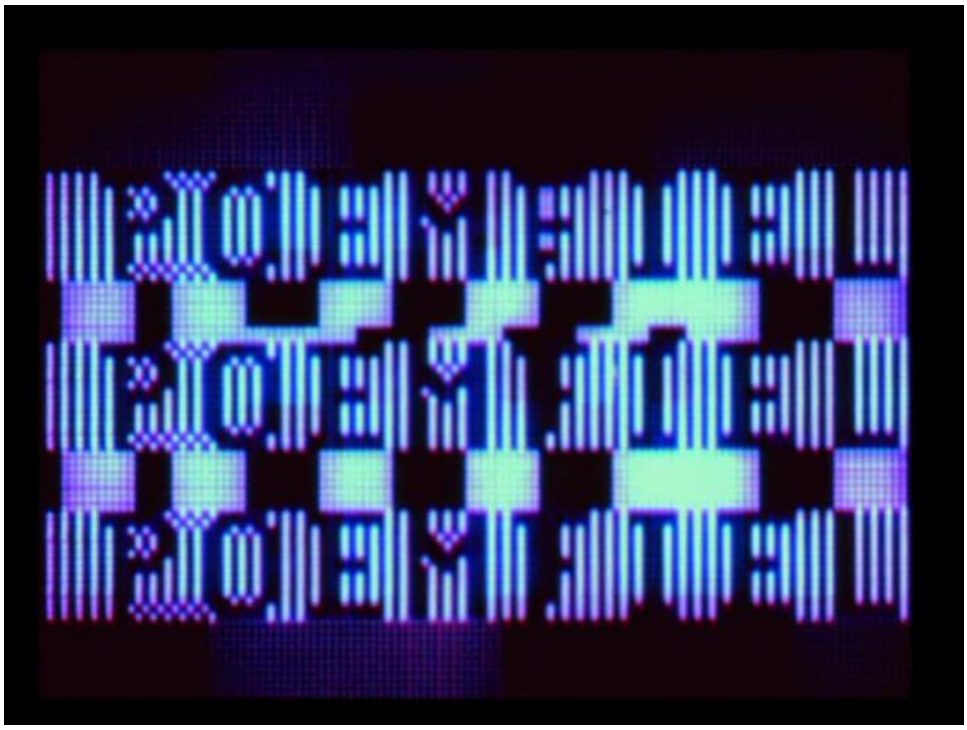


Figure 1.16 "Poemfield", Stan Vanderbeek and Ken Knowlton.

Figure 1.17 "Humming bird", Charles Csuri and James Shaffer.

Beflix was also used to create a series of computer-animated films called Poemfields between 1964 and 1967. The first digital morphing ever featured a hummingbird in a ten-minute computer animated film by Charles Csuri and James Shaffer.

The push towards computer-generated cartoons began when in 1968 Nikolai Konstantinov led a group of Soviet physicists to the creation of an animated cat walking, using mathematical equations.

In the sixties were held the early exhibitions of computer art. Generative Computergrafik, February 1965, at the Technische Hochschule in Stuttgart, Germany, featured work by Georg Nees. Computer-Generated Pictures, April 1965, at the Howard Wise Gallery in New York, featured works by Bela Julesz and A. Michael Noll and was reviewed as art by The New York Times.

A third exhibition was put up in November 1965 at Galerie Wendelin Niedlich in Stuttgart, Germany, showing works by Frieder Nake and Georg Nees. Analogue computer art by Maughan Mason along with digital computer art by Noll were exhibited at the AFIPS Fall Joint Computer Conference in Las Vegas toward the end of 1965.

Few years later, the Institute of Contemporary Arts (ICA) in London hosted one of the most influential early exhibitions of computer art called Cybernetic Serendipity (1968). The exhibition, curated by Jasia Reichardt, included many of the first digital artists, Nam June Paik, Frieder Nake, Leslie Mezei, Georg Nees, A. Michael Noll, John Whitney, and Charles Csuri. One year later, the Computer Arts Society was founded, also in London. One of the first morphing in history was in that exhibition, the computer graphics "Running Cola is Africa" showed a runner transformed first into a Cola bottle and then into Africa´s geographic outline.

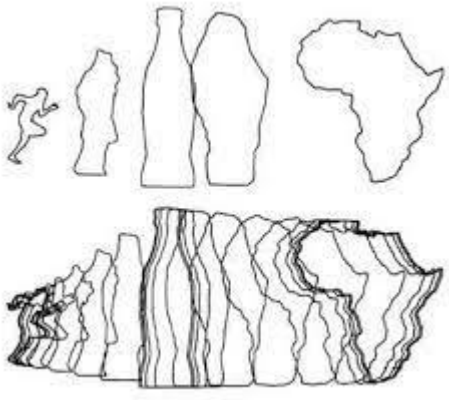


Figure 1.18 "Running Cola is Africa".

The seventies are a turning point for the development of software for three-dimensional animation.

Edwin Earl Catmull and Frederick I. Parke created the animation of a hand, shown three times first as wireframe, then halftone and finally with smooth shading (Figure 1.8, middle) and an animation of a face with Gouraud shading. They documented the animation in the making in "Halftone Animation" in 1972.

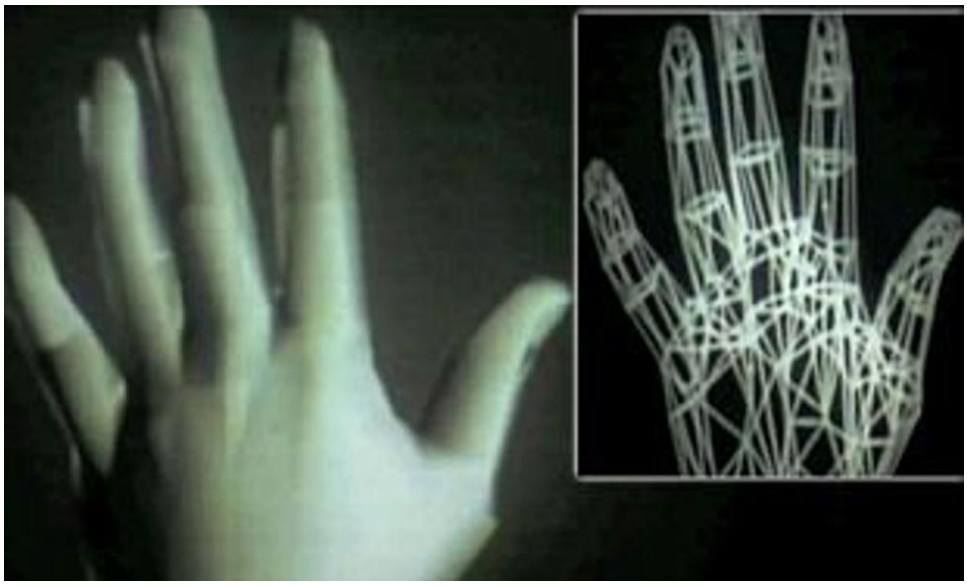


Figure 1.19 Edward Catmull's hand.

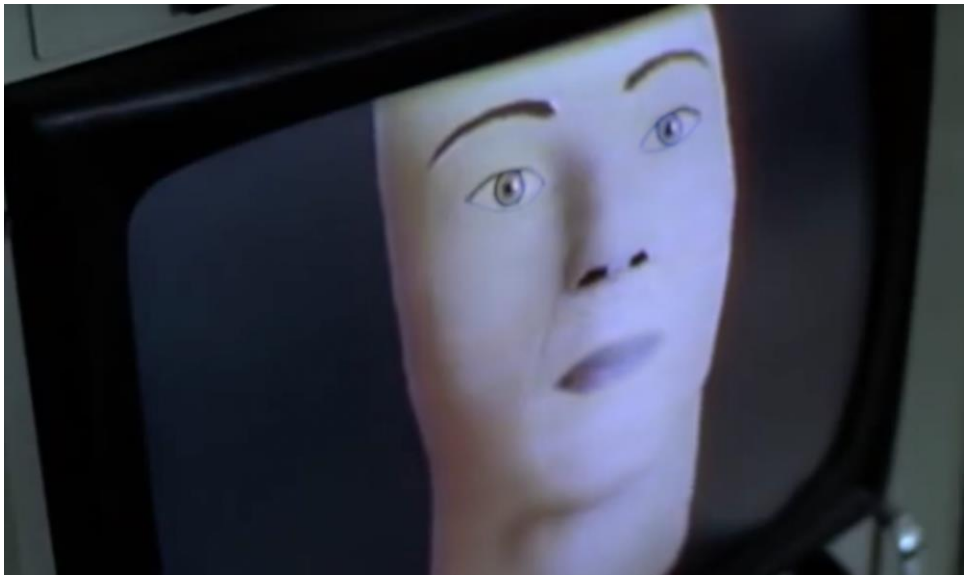


Figure 1.20 Human face model, "Futureworld".

The hand animation was picked up by a Hollywood producer and incorporated in the 1976 "Futureworld", the first film to use 3D computer graphics and a science-fiction sequel to the 1973 film Westworld, which was the first to use a pixelated image generated by a computer as an interpretation of how robots could see ("Gunslinger" vision).

This amazing work was quickly recognized by the Oscars about a decade later with a Scientific & Engineering Academy Award.

George Lucas, founder of Industrial Light & Magic, after creating the first film to fully embrace computer generated animation "Star Wars: A New Hope"(1977) showing a wireframe rendering of the Death Star designed to help give the Rebel Alliance some last minute training, expand the computer animation

division within ILM (Industrial Light and Magic) as he intended to increase its use in the sequel and approached Catmull in 1979 and asked him to lead a group to bring computer graphics, video editing, and digital audio into the entertainment field. Lucas had already made a deal with a computer company called Triple-I, the team behind the effects for both "Westworld" and "Futureworld", and asked them to create a digital model of an X-wing fighter from Star Wars, which they did. In 1979 Catmull became the Vice President at Industrial Light & Magic computer graphics division at Lucasfilm.

Figure 1.21 Scene from "Star Wars: A New Hope" applying CGI to a footage

But "Star Wars" was not the only one to have a unique and avant-garde visual style for the time.

In 1976, animator Steven Lisberger got an idea when he saw the video game Pong.

He imagined a warrior whose world was contained within the game. This idea would become the film Tron, the first science fiction film to focus on virtual reality and first movie of Disney to make great use of computer graphics.

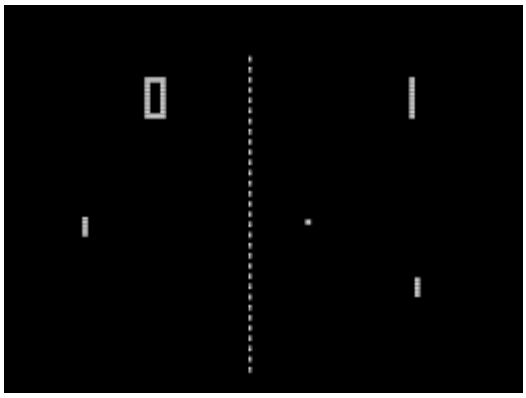


Figure 1.22 "Pong", two-dimensional sports game that simulates table tennis (1972).

Originally intended to be an animated film, computer scientist Alan Kay convinced Lisberger to instead use computer animation.

They enlisted the help of Triple-I and approached Disney with their test animations.

The final look of Tron was a combination of live-action, back-lit and computer animation that is still impressive today.

It was accomplished by filming in high-contrast black and white film with the actors acting against a black background.

This footage was then placed over a light table and re-filmed several times frame by frame using different color filters, with the computer animation matted in.

Figure 1.23 "Tron", Steven Lisberger .

Technology was used not only for incredible futuristic effects, but many artists used it as a media for storytelling. A major strive would be made with Peter Foldes' 1974 short, Hunger. The story, told without words, is a morality tale about greed and gluttony in contemporary society. Foldes uses interpolation, the act of animating the frames in-between key frames to create a smooth transition, to transform a man consumed by gluttony.

*Figure 1.24 M*an consumed by gluttony, *"Hunger"*.

In 1984, Eddie Garrick produced The Magic Egg: A Computer Odyssey, described as "a computer-graphics Fantasia." It was an audio/visual experience created entirely on computer, featuring synthesizers and animation commissioned from artists and

research labs. Contributors included John Whitney, as well as the Computer Graphics Lab at the New York Institute of Technology. At the time, NYIT was the top computer animation research facility in the world. It had been founded with the purpose of not only simplifying the animation process but of taking the medium in new directions.

Tony de Peltrie (1985) tells the story of a faded piano player recalling his past glory.

Tony was the first computer-animated human to convey emotion through facial expressions and body movements. While he very much falls into the uncanny valley, we as an audience are still able to feel his melancholy and his joy.

Figure 1.25 "Tony de Peltrie".

In 1979 and 1981, two first-timers were added to the history of the CGI. First, "Cosmos", a tv-series shot by Carl Sagan, with the first particle system computer animation, then Adam Power's "The juggler" with the first motion capture cgi animation.

Two other pieces of video would also outlast the era as historically relevant: Dire Straits' iconic, near-fully-CGI video for their song "Money for Nothing" in 1985, which popularized CGI among music fans of that era, and a scene from Young Sherlock Holmes the same year featuring the first fully CGI character in a feature movie (an animated stained-glass knight).

Figure 1.26 "Money for nothing", The Dire Straits

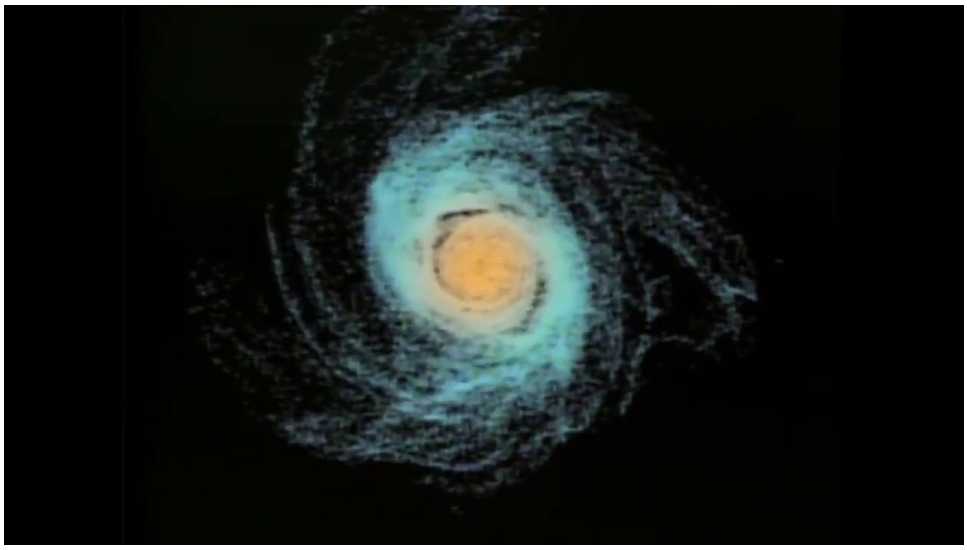


Figure 1.27 "Cosmos", Carl Sagan.

Figure 1.28 " Young Sherlock Holmes"

Figure 1.29 "The juggler", Adam Power (bottom-right).

"Rendez-vous in Montreal", directed by Nadia Magnenat Thalmann and Daniel Thalmann, showed Humphrey Bogart and Marilyn Monroe agree to come back from the dead, meeting at a Montreal Cafe. This marked the first time a deceased actor's likeness had been digitally reproduced, an effect accomplished by mapping out polygons on sculptures of the actors and creating a 3D model.

Figure 1.30 Marilyn Monroe (left), Humphrey Bogart (right)

But it is necessary to jump back in time to see another relevant breakthrough starring the already known, Ed Catmull, and Alvy Ray Smith.

In 1979, NYIT began production on what it hoped would be the first ever fully computer animated narrative film, titled The Works, but despite having the most powerful computers available, they abandoned the project because it still would have taken seven years to generate enough images needed for the planned 90-minute run-time. NYIT's goal of creating the first computer animated narrative would eventually be achieved in Montreal. Nadia Magnenat Thalmann, Philippe Bergeron and Daniel Thalmann had been experimenting with 3D modelling, specifically 3D character modelling. In 1982, they

produced the 8-minute short, Dream Flight, the story of an alien dreaming of flying through space and arriving on Earth.

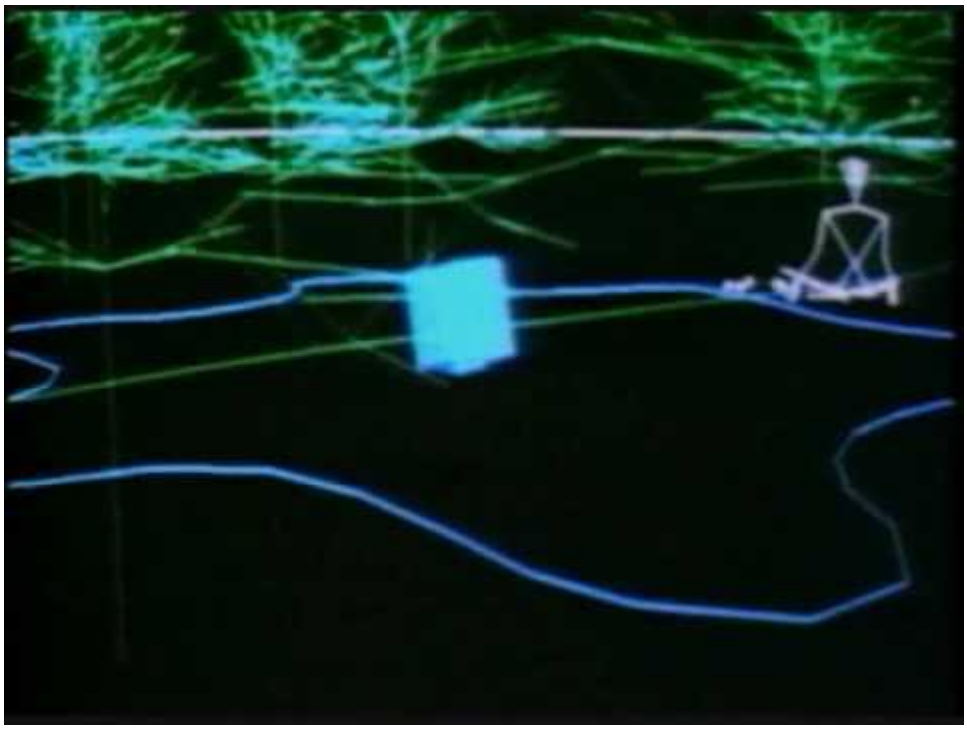


Figure 1.31 "Dream flight", Nadia Magnenat Thalmann, Philippe Bergeron and Daniel Thalmann

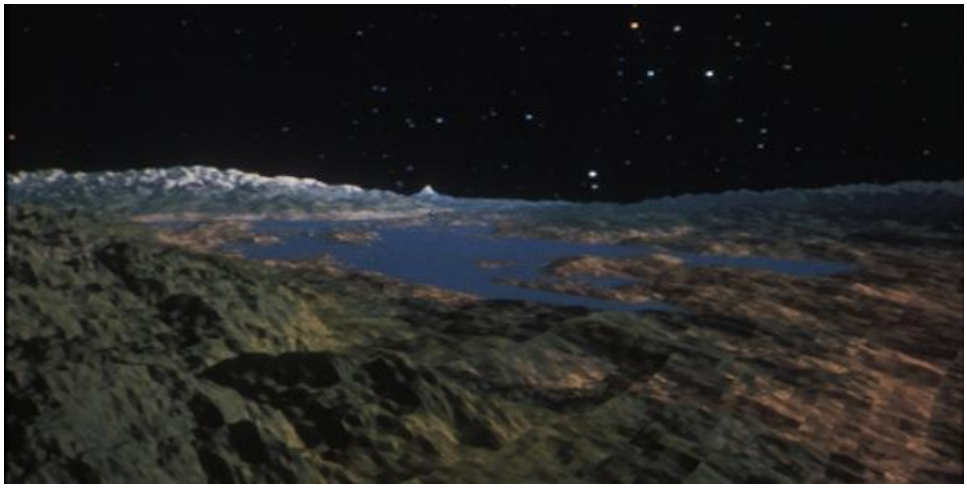


Figure 1.32 " Star Trek 2: The Wrath of Khan", Loren Carpenter.

Ed Catmull and Alvy Ray Smith had made several breakthroughs at NYIT, most notably, alpha compositing, but when it was clear that "The Works" was a doomed project, they joined ILM to produce the Genesis one-minute demo film sequence in Star Trek 2: The Wrath of Khan in 1982, the first completely computer-generated cinematic image (CGI) sequence.

The sequence presented a planet being revived by a "Genesis Torpedo". The "genesis effect" scene featured an entire fractally-landscaped planet produced by Loren Carpenter.

Agitated surfaces representing fields of heat energy among others were generated with particle systems.

A couple of years later, Alvy Ray Smith conceived and directed in 1984 "The Adventures of André & Wally B", the simple story of a funny character named André (inspired by the appearance of Mickey Mouse) and a pesky bee, Wally B. The short featured impressive animations for the time, thanks also to the unprecedented use of motion blur in CGI, while the rendering process was done thanks to Cray X-MP computers.

The short was a complete success and gave the opportunity to a division of Lucasfilm to break away from the company from the company with funding from Apple founder, Steve Jobs, and being independent under the well-known name of Pixar.

Under Jobs, they would release a commercial version of their hardware system known as the Pixar Image Computer and produce a series of animated shorts intended to promote the product's capabilities. unfortunately, the sales didn't go well, even though one of their clients, very pleased with their purchase, was the famous Walt Disney Company. Their Feature Animation department utilized the hardware for its Computer Animation Production System, or CAPS, which allowed hand-drawn animation to be digitally inked and colored.

The first Disney feature to be entirely produced with this process was The Rescuers Down Under (1990), followed by Beauty and The Beast one year later.

Beauty and The Beast also made use of CAPS ability to blend hand-drawn and computer animation. The film's famous ballroom scene finds Belle and Beast dancing through a 3D-environment.

Figure 1.33 "The Adventures of André & Wally B", Alvy Ray Smith

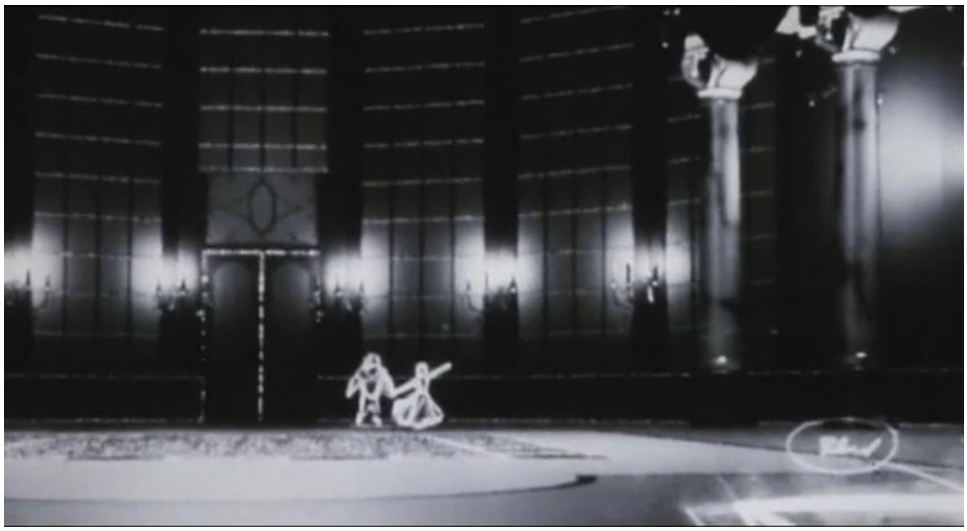


Figure 1.34 Ballroom scene, " The Beauty and The Beast".

Disney executives were impressed with this sequence and decided to invest more resources into computer animation. So, Steve Jobs agreed to finance Pixar's Tin Toy, despite the financial struggle caused by the poor sales of the Image Computer. The short was intended to showcase their Renderman software, which allowed photorealistic 3D rendering, and starred the toy of the title, a mechanical one-man band player named Tinny, and an infant named Billy.

The film won the Oscar for Best Animated Short, the first computer-animated film to do so, and gained so attention from Disney, that they approached Pixar with the possibility of producing a feature-length computer-animated narrative, which had long been a goal of Catmull and Smith.

After tense negotiations, they agreed to produce a film based on Tin Toy, entitled Toy Story.

Figure 1.35 "Tin toy" (left). "Toy story" (right).

The plot and characters went through many changes, with the original version being so dark and mean-spirited that the production was even temporarily shut down. Production restarted in February 1994, now with the story of two toys fighting for the attention of their owner. The film featured the voice talents of Tom Hanks and Tim Allen as pull-string cowboy Woody and astronaut Buzz Lightyear, who throughout the film go from rivals to the best of friends.

In order to render "Toy Story", the animators had 117 computers running 24 hours a day. Each individual frame could take from 45 minutes to 30 hours to render, depending on how complex. There was a total of 114,240 frames to render. Throughout the movie, there are over 77 minutes of animation spread across 1,561 shots.

There were a lot of limitations. For instance, at this time, Pixar hadn't quite figured out how to fully animate human characters. Animating clothes was time-consuming, so you'll notice a lot of shots of hands and feet in the movie from a toy's perspective. Additionally, they would sometimes choose not to fully light characters, being able to cover missing details. It took years of technological breakthroughs to get where they are now.

Getting a given character to move starts in rigging and modeling. A 3D model needs a rig, which is essentially an inner skeleton, and it gives to the animators a set of controls used to make different parts of a character move around.

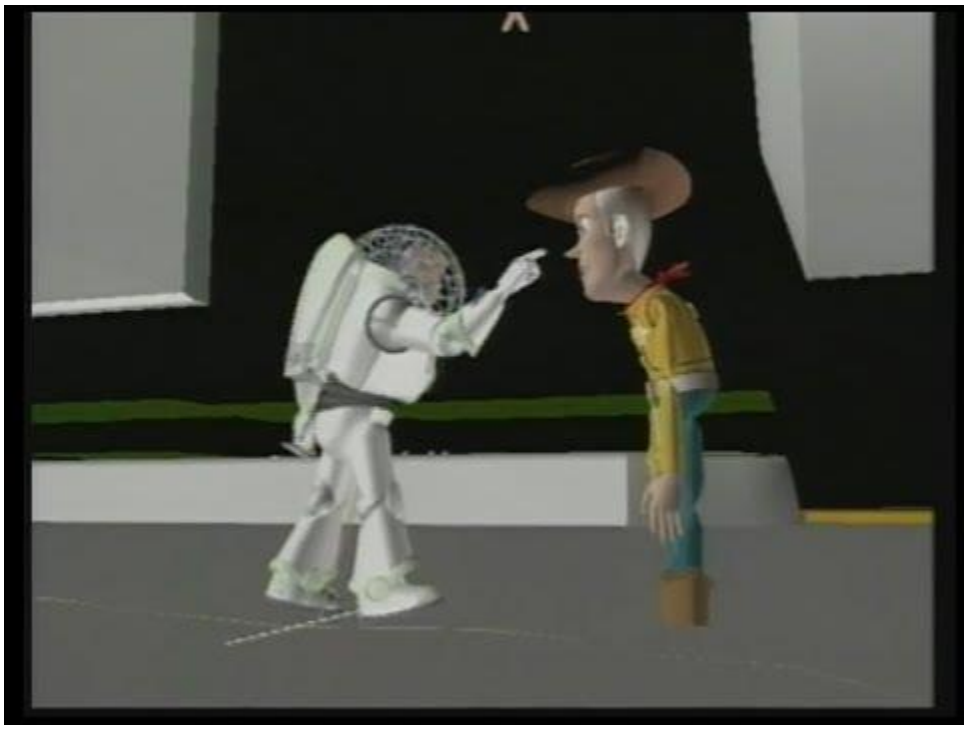


Figure 1.36 Buzz Lightyear and Woody.

Figure 1.37 Woody face rig.

The early version of Woody had 596 controls, which may seem like a lot but by 2019's toy story 4 Woody had more than 7 000.

To make things a little more complicated one control point is often connected to other control points, for example moving the eyebrow will influence the wrinkles on the forehead. Increasing the number of controls to the characters would have added realism and details to their movement but back then Pixar was making a brand-new rig for every single character, costing a huge amount of work and time.

With Toy story 2, things were different. Pixar introduced Geppetto, later updated to Presto which allowed animators to reuse and adapt rigs for multiple characters, speeding up the animation process and giving animators more control over movements and facial expressions. The sea lions in "Finding Dory" were constructed from dog rigs with legs folded into flippers.

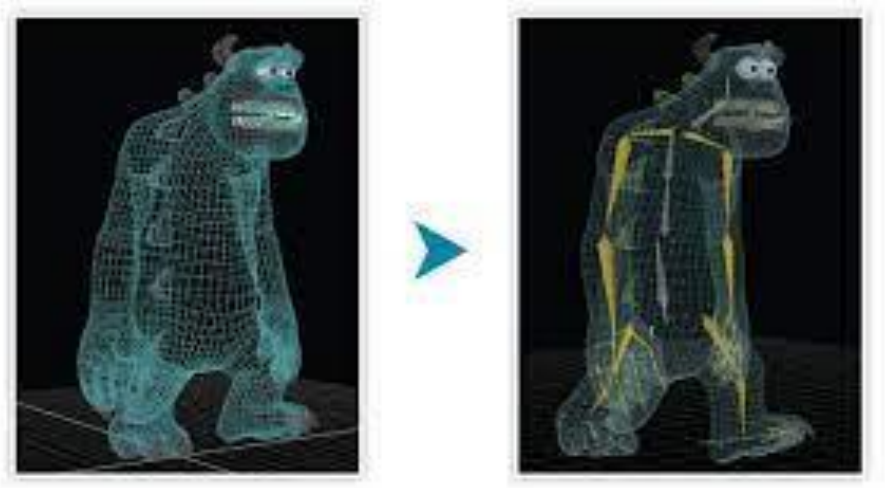


Figure 1.38 Geppetto applied to James Sullivan, "Monster & co."

Figure 1.39 Sea lions, "Finding Dory".

After making major strides with toys and sea creatures the most difficult challenge was yet to come humans.

Mr. Incredible, from The Incredibles (2004), consisted of 426 primary controls, 111 secondary animation controls for more subtle movements and 1061 modeling controls for sizing and other adjustments. To really make his movements realistic they used full body reference footage. Humans also have muscles which could bulge and flex and make the skin move, so Pixar developed a new system called Goo which allowed animators to integrate this kind of behavior in the characters, adding more control and realism to the movements.

For the movie "Cars", Pixar developed a system called Ground locking, which made sure the cars always stayed on the same path regardless of terrain without animators having to move them frame by frame.

Figure 1.40 Mr. Incredibles, "The Incredibles". Demonstration of the system Goo developed by Pixar

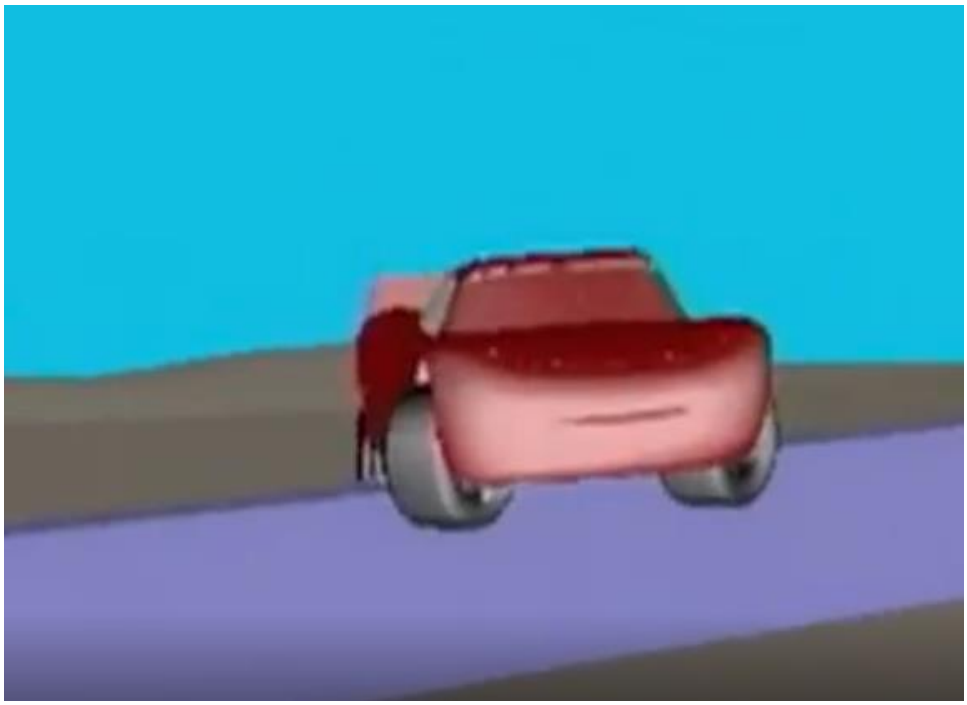


Figure 1.41 Saetta McQueen, "Cars".

By 2007 the number of controls in the human face went up to 150, so animators could give characters a much wider range of emotion. When Toy story 3 came out in 2010 Pixar's animators had gotten such precise control over how characters move, however there was one particularly hard toy to animate the octopus named Stretch. Despite the innovations and stretchiness from the Incredibles, making eight tentacles stretch and curl was still a daunting task. But it was a challenge they would eventually win with Finding Dory. Hank, the octopus in Finding Dory, needed to move around quite a bit and grab onto things and had more than 350 suckers combined on his tentacles. Those were so unpredictable that they had to be simulated with hair particles.

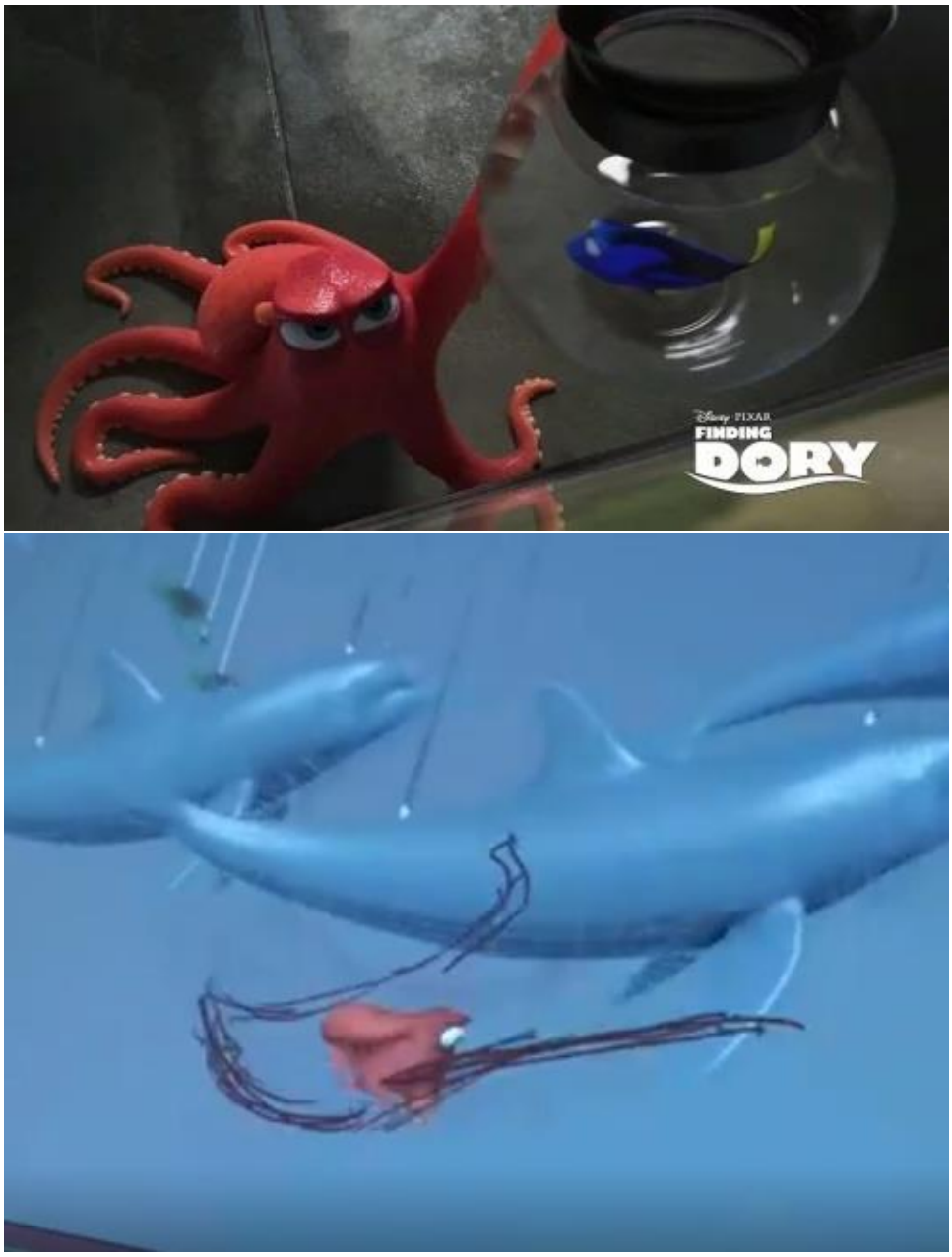


Figure 1.42 Hank, "Finding Dory"

In 2017 they produced Coco, which came with a new big challenge: Music. They had to figure out, how to Miguel, the protagonist, to accurately play guitar. They used reference video crafted for this purpose and mimicked the exact movements on the hand of the model in sync with the music. But they pushed it even further in Soul (2020) and make playing music look even more real. For the scene where Joe plays the piano, they went far beyond just matching the music directly to audio. They built a piano rig, which functions like a digital piano, they fed the audio files directly into the rig and the piano would play automatically in time with the music. They also developed a finger contact rig like the one used on Coco, in this way when Joe's finger intersected the space of a key it would calculate the proper angle that piano key needed to rotate to make it look like he's pushing it. While the piano played on its own,

the hands still needed to capture the complicated motions of playing which is why Joe was equipped with 292 controls in each of his hands.

Figure 1.43 Coco playing guitar, "Coco"

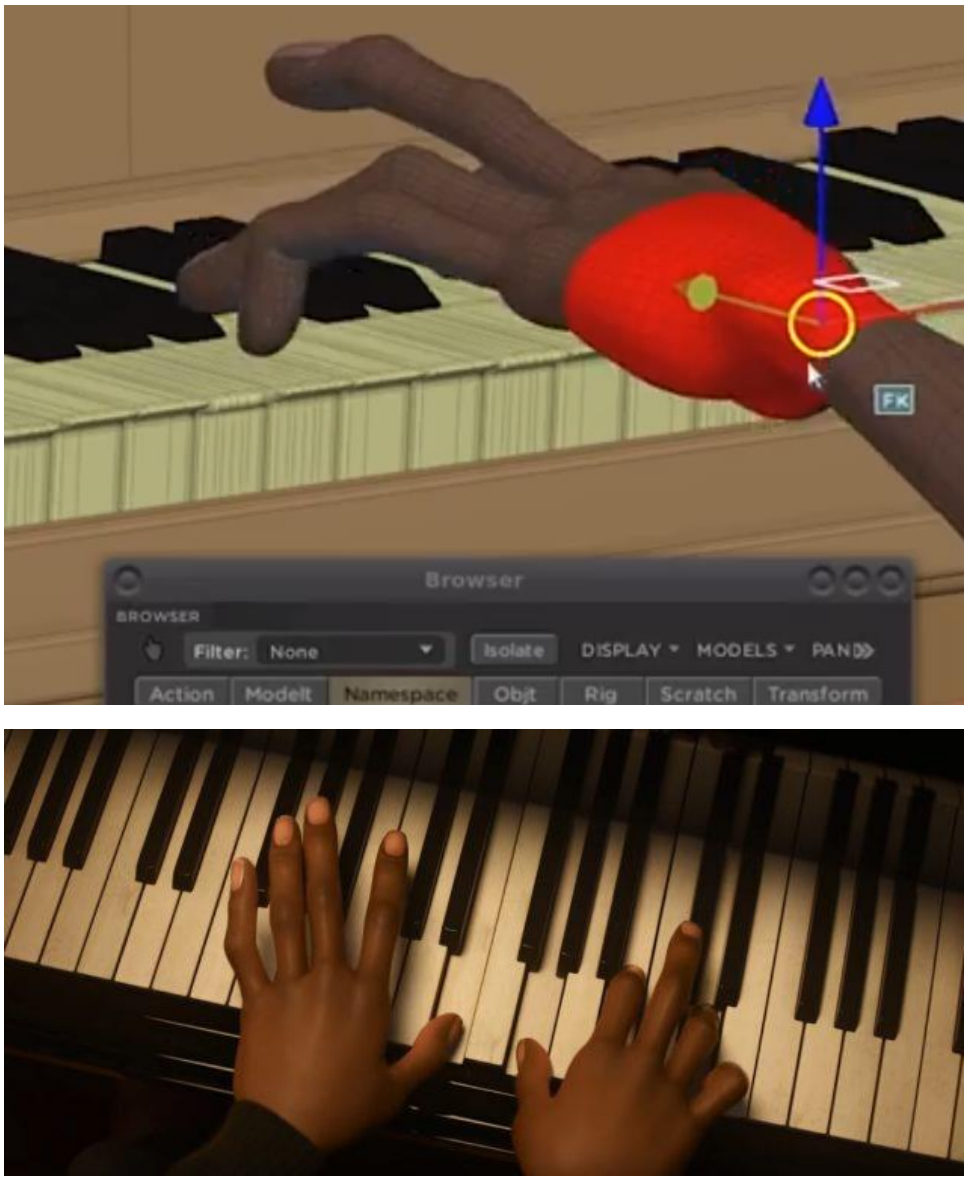


Figure 1.44 Joe playing piano, "Soul" (right)

From 2010 to 2019, technology changed in major ways, which had an enormous impact on the movies.

From realistic environments to convincing fake animals, to aging and de-aging actors and even doubling them, it almost felt like there was nothing CGI could not do.

Armie Hammer played twins Cameron and Tyler Winklevoss in "The Social network" (2010), but actors playing double is nothing new.

Typically, when an actor plays two characters that interact, filmmakers use techniques like split screens and over-the-shoulder shots, which this movie certainly used. When the twins needed to be in the same scene, the movie used two different actors. Actor Josh Pence served as Armie Hammer's body double; Hammer's face was then grafted onto Pence's body in post-production. The two would switch off who would be Cameron and who would be Tyler from scene to scene.

Figure 1.45 CGI doppelgänger, "The social network"

Face replacement did not start with "The Social Network", but this movie was, at the time, the first utterly convincing version of the technology, especially since it was a movie with no sci-fi elements.

These techniques paved the way for an entire movie about doubles in the form of "Us"(2019) and for Chris Evans to fight himself in "Avengers: Endgame"(2019).

Lola Visual Effects took what it learned on "The Social Network" and applied it to the Marvel Cinematic Universe, in particular the face replacement in "Captain America: the first avenger"(2011).

Before becoming into Captain America, Steve Rogers is small and scrawny, nothing like actor Chris Evans.

For some shots, actor Leander Deeny doubled as the skinny Steve Rogers. For others, the VFX team replaced Deeny's entire head with Evans' head. First Evans would shoot the scene looking like himself. Deeny would watch Evans perform the scene and precisely mimic his every move. The crew would shoot a clean plate without characters. Then, VFX artists would place Evans' face over Deeny's.

But for most of these scenes, the VFX team digitally shrunk Evans down.

For this, Lola Visual Effects invented a new method for shrinking, which it called "Steve slimming."

Lola Visual Effects' artists stated that they learned a lot from this experience, like how to highlight an actor's performance without letting CGI get in the way.

This lesson turned out to be extremely useful in case of de-aging.

"Captain America: Civil War" featured a de-aged Robert Downey Jr. as Young Tony Stark. Some cases of de-aging involve a body double, and others involve replacing the actor with a digital double.

Figure 1.46 Chris Evans' face replacement, "Captain America: the first avenger".

Figure 1.47 Robert Downey Jr.'s de-aging, "Captain America: Civil War".

However, innovative technology allowed Downey to perform the scene normally as himself. VFX artists used digital compositing to de-age the actor. This way, Downey could still perform the scene as usual and the VFX team would not lose the actor's many nuances, which is a technological achievement that helped three years later in "The Irishman"(2019). One of the biggest challenges of de-aging a well-known star is that audiences are familiar with how they looked when they were younger, especially with an actor like Downey, who has been in movies since he was a child. So the VFX team analyzed old footage of him in 1987's "Less Than Zero," which helps explain why this young version of Tony Stark was so convincing.

Technology can also achieve the inverse process: aging.

Hayley Atwell was aged up for a pivotal scene where she plays a much older Peggy Carter. Five years later, Captain America would finally get to act his age in "Avengers: Endgame," as Chris Evans was aged up using a mix of CGI, makeup, and a body double.

If the visual appearance does not seem a problem anymore, there is another issue more subtle that need to be solved: de-aging an actor's movements. It may be a CGI breakthrough for the 2020s.

Figure 1.48 Robert De Niro's de-aging, "Irishman"

Figure 1.49 Chris Evans's aging, "Avengers: Endgame"

Thanks to motion-capture technology, actors could show their full range of emotions while being transformed into any kind of creature. Mo-cap was refined in the 2000s in the "Lord of the Rings" trilogy, "King Kong," and "Avatar." It became commonplace in the 2010s, hitting its stride over the course of the "Planet of the Apes" reboot trilogy. Much of the work on all these movies was done by the famed visual effects studio Weta Digital.

By the time of "Rise of the Planet of the Apes," cameras were able to capture the most subtle of movements and details, which made the apes more believable. And it was also the first movie to ever shoot motion capture in a live-action location, as opposed to just a bare soundstage.

After Gollum ("The lord of the rings" trilogy), Cesare ("Planet of the Apes"), Kong ("King kong"), Andy Serkis played the menacing Supreme Leader Snoke, showing up as a hologram in "The Force Awakens" and then in person in "The Last Jedi"(Star Wars).

By the time "The Last Jedi" rolled around, they were able to add so many details to the skin, like age spots and veins.

The visual effects team shot Serkis with high-resolution cameras positioned at every conceivable angle to create a digital clone of him. As Serkis performed, they could automatically capture every subtle facial move, from how he smiles to how he furrows his brow. They even studied the elderly to see how light would realistically hit Snoke's skin.

Figure 1.50 Andy Serkis' facial capture, "Rise of the planet of the apes".

Figure 1.51 "The jungle book"

In the live action "Jungle Book" remake, actor Neel Sethi played Mowgli and was the only human character in the film. He acted on a completely empty soundstage, besides puppets and creature performers that would then be digitally replaced with the actual CGI characters.

The biggest breakthrough on this movie: muscle movement. MPC Film developed new software for animating an animal's muscle structure.

Developed by George Lucas' Industrial Light & Magic, it was pioneered in the production of "The Mandalorian". It is a new way of thinking about the set, with LED screens that reproduce the background and adapt to the movements of the camera.

The actors could see their surroundings, but the surroundings were not actually there.

All of this is just LED screens displaying backgrounds pre-made in a video game engine. LED walls make the lighting better, filming smoother, and in certain cases, cost a lot less than using green screens.

The idea is not entirely new, the predecessor to what is seen on "The Mandalorian" is a driving scene like this one, from "Dr. No".

There is the actor in the car and a screen with footage of the road moving, creating the illusion that actor is driving.

But the technology was limited, changing the camera angle was impossible. The projected footage can't move with the camera.

But by using Unreal Engine, tech borrowed from the video game field, that problem is solved.

Artists can create a photorealistic 3D background that moves strictly with the camera's field of view, known as the Frustum.

So, if the camera moves and rotate, the background follows camera movement very precisely and motion-tracked cameras can execute traditional cinematography techniques within the virtual set, achieving cinematic movements like the parallax effect, where an object in the foreground moves at a different speed than the background, amplifies the illusion of filming at an actual location.

Lighting is one of the key benefits of working with virtual sets.

The light coming from the LEDs provide realistic colors and reflections on the actors and props, something exceedingly difficult to achieve with green screen.

1.3 Fields of application of 3D

1.3.1 Manufacturing & Engineering

Whether it is reverse engineering or conceptualization of an entirely new product, 3D modeling is extensively used to improve the design, re-fabricate parts, and rush products to market. 3D modeling has also the opportunity to go beyond the creation of virtual models. With the advent of reliable, multi-media 3D Printing technologies, it is not so strange to imagine that virtual models can be converted in real product.

Automobiles	Jewelry	Spare & Replacements Parts
Aerospace	Glasses and Eyewear	Shoes
3D Food printing	Fashion and Smart Clothing	3D Bioprinting

1.3.2 Healthcare

3D-printed models have been used in many medical areas ranging accurate replication of anatomy and pathology to assist pre-surgical planning and simulation of complex surgical or interventional procedures, the creation of medical tools and equipment, custom-made prosthetics.

Audiology	Dentistry	Prosthetics
Surgery	3D Printed Organs	

1.3.3 Construction

As an industry already based on geometric design, prototyping and modeling, architecture stands to gain enormously from advances in 3D technologies. The possibility of preview an entire construction could be a useful tool in real estate, too. Customers can see their home even before the first brick is placed.

The limit of 3D printing lies in the production of smaller objects, but, perhaps one day, it will be possible to create homes and large structures that are entirely the product of 3D printing, opening new frontiers in sustainable living and construction.

Architecture	Real Estate	Housing and construction

1.3.4 Art & Entertainment

Art & entertainment are probably the fields that use most the advantages of 3D technologies. But visual effects are not the only subject 3D can be used for, it is a useful tool to create photographies and pieces of art, such as sculptures, that can be 3D printed. It also has the power to bring entirely new dimensions to forms like dance and music. Using advanced sensing technologies, they transform movement, orientation, and touch into music.

3D printing can be considered also for manufacture musical instrument, especially the ones that need a certain level of perfection like violins or personal customization.

Finally, 3D technologies have been used to restore the works of some of history's most famous artists, returning works by the likes of Michelangelo and da Vinci to their former glory, but they are also useful to recreate works of art or places that does not exist anymore.

Sculptures	Photography	Art Restoration
Movies, VFX and Gaming	Dance and Music	Musical Instruments

1.3.5 Research

Digital technologies can be of tremendous use in legal investigations and can augment the abilities of forensic artists to reconstruct accurate models of persons of interest, victims, or crime scenes. Forensic facial reconstruction is a method used in forensic anthropology to aid in the identification of skeletal remains.

Paleontologists can also benefit 3D printing, as it may help complete dinosaur skeletons by printing elusive missing bones. The Smithsonian Museum staff experimented recently by printing the missing bones of a T. rex exactly to specification.

The studies of linguistics, philology, literature, history, archeology, history of visual arts, musicology, human-machine interaction, librarianship and the teaching sector are field of application of the digital humanities, which integrates computational procedures in disciplines that traditionally uses other approaches.

Forensics	Paleontology
Digital humanities	

UNDERSTANDING THE FUNDAMENTALS OF 3D IN BLENDER

This chapter will introduce the basic knowledge to understand better the tools and methods used during the project.

In Blender, any project starts with a default set up with a cube, a light, and a camera. These objects are visible in the 3D viewport, a window that allows you to view and interact with the entire 3D scene. It serves many purposes, allowing you to sculpt, paint, animate, deform, move elements, and position cameras and lights, but one of the most important is navigating the scene. You can also use Camera view to preview what will be rendered by the camera.

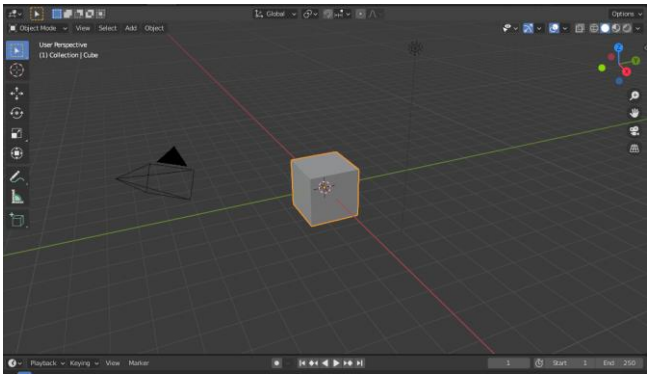


Figure 2.1 3D viewport full view.

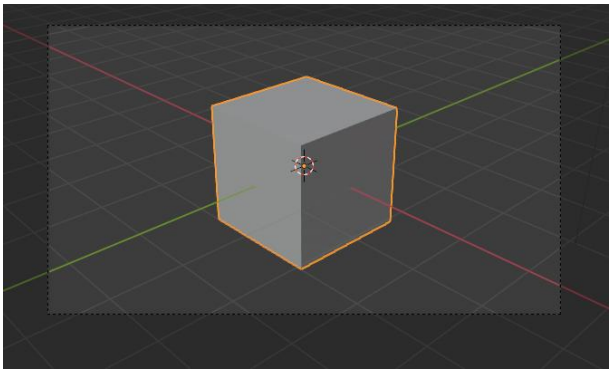


Figure 2.2 3D viewport camera view.

A 3D scene can be composed of many objects, whose visibility can be disabled in the 3D viewport or during the rendering phase by the user, but there are also some types of objects that are not rendered by default.

Each object stores a data structure composed by two parts:

1. Object: it holds information about the position, rotation, and size of a particular element.

2. Object Data: it holds everything else, such as mesh, modifiers, actions.

To determine where the object is in 3D space, each object has an origin point that will be used for this purpose. The origin point of each object can be modified. The origin point is considered by the software as the center of gravity, and it can be used as a pivot point for rotating an object.

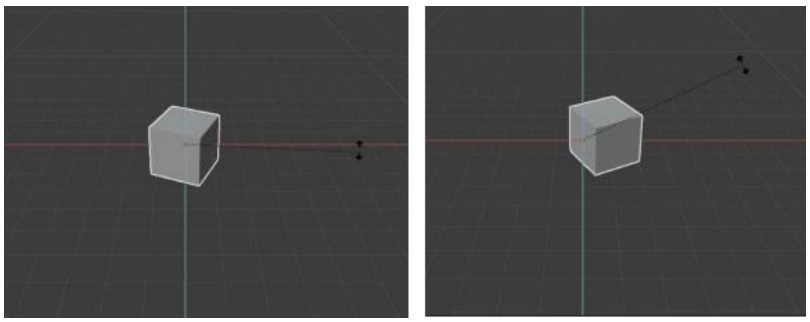


Figure 2.3 The origin point used as a pivot for rotating the object on the Z axis. First on the center of the object (left), then on the center of one of the faces (right).

Each Blender object has a type:

- Mesh: objects composed of vertices, edges, and polygonal faces.
- Curve: mathematically defined objects which can be manipulated with control handles or control points (instead of vertices), to edit their length and curvature.
- Text: a two-dimensional representation of a text.
- Empty: null objects that are simple visual transform nodes that do not render. They are useful for controlling the position or movement of other objects.
- Light: empty objects that emit light and are used for lighting the scene in renders.
- Camera: object used to determine what appears in the render.
- Force Field: objects that give simulations external forces, creating movement, and are represented in the 3D Viewport as small control objects.

The 3D viewport allows to display / shade objects in the scene in 4 modes:

- Wireframe: only the edges of the objects are displayed.
- Solid: the surface of the objects is rendered using the Workbench engine, which is used to shade and light it and assign random colors to objects.
- Material preview: instead of using default color, the software will apply the materials to the model, that is the set of information that determines its appearance.
- Rendered: the entire scene is displayed using the Render engine selected for rendering the images.

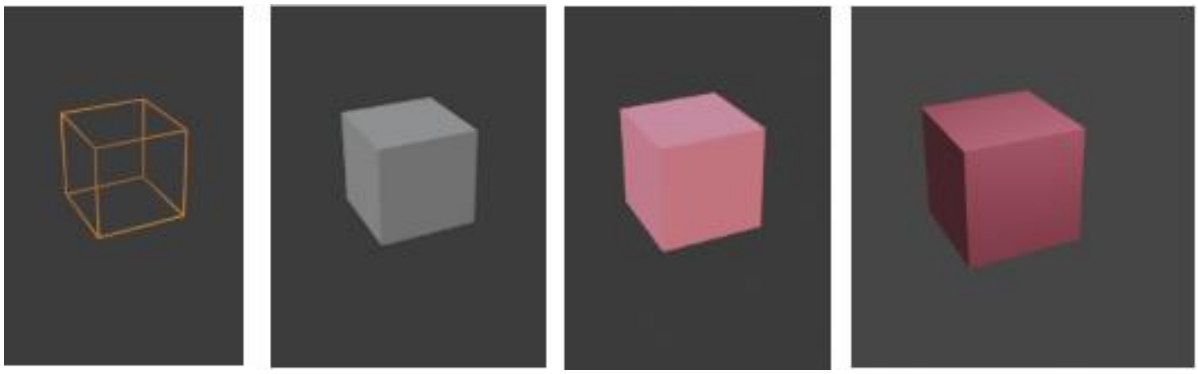


Figure 2.4 A cube is displayed as wireframe, solid, material preview and rendered.

The functioning of a standard graphics system is typically described by an abstraction called the graphics pipeline. The term "pipeline" indicates a process that involves multiple steps, which are performed in sequence in a typical architecture, and usually involves:

1. Modeling
2. Shading
3. Lighting
4. Animating
5. Rendering

2.1 Modeling

The starting point of any more or less complex object are those shapes that can be automatically generated by the software, called Primitive meshes, and they include cubes, spheres, planes, cylinders, torus, etc. Meshes are composed by vertices, edges, and faces.

Typically, models are created in a convenient coordinate system; a cube that is to be used as one of a pair of dice might be modeled as a unit cube, centered at the origin in 3-space, with all x, y, and z coordinates between −0. 5 and 0. 5. This coordinate system is called modeling space or object space.

This cube is then placed in a scene, a model of a collection of objects and light sources, and it has global coordinates which are said to be in world space.

The attributes of virtual cameras and lights are also linked to the world space coordinate system. Cameras are used to render 3D model into a 2D plane, so all objects in world space must have the corresponding coordinates in camera view, which captures only a portion of a scene; these coordinates are called camera-space coordinates or simply camera coordinates.

These camera coordinates are transformed into normalized device coordinates, which decide which objects are inside and outside the camera view based on their distance from the camera and solve the

issue of overlapping objects. Finally, the visible fragments are transformed to pixel coordinates to visualize the image on the display.

A vertex is the lowest-level component that makes up a 3D model. Each point exists in 3D space with specific coordinates on X, Y, Z axis. When two points are connected, a line is drawn between them. When three or more points form a closed broken line, they can become corners of surfaces on a model called a polygon (or simply face). A triangle, for example, consists of three points and one polygon. Multiple polygons can share the same points when used on a contiguous (seamless) mesh. Every point in an object stores information about its position. Points also can store a variety of additional information using vertex maps.

Texture, or UV, maps store texture placement information. Weight maps store a single value defined as weight. The most common use of a weight map is for defining a bone's influence on a point when rigging.

Morph maps (or Shape keys) store offset information (alternate XYZ values) for a point's position and are commonly used for associating different shapes with a single mesh.

Color maps hold values for Red, Green, Blue, and Alpha (RGBA) color information associated with the vertex.

An edge is a one-dimensional line that connects two points in a polygon. Like points, multiple polygons can share the same edges when used on a contiguous mesh. Edge weighting increases or decreases the sharpness of an edge between two subdivision surface polygons, allowing for harder or softer corners without additional geometry being added.

Faces are geometric shapes consisting of several points that define the surface of a 3D object, and they are what is rendered by the software.

The organization, flow and structure of these components is commonly called Topology.

The topology of meshes defines what is connected to what and is of primary importance in many mesh algorithms, which often proceed by adjacency search like depth-first or breadth-first search.

A good topology is usually made of faces quads and as even-sized squares as possible and with the least number of vertices.

Good topology makes geometry easy to select, manipulate, and construct, and even better for sculpting onto. Also, generating and using UV maps are easier with clean topology.

It makes deformed animations look better, is not affected by weird artefacts (pinching and stretching), get better result with modifiers, uses less memory.

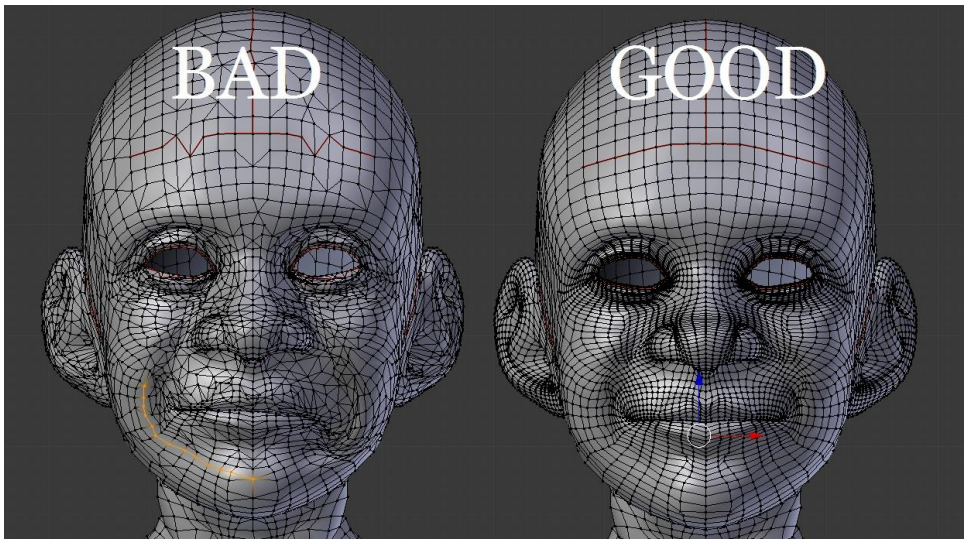


Figure 2.5 A good topology vs a bad topology.

Meshes are not the only object type which can be used for modeling, there are also Curve objects that have completely distinctive characteristics.

A spline is a curve in 3D space defined by at least two points. The most common spline used in digital modeling is the Bézier curve. Bézier curves are used to model smooth curves using far fewer points than a polygonal model would require. Control points make up the curve and can be used to dramatically manipulate the curve with little effort.

A Non-Uniform Rational B-Splines (NURBS) surface is a smooth mesh defined by a series of connected splines, which are polynomial curves. This smooth surface is converted to polygons at render time, so NURBS surfaces can contain an arbitrary number of polygons. NURBS can be converted to Mesh objects and are useful for constructing many types of organic 3D forms because of the minimal nature of their curves. NURBS geometry is smooth by default and doesn't need to be subdivided to "become" smooth like polygon geometry does.

As already said, Blender provides a lot of tools for modeling, but the way you can interact with the object depends on the Mode currently enabled on the 3D viewport.

Each mode is designed to edit an aspect of the selected object.

- Object mode: it is the default mode, available for all object types. It is dedicated to applying transformation such as translation, rotation, and scale to an object in the world space, considering the entire mesh as a single point (the origin point).
- Edit mode: this mode lets you to manipulating mesh data (vertices, edges, faces, normals) in a local coordinate system, directly selecting the parts of the mesh which needs to be manipulated.
- Sculpt mode: it is similar to Edit Mode because it has the same purpose, but a different approach: instead of dealing with individual elements (vertices, edges, and faces), an area of the model is altered using Blender's mesh 3D-sculpting tools, called brushes. In other words, instead of selecting a group of vertices, Sculpt Mode manipulates geometry in the brush region of

influence. Brush options include the type of brush, radius, strength. Brushes can be also customized, using textures to add fine detail to each stroke. Traditional polygonal modeling is perfect for items with rigid shapes and surfaces, while 3D sculpting shines in creating objects with organic and smooth shapes.

- Vertex paint: vertex painting works directly on the mesh. Each individual vertex can be assigned a vertex color, but the colors applied using vertex painting are visible even if there is a UV texture or a material applied. It is a straightforward way of painting color onto an object, by directly manipulating the color of vertices, rather than textures, and is straightforward. When a vertex is painted, the color of the vertex is modified according to the settings of the brush. The color of all visible planes and edges attached to the vertex are then modified with a gradient to the color of the other connected vertices.
- Weight paint: it works exactly as vertex paint, but the color is used to assign weight to the vertices on a vertex group. It is primarily used for rigging meshes, where the vertex groups are used to define the relative bone influences on the mesh. But we use it also for controlling particle emission, hair density, many modifiers, shape keys, etc. For example, if a bone is moved, it will move the vertices of a mesh, accordingly to the amount of influence. Weights are visualized by a gradient using a cold/hot color system, such that areas of low value (with weights close to 0.0) are displayed as blue (cold) and areas of high value (with weights close to 1.0) are displayed as red (hot). And all in-between values are displayed as rainbow colors (blue, green, yellow, orange, red).
- Texture paint: this last mode allows to paint a texture applied on a model, either using the brush to paint on the image or directly on the mesh. The process of UV unwrapping is necessary before trying to paint on the model.

2.1.1 Modifiers

Modifiers are procedural, non-destructive methods for manipulating and generating a mesh's geometry.

Procedural means that modifiers can be combined into a stack, called Modifiers Stack, that stores them in a specific order, creating a chain of operations executed in sequence. Switching position of modifiers change the final result in most of the cases.

Non-destructive means that the object's geometry does not really change until the modifier is applied permanently. They just affect how an object is displayed and rendered and it is still possible to edit the mesh, for example a low poly mesh can maintain its obvious advantages and be displayed as a high-resolution mesh using a subdivide and smooth modifier.

Modifiers are controlled with just a few actions and can save a lot of time.

Using modifiers, it is important to have a good topology and normalize the scale of an object to work in the best way possible.

Some modifiers generate new geometry, replacing or adding to your mesh, while others deform existing geometry according to certain rules or offer ways to plug simulations and other more complex entities into your scene.

- Modify: they usually do not directly affect the geometry of the object, but some other data, such as vertex groups.
- Generate: constructive/destructive tools that will affect the whole topology of the mesh.
- Deform: these only change the shape of an object, without altering its topology.
- Physics: those represent physics simulations. Their only role is to define the position in the modifier stack from which is taken the base data for the simulation they represent.

2.1.1.1 Generate

Generate modifiers are constructive/destructive tools that will affect the whole topology of the mesh.

The Array modifier creates an array of copies of an object in a desired arrangement. Vertices in adjacent copies can be merged within an editable tolerance to have a continuous geometry and a smooth subsurface. This modifier can be useful for creating complex repetitive shapes. Multiple array modifiers may be active for an object at the same time.

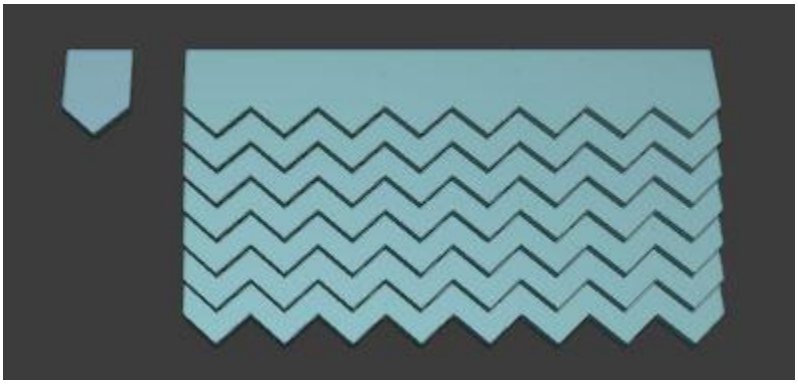


Figure 2.6 A roof is composed by a set of tiles, which makes it a perfect example for the usage of the Array modifier.

The Boolean modifier performs operations on meshes that are otherwise too complex to achieve with as few steps by editing meshes manually. It uses one of the three available Boolean operations to create a single mesh out of two mesh objects: union, intersection, or difference. The second mesh object or the collection of objects that is used as second operand is called target.

The result depends on the geometry of the two objects, specifically faces and vertices. For example, in case of a difference the base object will be cut wherever it intersects the target object, this could create strange geometry adding vertices in an unwanted position.

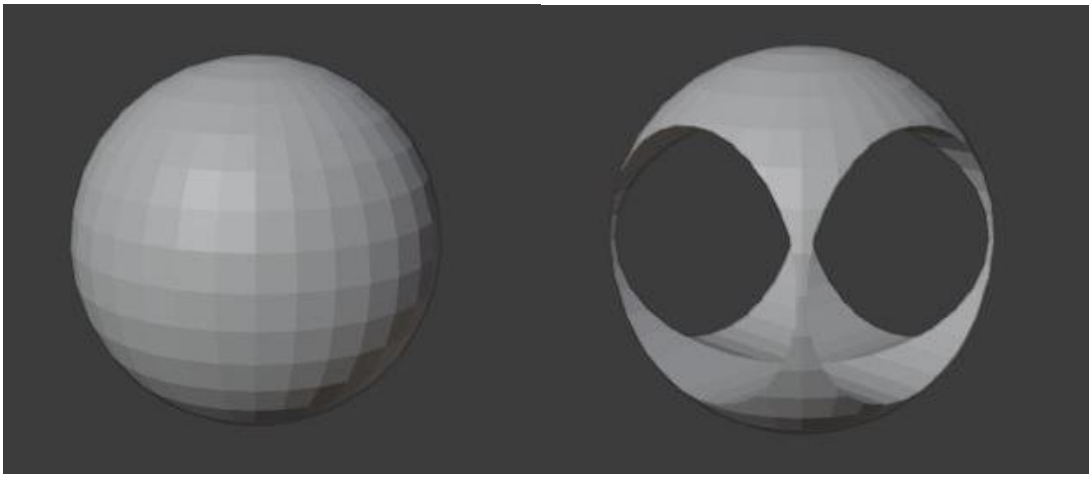


Figure 2.7 Boolean modifier is extremely useful to create holes in a mesh

The Build modifier causes the faces of the mesh object to appear or disappear one after the other over time. Faces appear in a specific sequence, accordingly with the index to which they are assigned (by default, the order of creation).

The face/vertex order can be altered in Edit Mode by using Sort Mesh Elements.

Geometry nodes is a particular modifier because it modifies an object's geometry in way more complex than the other modifiers and it works with nodes.

Nodes are divided in categories:

1. Attribute: Nodes for working with data stored per object element, e.g. size, location, rotation).
2. Color: Nodes for modifying color data passed through color sockets.
3. Geometry: Nodes that can operate on different geometry types (volume, mesh).
4. Input: Nodes used as input to other nodes.
5. Mesh: Nodes that only operate on meshes.
6. Point: Nodes that modify the object elements, e.g. vertices.
7. Utilities: Nodes with general purpose for modifying data.
8. Vector: Nodes for modifying vector quantities.

Each node performs some operations, and they are connected via sockets to form a node tree. Each node is executed from the beginning of the chain in sequence and all operations converge to a socket, which contains all the geometry information, at the end.

The Mirror modifier mirrors a mesh along its local X, Y and/or Z axes, across the Object Origin. It can also use another object as the mirror center, then use that object's local axes instead of its own. It is especially useful to model object with symmetry, allowing you to work on just a part of the model.

The Mask modifier allows vertices of an object to be hidden dynamically based on vertex groups. This modifier needs two inputs: a vertex group and a threshold.

Each vertex inside the vertex group has a specific weight, so when the threshold is greater than the weight of the vertex, the vertex will be hidden. The operation is reversible.

The Solidify modifier extrudes the geometry along its normals to create a solid shell from surface.

The Subdivision Surface modifier is used to split the faces of a mesh into smaller faces, giving it a smooth appearance. It is produced by a recursive algorithmic method and enables to create complex smooth surfaces while modeling simple, low vertex meshes. It avoids the need to save and maintain enormous amounts of data and gives a smooth "organic" look to the object.

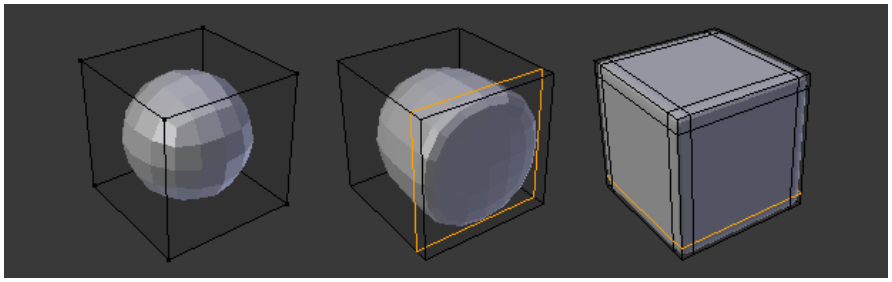


Figure 2.8 Subdivision surface modifier smooths the curvature of a surface increasing the resolution of the mesh and averaging the location of the points in between the original vertices.

2.1.1.2 Deform

The Curve modifier provides a simple but efficient method of deforming a mesh along the path of a curve. The object can be deformed on all global axis, X, Y, or Z.

The Explode modifier uses both the mesh geometry and a particles system to simulate an object that breaks into pieces.

Each face of the mesh will attach to an emitted particle, following its path, so both the number of emitted particles and number of faces determine how granular the effect is. It is also possible to split the mesh in pieces based on location of emitted particles, instead of using existing faces, decide if the faces are visible when their attached particles are unborn, alive, or dead and the scale factor of each face using the size of its attached particle, once that particle is alive.

The Shrinkwrap modifier allows an object to "shrink" to the surface of another object. It moves each vertex of the object being modified to the closest position on the surface of the given mesh (using one of the methods available). It has several usages from retopology, placing object on top of other object (like a house to the ground, or windows to walls), making surfaces automatically respond to changes on other surfaces (for example wheels on a dirt road)

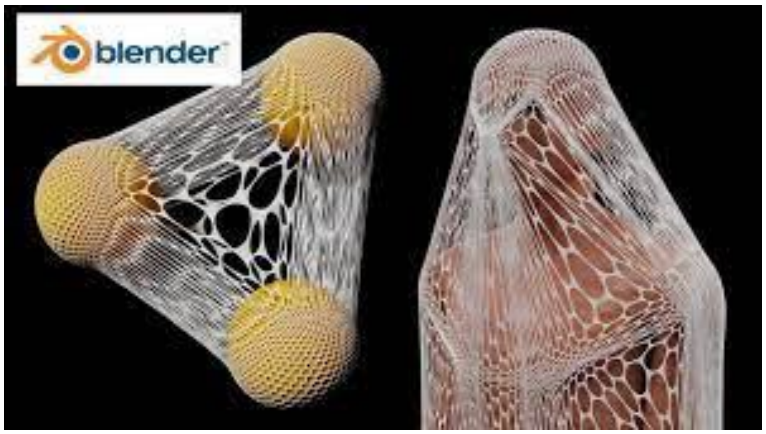


Figure 2.9 Example of Shrinkwrap modifier usage on nets which wrap objects

2.1.1.3 Physics

2.1.1.3.1 *Particles systems*

Particles systems contain a set of individual particles, each of which can be efficiently simulated as a single point mass. Particles can be emitted from a surface or attached to it.

Particles systems can have millions of particles that play a role analogous to individual molecules and are frequently used to model objects with certain kind of behavior or amorphous compressible objects.

The Emitter type is ideal for replicating phenomena - such as sparks, rain, rock falls, clouds, fog, snow, dust, smoke, fire, stars, or abstract visual effects like magic spells - which requires particles that fade out quickly and are then re-emitted from the effect's source.

The other type, named Hair, is ideal for things that contain many strands - such as fur, hair, and grass - which requires scattering objects on the surface.

The particles are set into motion using some algorithm and can be rendered in different ways, also as objects or entire collections of objects.

Controls for creating moving, changing, and deleting particles during their lifetime are part of the system.

Particles can interact with the forces in the world based on five options:

- No: particles will not move.
- Newtonian: particles are subject to gravity, acceleration, and force fields defined on other objects.
- Keyed: it makes the movement of one particle system dependent on the
- movement of another particle system.
- Boids: particles are controlled by a limited artificial intelligence, which can be programmed to follow basic rules and behaviors.
- Fluid: it mimics the properties of fluid dynamic systems.

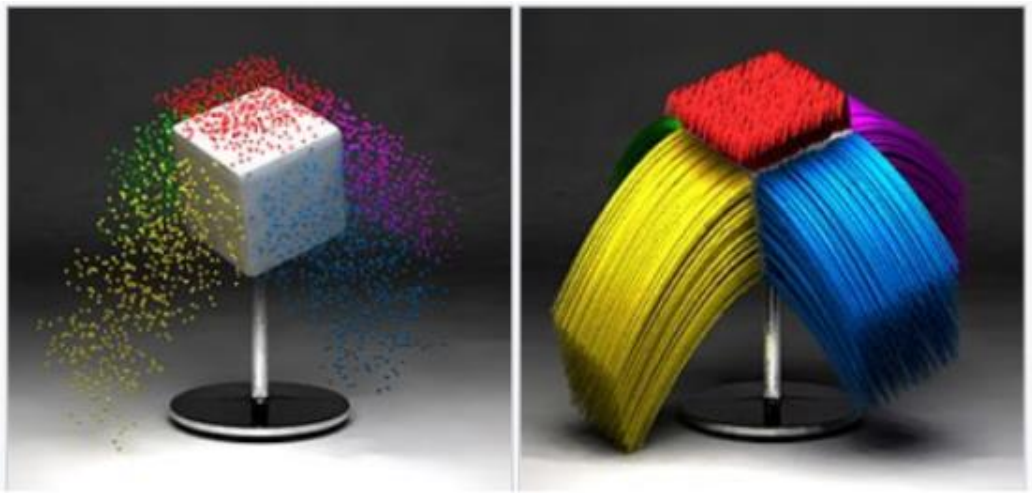


Figure 2.5 The same particle system set as Emitter and Hair.

2.1.1.3.2 *Rigid body*

The rigid body simulation can be used to simulate the motion of solid objects. It affects the position and orientation of objects and does not deform them.

There are two types of rigid bodies: active (dynamically simulated) and passive (static). To create more complex simulation, it is possible to combine the animation of an object based on keyframes and the simulation of the rigid body. There are several factors which affect the simulation such as scale, mass, origin point, and the shape used to perform calculation to detect collisions.

Primitive shapes are best in terms of memory and performance but do not necessarily reflect the actual shape of the object. On the other hand, mesh-based shapes (Convex Hull and Mesh) are calculated based on the geometry of the object, so they require more complex operations.

2.1.1.3.3 *Soft body*

Soft body dynamics refers to the way that flexible materials deform in response to the forces on them. It works by treating a mesh as a network of springs. The edges of the mesh are given springlike behavior. Parameters are used to determine how these "springs" react to forces and how the soft body simulation itself interacts with the mesh. Soft body simulations can provide a stretchy, flexible surface effect like cloth. Cloth simulation is similar to a soft body simulation, but it has complex internal and environmental interactions that makes it a separate subject.

2.1.1.3.4 *Forces & Collision*

Many natural forces, from wind to magnetism, can be made to act on particle systems by using force fields. Forces offers a way to influence simulation. Collision can be detected on an object only in relation to particles, soft bodies, and cloth objects.

It is possible to control some properties to simulate impacts between different object, such as objects that fall in a dense fluid or bounce on an elastic surface.

2.1.1.3.5 *Dynamic paint*

Dynamic paint is a modifier and physics system that can turn objects into brushes or canvases.

Object, used as brushes, will paint on object, used as canvas.

The painted color can have different based on the type of the canvas surface:

1. Paint: the brush will paint on the vertex of the surface, as in the vertex paint mode.
2. Displacement: the brush will paint on the surface and the information is used to displace the vertices of the mesh.
3. Wave: It produces simulated wave motion every time that the brush touches the canvas surface. The resulting effect will be considering the velocity with which the brush touches the canvas to reproduce the strength of the wave motion.
4. Weight: the brush will affect the weight of the vertices, changing its color.

2.2 Shading

Shading refers to the process of creating the final look of an object in the 3D scene. This process could affect either the appearance of an object or the geometry of the surface.

2.2.1 Shaders

Shaders are small programs that describe how subsequent polygons and images are to be processed during rendering.

An important aspect of the rendering process is the lighting setup because light and shadow are the information which allow humans to perceive the world, but shader allows to create a set of operations, named Materials, which define how light rays interacts with the objects.

Materials are what you layer on top of a 3D object, to control the way the object is perceived when rendered. They define how light rays interacts with the surface, or the nuances of color, texture, transparency, and reflectivity, as well as a material, such as wood, marbles, behaves in the real world.

The simplest shading model is based on an observation made by Lambert in the 18th century: the amount of energy from a light source that falls on an area of surface depends on the angle of the surface to the light.

Lambertian shading is view independent: the direction from which you look does not change the color of a surface. It is also called diffuse, does not produce any highlights, and leads to a very matte, chalky appearance. Many real surfaces have some level of shininess, producing highlights, or specular

reflections, that appear to move around as the viewpoint changes (Fresnel effect), so many shading models add a specular component to Lambertian shading.

Materials can simulate object which are soft like leather, or clear and transparent and bouncing light around like glass or crystal. They can define the appearance of the surface, the volume inside the object, and the displacement of the surface.

The surface shader controls the textures and light interaction at the surface of the mesh.

The volume shader defines the interior of the mesh. A material can have just a volume shader for cases like smoke and fire, or it can be combined with a surface shader for materials like cloudy glass. The shape of the surface and the volume inside it may be altered by displacement.

Figure 2.6 Photorealistic materials applied to spheres

2.2.2 Texture mapping

Texture mapping refers to using an image, called a texture map, texture image, or just a texture, to store the details that you want to go on a surface, then mathematically "mapping" the image onto the surface.

The most basic use of texture mapping takes a polygon (or a collection of polygons) and assigns a color to each point via a lookup in a texture image. This is called diffuse mapping.

The idea of using a texture to modify the color characteristics of each point of an image is only one of many applications of texture mapping. The central ideas of texture mapping have been generalized and applied to many surface properties.

The appearance of a surface, for instance, depends in part on the surface's normal vector (or normal), which is the vector that is perpendicular to the surface at each point. This normal vector is used to compute how light reflects from the surface.

If instead of using the true normal to a surface (or its approximation by interpolation) we use a different one at different points of each polygon, the surface will have a different appearance at different points,

appearing to tilt more toward or away from us, and this idea can be used to generate surfaces which seems lumpy, while the underlying shape is smooth. Unfortunately, near the silhouette of the surface the unvarying nature is evident; this is a common limitation of such mapping tricks. On the other hand, being able to draw just a few normal-mapped polygons instead of thousands of individual ones can be enough of an advantage to make this choice appropriate.

Normal mapping takes advantage of this fact by making the shading normal depend on values read from a texture map. The simplest way to do this is just to store the normals in a texture, with three numbers stored at every pixel, instead of as the three components of a color, as the 3D coordinates of the normal vector.

Displacement mapping is the alternative to normal mapping, and it is used to alter the geometry, instead of the normals. A scalar (one-channel) map gives the height of each point and changes the surface, moving each point along the normal of the smooth surface to a new location. The most common way to implement displacement maps is to tessellate the smooth surface with many small triangles, and then displace the vertices of the resulting mesh using the displacement map. Usually, you use either a displacement map or a normal map, not both.

Roughness mapping describes the surface irregularities that cause light diffusion. The light intensity does not change, what changes is the direction of the rays. Rougher surfaces will have larger and dimmer-looking highlights. Smoother surfaces will have sharp specular reflections.

Metallic mapping describes which areas in the base color should be interpreted as reflected color(dielectric) and which areas denote metal reflectance values.

2.2.2.1 UV unwrapping

UV unwrapping describes the process of creating a set of 2D coordinates for the faces of a model, which we can then use to apply textures to the model.

Objects need to be cut on the surface (along edges called seams) and spread it out flat.

Each vertex has its coordinates x, y and z in the 3D space. A UV map associates to each vertex a separate set of 2D coordinates that will be used to map a 2D image to the surface of the mesh.

Figure 2.7 Unwrapping a cube

2.3 Lighting

Light and shadows are used by the visual system as cues to determine depth perception and distance. They are key factors that most impact how we perceive a scene.

Lighting can change the mood of an image and make characters and environment look either scary or friendly, cold, or warm. Blender provides many ways to add light to scene.

2.3.1 Type of lights

The Point light is an omni-directional point of light that emits energy equally in all directions. Being a point light source, the direction of the light hitting an object's surface is determined by the line joining the light and the point on the surface of the object itself.

As in real life, the intensity of a ray decays based on (among other variables) the distance that run from the Point light to the object. In other words, surfaces that are further away from the light will be rendered darker.

The Spotlight is like a spotlight at a theater or a flashlight. It emits a cone-shaped beam of light from their origin, making them a valuable tool for lighting specific areas. They support both ray-traced shadows and shadow buffers. Shadow buffers produce shadows without ray tracing and are significantly faster to render. This makes them very versatile and useful for scenes that need a high degree of control over lighting.

The Area light is a surface that emits light from all over its area. It produces shadows with soft borders by sampling a light along a grid the size of which is defined by the user. This is in direct contrast to point-like artificial lights which produce sharp borders. Area lamps support ray-traced shadows and are well suited for creating soft shadows and studio style lighting, but they can slow render time significantly because they require more samples than other lamp types to produce noise-free results.

The Sun light provides light of constant intensity emitted in a single direction from infinitely far away. A sun light can be very handy for a uniform clear daylight open-space illumination.

The World defines the scene's background, as well as the environment lighting. By default, world use the Background Shader node which lights the scene and sets the background color or texture and adds to the overall lighting a constant amount of light.

2.3.2 Additional methods

Environment texture can also be used to introduce detail into the illumination without having to model complicated light source geometry. HDRI (high-dynamic-range image) are used in 3D programs

to act as a light source to illuminate your scene, creating very natural lighting effects. They also provide much more realistic reflections.

Shadow maps are a technique for using the machinery of texture mapping to get shadows from point light sources. Shadows are simple to include in ray-traced images, but it's not obvious how to get shadows in rasterized renderings, because surfaces are considered one at a time, in isolation.

Material can make objects emit light through the Emission shader. It is useful to build customized lamps, since the light objects does not have a shape to render, or reproduce phenomena such as fluorescence, incandescence.

2.3.3 Camera

Camera objects captures a portion of the scene which will be used in rendering. They are invisible in renders, so they do not have any material or surface to edit. Many cameras can be added to the same scene, switching between them to choose which is the active one.

Perspective camera is the most common type of camera because it simulates the behavior of real cameras. Considering perspective, distant objects will appear smaller than objects in the foreground, and parallel lines will converge to a vanishing point.

The Focal Length controls the angle of view of a lens. Longer focal lengths result in a smaller FOV (more zoom), while short focal lengths allow you to see more of the scene at once (larger FOV, less zoom). The smaller the focal length, the more the perspective distortion.

Figure 2.8 Perspective camera with 35 mm focal length (left). Perspective camera with 210 mm focal length instead of 35 mm (right).

The Lens shift allows for the adjustment of vanishing points. Vanishing points refer to the positions to which parallel lines converge.

Especially when rotating the camera, it happens that the perspective is distorted much more. Perfect vertical lines start to converge in one point. Using the lens shift can avoid this distortion, it is equivalent to rendering an image with a larger FOV and cropping it off-center.

Figure 2.9 Horizontal lens shift of 0.330 (left). Rotation of the camera object instead of a lens shift (right).

Orthographic cameras, on the other hand, completely discard perspective. Parallel rays are shot from the camera. As a result, all objects appear the same in size, no matter what their distance from the lens is. Orthographic projections come in handy for precise measurements when creating architectural and engineering renderings.

2.4 Animating

Animation is derived from the Latin "*anima*" and means the act, process, or result of imparting life, interest, spirit, motion, or activity. Animation without computers, which is often referred to as "*traditional*" animation, has a long and rich history of its own and along with it some time-tested rules which have been crystallized, giving general high-level guidance to how animation should be done. These principles apply equally to computer animation.

When you see a sequence of related images in rapid succession, they blend and create the perception that objects in the images are moving. The individual images are called frames and the entire sequence is called an animation.

Animation can be achieved through:

1. Keyframing: it gives the most direct control to the animator who provides necessary data at some moments in time and the computer fills in the rest.
2. Procedural animation: it involves specially designed, often empirical, mathematical functions and procedures whose output resembles some particular motion.
3. Physics-based techniques: they solve differential equation of motion.
4. Motion capture: it uses special equipment or techniques to record real-world motion and then transfers this motion into that of computer models

Producing the animations manually is expensive and slow because of the time and skill that the artists require in the process. In a key pose animation scheme (a.k.a. key frame, interpolation-based animation), an animation specifies the poses to hit at specific times, and an algorithm computes the intermediate poses, usually in the absence of full physics.

Using dynamics for creating animation is computationally challenging but it makes the effort of the artists less. Algorithms and computer hardware has become both more efficient and less expensive. In a dynamics (a.k.a. physically based animation, simulation) scheme, objects are represented by position and velocity and a procedure must be computing the new object position at each frame.

Modern animators combine poses with the efficiency and realism of physically based simulation. A classic method is inverse kinematics (IK). An IK solver is given an initial pose, a set of constraints, and a goal. It then solves for the intermediate poses that best satisfy these, and the final pose if it was not determined.

2.4.1 Keyframing

The term *keyframing* can be misleading when applied to 3D computer animation since it does not involve frames as images but as moment in time. It means storing a specified set of values that will describe the 3D scene. Animating a scene implicates that some subset of these values must change with time.

One can, of course, explicitly set these values at every frame, but this will not be particularly efficient. Instead, frames can be chosen along the timeline of animation to create keyframes for parameters and their values.

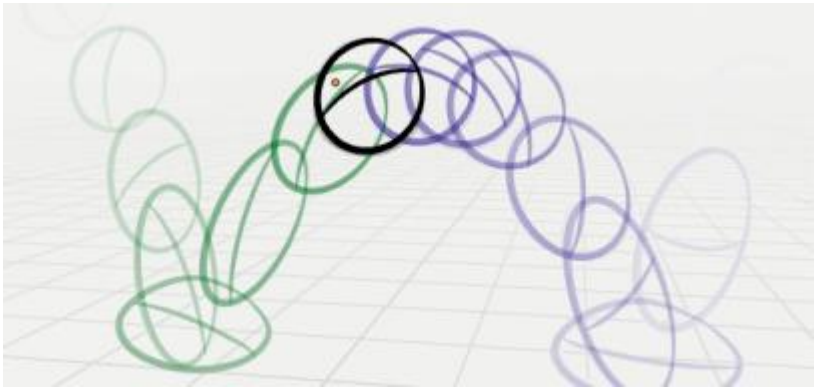


Figure 2.10 The ball animation is recreated using keyframes with the object in a specific configuration

Once the animator sets the key, the animation system interpolates values for in-between frames.

The basic types of interpolation are:

1. Constant: There is no interpolation at all. The curve holds the value of its last keyframe, giving a discrete (stairway) "curve".
2. Linear: This simple interpolation creates a straight segment, giving a non-continuous line.
3. Bézier: The more powerful and useful interpolation, and the default one. It gives nicely smoothed curves, i.e. smooth animations.

Although we are interested only in a discrete set of values, keyframe interpolation is represented and controlled by animation curves, also known as F-Curves. These curves can change the type of

interpolation between each pair of keyframes. Each keyframe has a handle that determine the slope of the curve, creating steeper or smoother changes.

2.4.2 Rigging

Rigging is a general term used for adding controls to objects, typically for the purpose of animation. A rig consists of a hierarchy of transform nodes that is attached to a character.

It often involves using one or more of the following features:

1. Armatures: they allow mesh objects to have flexible joints and is often used for skeletal animation.
2. Constraints: to control the kinds of motions that make sense and add functionality to the rig.
3. Object Modifiers: mesh deformation can be quite involved, there are multiple modifiers that help control this.
4. Shape Keys: to support different target shapes (such as facial expressions) to be controlled.
5. Drivers: your rig can control many different values at once, as well as making some properties automatically update based on customizable mathematical expression and a set of variables.

2.4.2.1 Armature

An armature in Blender can be thought of as the structure of a real skeleton, and just like it an armature can consist of many bones. These bones can be moved around, and the bones attached directly and indirectly will move and deform in an analogous way. Once an armature is ready and pose in a resting position, it must be applied to the mesh that you want to animate. Just like marionettes, bones work as the strings attached to the puppet, so each bone moves certain parts of the mesh, which depends on the vertex group associated with the bone and the weight associated with the vertices in the same vertex group.

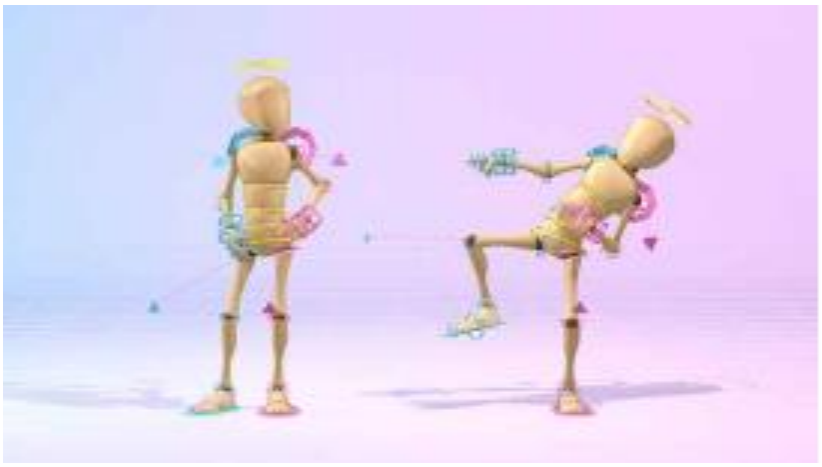


Figure 2.11 A rig applied on a character model

2.4.2.2 Constraints

Constraints are used to limit and control an object behavior using numerical values or another object, called "target".

A similar concept is parenting, an object can have children and any translation, rotation, or scale of the parent will also affects its children.

In both cases, you can control an object's animation through the targets used by its constraints (this is a form of indirect animation).

Copy constraints force its owner to match the property of a target, such as location, rotation and/or scale.

Limit constraints set lower and upper bounds defining where the object can move in relation to the space or a target object, how much it can rotate or be scaled.

Transformation constraints control location, rotation, or scale of an object mapping one of the same set of properties from a target object. For example, the wheels rotate in relation to how much the vehicle to which they are coupled moves, transforming a position value with a rotation value.

The Track To constraint is used to track the movement of another object, making its owner always point to the target, specifically at its origin point.

The Follow Path constraint forces an object to follow a specific path. The path must be an object of type Curve.

2.4.2.3 Shape keys

The purpose of shape keys is creating different versions of the same mesh, and this is useful in many cases. First, shape keys are the most efficient way to morph a mesh using target shapes, for example growing trees or flowers. Furthermore, they can store the poses generated by an armature, allowing to create libraries of poses for common movements such as facial expressions and gestures.

Shape keys can also be used as a corrective tool to solve rigging problems like strange deformations which would occur mostly in joints.

2.4.2.4 Drivers

Drivers are a way to control values of properties by means of a function, or a mathematical expression. A driver is associated with a property of an object that can be animated and it uses a function in which variables can be properties of other objects or global variables. They can be used, for example, to set constraints, make objects move with a certain speed or animate a corrective shape key associated with a movement that will cause strange deformations.

2.5 Rendering

Rendering is a process that takes as its input a set of objects and produces as its output an array of pixels. It takes what is viewed from the camera and transforms it in an image.

Each object in the camera view contributes to each pixel but how it contributes depends on the rendering approach. It is important to know that during the rendering process, the mesh of an object is divided in triangles.

2.5.1 General approaches

There are two generalized approaches:

1. In **object-order rendering**, each object is considered in turn, and for each object all the pixels that it influences are found and updated.
2. In **image order rendering**, each pixel is considered in turn, and for each pixel all the objects that influence it are found and the pixel value is computed.

Ray casting creates one ray per pixel and casts it at every surface. Ray casting lets us process each pixel to completion independently. This suggests parallel processing of pixels to increase performance. It also encourages us to keep the entire scene in memory, since we do not know which triangles we will need at each pixel.

Rasterization allows us to process each triangle to completion independently, typically implemented by marching along the rows of the image, which are called *rasters*.

It solves the problem of holding the entire scene in memory and allows to render much larger scene, because it processes one triangle at a time. It suggests triangles as the level of parallelism. The computation about triangles can be stored in registers or cache to avoid memory traffic, and only one triangle needs to be in memory at a time.

Furthermore, each pixel is touched multiple times and it needs a data structure that helps resolve visibility between visit, comparing the distance to the current point and to the previously closest point. In this way, each pixel is affected only by nearest point and not by one which is hidden. This array is called a **depth buffer** or a **Z-buffer**. Because computing the distance to a point is potentially expensive, depth buffers are often implemented to encode some other value that has the same comparison properties as distance along a ray.

Ray tracing is an image-order algorithm, works by computing one pixel at a time, and for each pixel the basic task is to find the object that is seen at that pixel's position in the image. Each pixel sends a ray in a different direction, and any object that is seen by that pixel must intersect it. The object closest to camera, intersecting the same ray as other objects, will block them from being viewed by the camera. Once that object is found, a shading computation uses the intersection point, surface normal, and other information (depending on the desired type of rendering) to determine the color of the pixel.

A basic ray tracer therefore has three parts:

1. *ray generation*, which computes the origin and direction of each pixel's viewing ray based on the camera geometry.

2. *ray intersection*, which finds the closest object intersecting the viewing ray.

3. *shading*, which computes the pixel color based on the results of ray intersection.

2.5.2 Render engines in Blender

Cycles works by casting rays of light from each pixel of the camera into the scene. They reflect, refract, or get absorbed by objects until they either hit a light source or reach their bounce limit. Following just one ray of light from one pixel is not that accurate since many objects or texture details can be smaller than one pixel, so Cycles fires additional randomized rays (samples) from that pixel and averages the result over time. This brute force approach of averaging randomized samples is called a Monte-Carlo simulation, and when applied to paths of light it is called path tracing.

Eevee uses rasterization via OpenGL 3.3, which is why it can be blazing fast in comparison to other types of rendering. Rasterization estimates the way light interacts with objects and materials using numerous algorithms. The key thing to remember is that this speed comes at the cost of accuracy because it is working with pixel information instead of paths of light.

There are some differences in term of:

1. **Render time**: One of Eevee's most outstanding characteristics is its super-fast render time. It can be up to 12 times faster than Cycles using the same scene and hardware. That means you will not need to wait as long for your scene renders to finish.
2. **Quality**: Due to a key difference between the two renderers, Eevee manages to render your scenes with decent quality. That said, it is not always better than Cycles, especially for photo-realistic renders.

Figure 2.12 A scene of Spring (2019) rendered with cycles and eevee

PROJECT PRESENTATION

The "Virtual reconstruction of Pavia in the 16th century" project has the goal of producing a 3D reconstruction of Pavia during the Renaissance and the changes that have affected the city in this historical period.

The project aims to:

1. Create a virtual interactive environment of Pavia in the 16th century that represents palaces, churches, and monuments as accurate as possible.
2. Reproduce different views of the city and produce videos in which the camera can move freely to explore the city as a tourist.
3. Develop an app for smartphones and tablets to access information relating to the framed area, with the possibility of comparing existing buildings with those digitally reconstructed in 3D on site.

As part of this larger project, the work that will be presented will focus on:

1. The creation of a video of the virtual tour n ° 8. The itinerary shows the buildings built in the second half of the 1500s (Collegio Borromeo, S. M. di Canepanova church and Collegio Ghislieri).
2. Finding a method to represent the temporal changes of the city during the 16th century.

Equipment

Hardware		*Software*	
PC	Asus P8Z77V-LX	3D	Blender
CPU	Intel i7 3770	Image editing	Gimp
RAM	32 GB	Video editing	DaVinci Resolve
GPU	Nvidia GTX TitanZ 12288Mb GDDR5		

3.1 Preparing asset

3.1.1 Buildings

3.1.1.1 Collegio Ghislieri

The Collegio Ghislieri was founded in 1567 by Pope Pius V Ghislieri for young students from his native place with the intention of promoting a cultural and moral renewal of society through the formation of a professional and religiously trained ruling class. The construction of the building, intended to house it, was undertaken in 1571 under the direction of Pellegrino Tibaldi, one of the greatest architects of the

time, who followed the works until 1585, the year in which he was called to Spain by Philip II. In 1588 the Milanese architect Martino Bassi took over, whose project for the church (later built between 1601-1605 by the engineer Alessandro Mollo) with a central development with a monumental entrance from the square, was not implemented. The interior is now defined in the Baroque reform that should have been extended to the facade as well.

The austere spirit of the Counter-Reformation pervades the whole building, starting from the severe facade which, closed at the edges by ashlar and pilasters and articulated in the sequence of windows cut directly into the masonry and resting on string courses, presents the sumptuous portal of Roman school and the lantern tower, typical elements of Tibaldi's formal repertoire.

Figure 3.1 Facade of the college on piazza Ghislieri (left). Central courtyard(right).

Figure 3.2 Rendering of Piazza Collegio Ghislieri.

Figure 3.3 Rendering of the central courtyard of the Collegio Ghislieri.

3.1.1.2 Collegio Borromeo

Pope Pius IV's bull of foundation "Ad apostolicae dignitatis apicem" (15 October 1561) established the start of the San Carlo project to build a university college for young students of the diocese of Milan on the area of a house already owned by the Borromeo family in Pavia: the project was entrusted to the architect Pellegrino Pellegrini known as Tibaldi (1526-1596).

The foundation of the College is part of both the more complex framework of the educational and pedagogical reform promoted by Carlo (archbishop of Milan since 1560).

The first stone was in fact laid on June 19, 1564, and the construction of the large square-shaped factory proceeded for over twenty years (finished in 1588) with a formal Mannerist repertoire, expressed in the facade with a strong plastic prominence and the monumental portal.

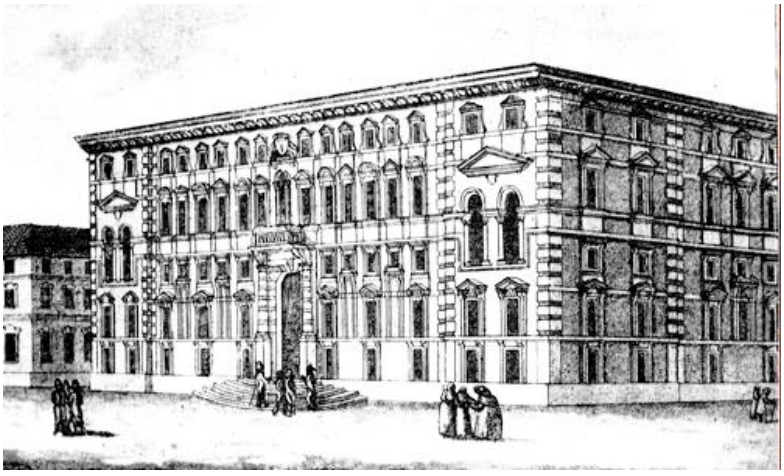


Figure 3.4 Drawing of the façade by Federico Zuccari, 1604

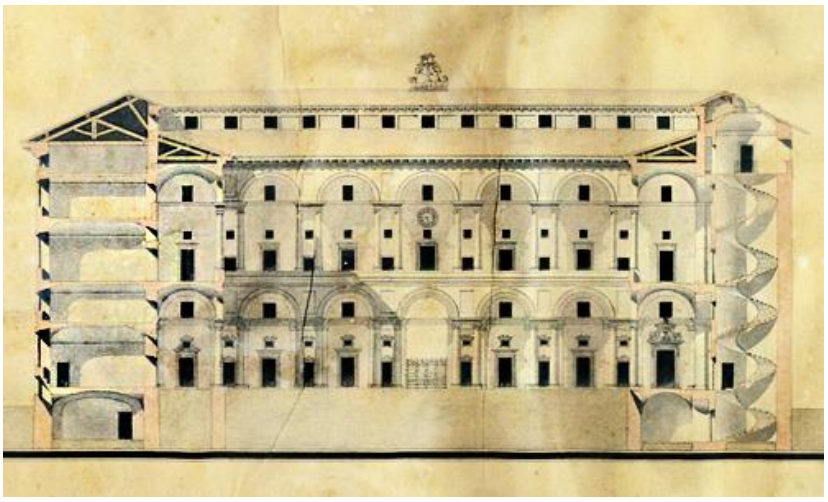


Figure 3.5 Project of the structure depicted by Giorgio Vasari, 1586.

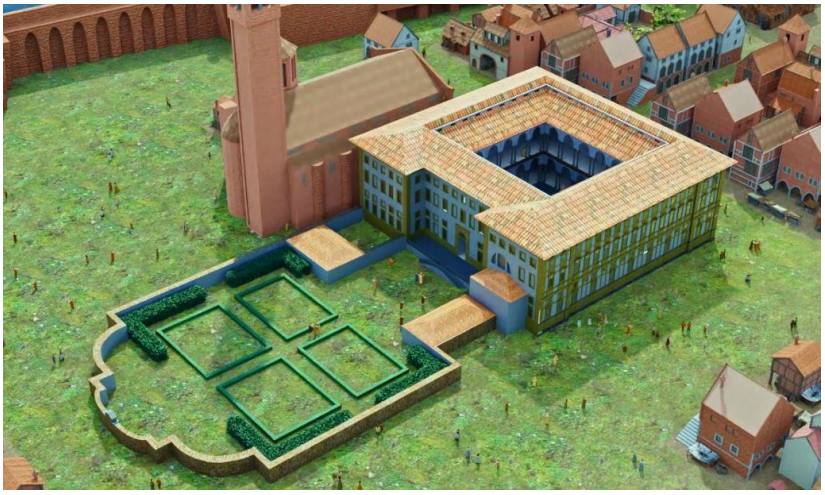
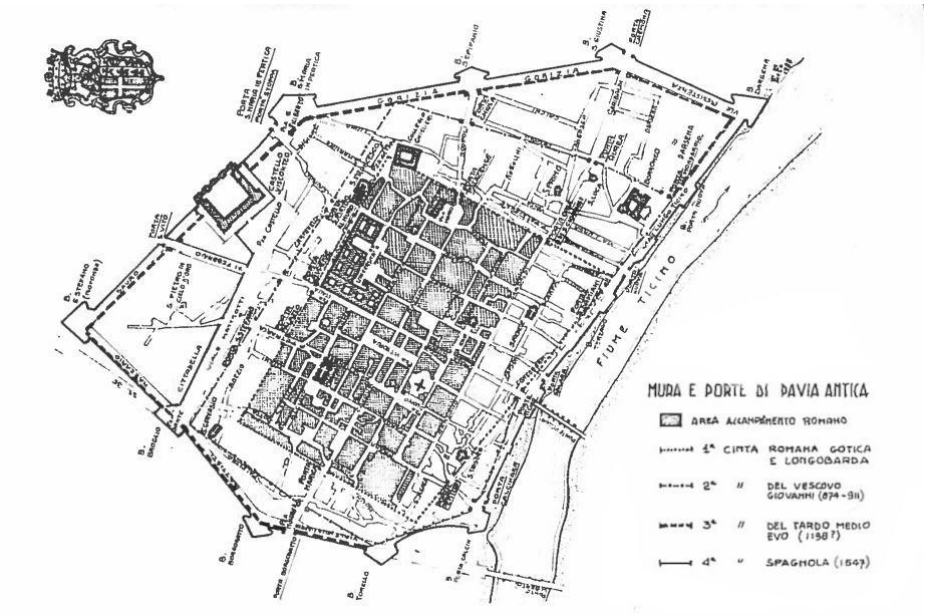


Figure 3.6 Rendering from above of the Collegio Borromeo

Figure 3.7 Rendering of the central courtyard of the Collegio Borromeo.

3.1.1.3 Walls

The city of Pavia was defended by a formidable circle of Roman walls, of which no traces remain. At the end of the 12th century a new medieval wall fortification was built and only some sections between Porta Stoppa and the Castle and Porta Nuova on the Lungo Ticino Sforza still survive.



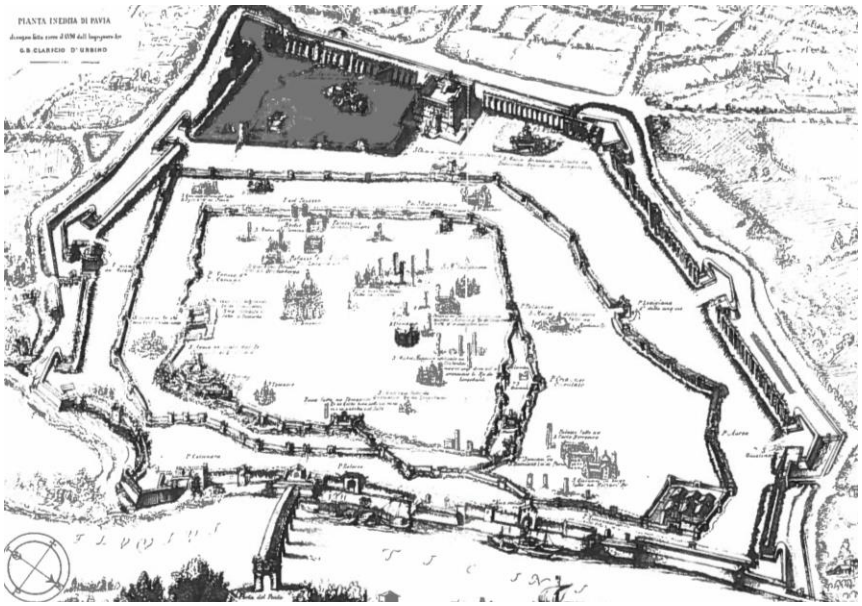


Figure 3.8 Maps of Pavia which reconstructs the perimeters of the successive walls over time.

In the sixteenth century the city, under the Spanish domination, was surrounded by a new bastion of walls, however, using the new pentagonal bastions in place of the old towers. It followed the pre-existing layout of the medieval walls, joining for a good stretch to the south along the Ticino with the relative gates (Porta Nuova and Porta Calcinara).

The first bastion fortifications were erected by the French in 1506, but the new defensive system was made by the Spaniards who, from 1557 to 1560, erected a series of mighty curtains around the city reinforced at the corners by twelve bastions.

The Spanish age wall built around the middle of the 16th century was partially destroyed in the 19th century for reasons of visibility and demolished after the First World War.

Of the ancient gates remains Porta Milano, from the early nineteenth century with the two marble towers surmounted by the statues of the Po and the Ticino.

Figure 3.9 3D representation of the medieval walls

Figure 3.10 3D representation of the Spanish walls.

3.1.1.4 Church of S. Maria di Canepanova

It is the most important 16th century monument in the city of Pavia. A religious congregation promoted in 1492 the construction of the sanctuary in honor of the Virgin due to the miracles performed by the fresco already painted outside a Canepanova house.

The construction of the church began in 1500 and the first part was completed in 1507. The project of the church, with a central plan and dome, with a square choir, was then enlarged between the end of the 16th century and the beginning of the 17th century on the area of a fifteenth-century building (the so-called Canepanova cloister) still partially subsisting was probably defined by Bramante, as some scholars claim, while the works were carried out under the direction of Giovanni Antonio Amadeo.

Figure 3.11 Map of the church from the drawing of Ottavio Ballada, 1617

Figure 3.12 Facade of the church.

Figure 3.13 Rendering from above of the church of S. M. di Canepanova.

3.1.1.5 Monastery of San Felice

It was one of the main female Benedictine monasteries of Pavia. Originally founded in the Lombard period, it was completely rebuilt in 1500, by Giovanni Antonio Amadeo or one of his followers, by order of Abbess Andriola de Barrachis (whose name is inscribed, with the date, 1500 in a capital of the cloister). Painter and sensitive interpreter of Renaissance values, she appears to be the promoter of a renewal integrated with the pictorial decoration that in the cloister also fakes the architectural illusion of candelabra columns painted along the perimeter walls.

Figure 3.14 Map of the monastery from the drawing of Ottavio Ballada, 1617.

Figure 3.15 Central courtyard of the monastery.

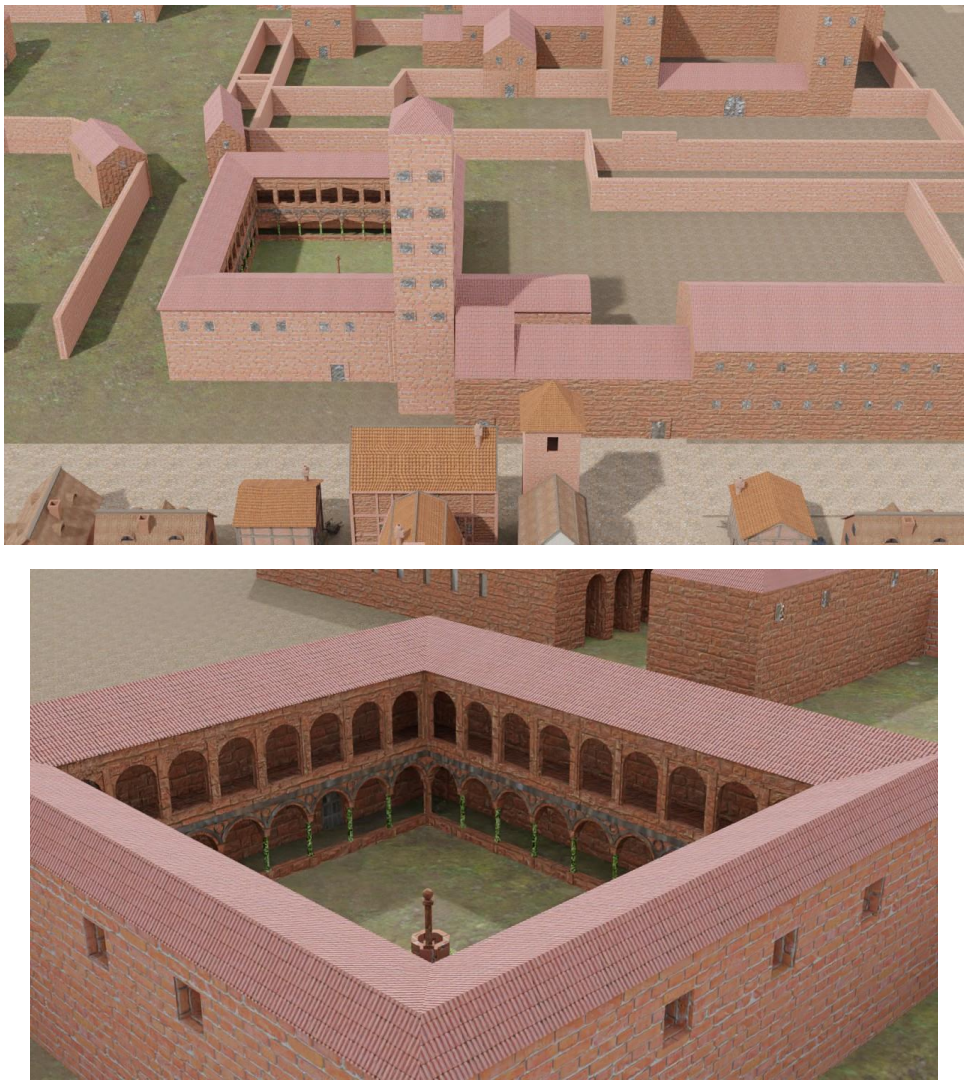


Figure 3.16 3D representation of the monastery of S. Felice.

3.1.1.6 Palazzo Broletto

Palazzo Broletto in Pavia is a twelfth century building located in Piazza della Vittoria. It was built at the behest of the bishop San Damiano who elected it as a bishopric.

During the sixteenth century it was modified with the addition of two loggias:

- The loggia on Piazza Grande: except for the head to the west, erected with the staircase shortly after the mid-1500s, the loggia with a double order of arches, with profiles and oculi in terracotta, renews in widely recurring forms in the cloisters and courtyards of the early 1500s in Pavia the Gothic facade (partly preserved in the back wall) of the thirteenth century Broletto.
- The loggia of the notaries: it was erected as a vestibule by the college of notaries which held its seat on the ground floor of the Broletto. Dated to 1539 by two epigraphs still walled up on the front, it denotes the final evolution of the Renaissance language in Pavia.

Figure 3.17 Loggia on Piazza Grande.

Figure 3.18 Loggia of the notaries.

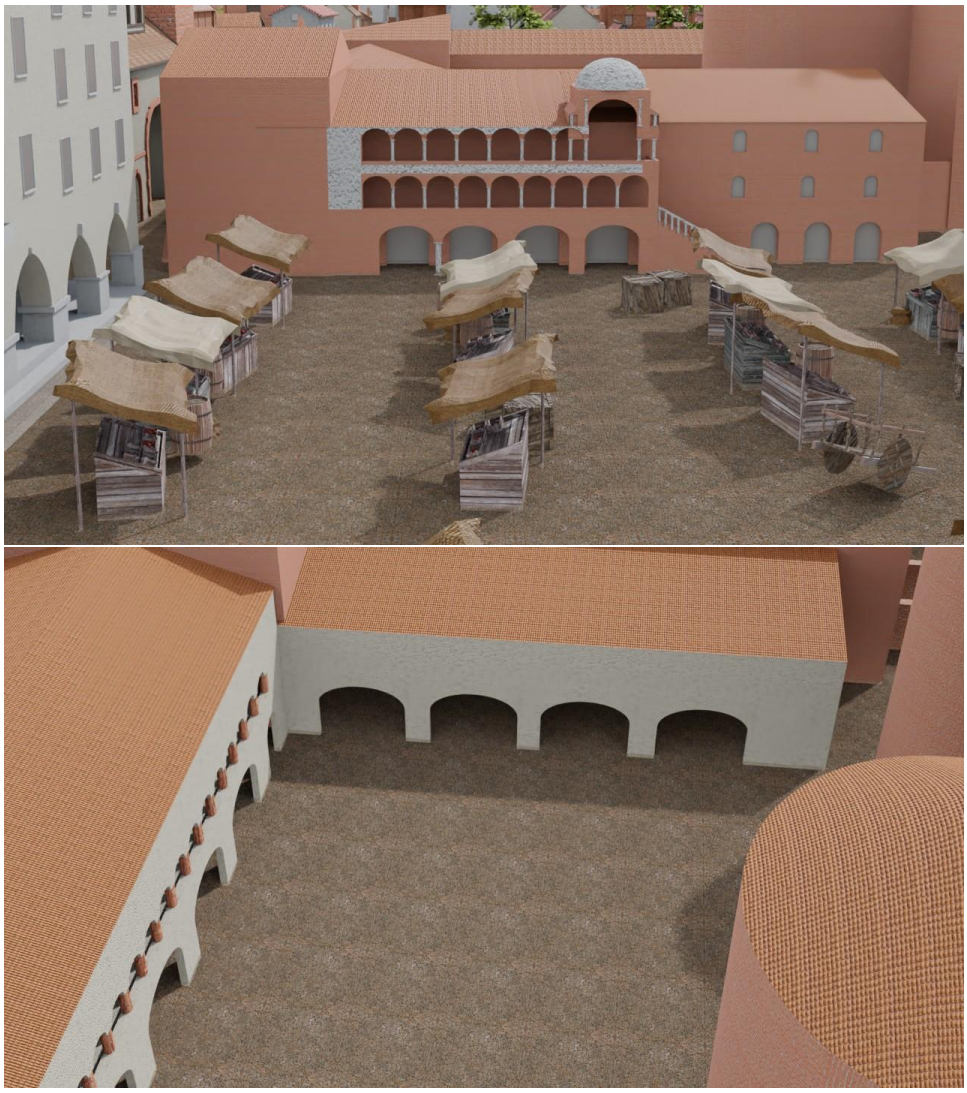


Figure 3.19 3D representation of Palazzo Broletto.

3.1.1.7 Visconti castle

In 1359 Galeazzo II Visconti conquered Pavia after a long siege, as soon as he took the city the lord decided, perhaps due to disagreements with his brother Bernabò, to move his court from Milan to Pavia.

He had the houses and churches located in the area chosen for the castle, west of S. Pietro in Ciel d'Oro, demolished, and the works proceeded quickly and in just 5 years the building was completed. Galeazzo II ordered his subjects to send workers and craftsmen, while the Piacentini were forced to dig the moat. The Castle of Pavia was above all the splendid seat of a refined court: a square layout (150 meters per side) with square corner towers and subdivision of the buildings into square spans, a typical design of the Visconti castles, characterized by large, mullioned windows, the air porch of the courtyard and the frescoes of the internal rooms, elements that reflect the taste of the international Gothic.

The castle was the scene of the famous battle of Pavia in 1525 where it lost the northern part, which fell under the blows of French artillery during the siege of 1527 and in its place was built a section of the bastion wall made by the Spaniards in the middle of the century, which allowed the soldiers to have a wide view of the area outside the walls.

Between the sixteenth and seventeenth centuries, a foundry was created inside the castle to make cannons, one of these pieces of fire is kept in Lisbon and dates to 1572 and could be transported on the walls built by the Spaniards through a slide in case of enemy attacks.

The castle continued to be the residence of the castellans and the Spanish general Antonio de Leyva placed his residence in the eastern part of the building.

Figure 3.20 The Visconti castle from above.

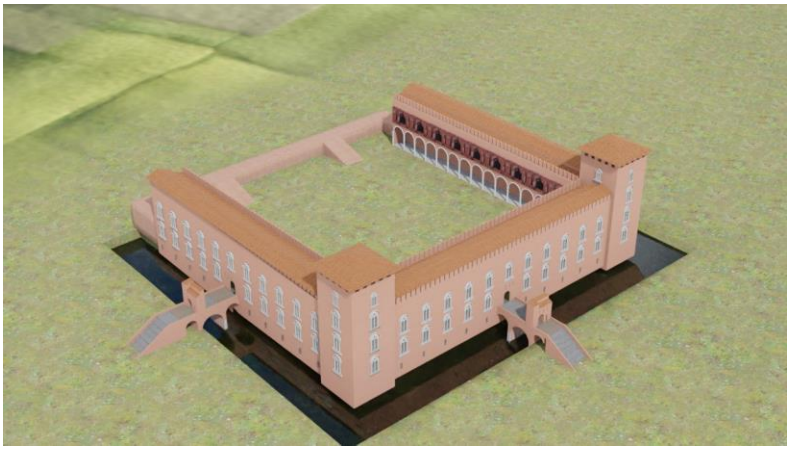


Figure 3.21 3D representation of it (right).

3.1.2 People

3.1.2.1 Villagers

Figure 3.22 Generic female characters

Figure 3.23 Generic male characters

Figure 3.24 A friar and a student

3.1.2.2 Soldiers

Figure 3.25 Soldiers include knights, arquebusiers and infantrymen

3.1.3 A closer look to the asset

Being the combination of the work of many students and some of them worked on different software, it was necessary to examine the status of the various elements and possibly make corrections.

3.1.3.1 Common issues

3.1.3.1.1 *UV Mapping*

Figure 3.26 Typical UV mapping errors include texture stretching, texture scale and rotation.

Uv map are created in blender using various algorithms including:

- UV- unwrap: This tool unwraps the faces of the object to provide the "best fit" scenario based on how the faces are connected and will fit within the image and considers any seams within the selected faces. If possible, each selected face gets its own different area of the image and is not overlapping any other faces UVs. If all faces of an object are selected, then each face is mapped to a part of the image.

- Smart UV Project: it cuts the mesh based on an angle threshold (angular changes in your mesh), automatically creating seams. This algorithm examines the shape of your object, the faces selected and their relation to one another, and creates a UV map based on this information.
- Project from View: it takes the current view in the 3D Viewport and flattens the mesh as it appears.

Most of the time, smart UV project is a fast way to obtain a good result with minimum effort. Once the Uv map is generated, it can be modified manually moving the vertices of the mesh on the image flat plane.

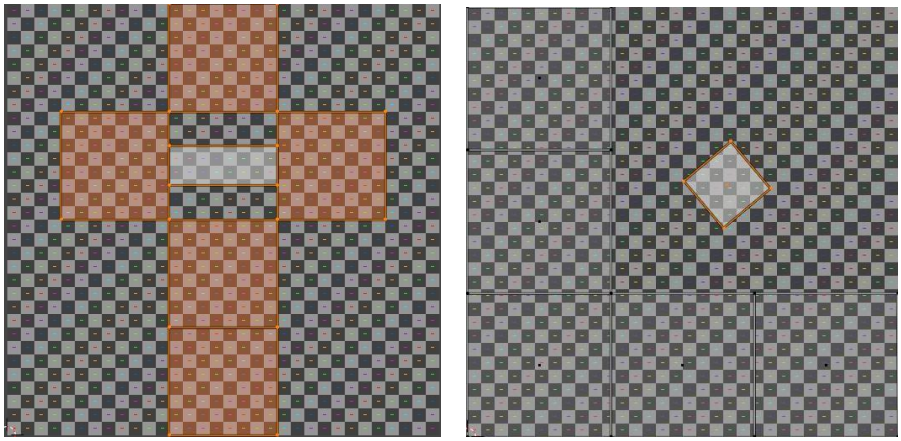


Figure 3.27 Editing a Uv map of a cube in the Uv editor

3.1.3.1.2 *Bad topology*

The chance of finding problems in a mesh is usually very high. One of the most subtle problems to identify, but extremely easy to solve, is doubles. Doubles are overlapping vertices or so close that are perceived as one and result in a waste of memory and in an increasing mesh complexity that could compromise the application of smooth shading or subdivision surface modifiers. Blender provides an algorithm that merges all the vertices that are close to each other within a certain distance. Sometimes, if the model has been already unwrapped and shaded, this method, being automatic, can change the final look because some important vertices are displaced, so you need to be careful using it.

3.1.3.2 Updating asset

3.1.3.1.1 *Collegio Ghislieri*

The Collegio Ghislieri was surrounded by a courtyard on the south side and one on the west side. In the west one we find the church of San Pio and a small house.

A quick create to create a simple low-poly house is to:

- Block the big shapes – start with a cube, change the size to create a rectangular shape and adding a triangular shape for the roof.
- Add a new layer of detail – windows and doors can be obtained making holes in the mesh. Once you have the shape of a window, the Boolean modifier will help to modify the original mesh faster subtracting the target shape

Figure 3.28 Map of the Collegio Ghislieri from the drawing of Ottavio Ballada, 1617

Figure 3.29 Rendering of the Collegio Ghislieri

Figure 3.30 Rendering of the Collegio Ghislieri.

3.1.3.1.2 *Medieval Walls*

The medieval walls are based on four architectural elements: the wall, the arches, the doors and the towers. They are inside the walls made by the Spaniards and follows the same path surrounding the city.

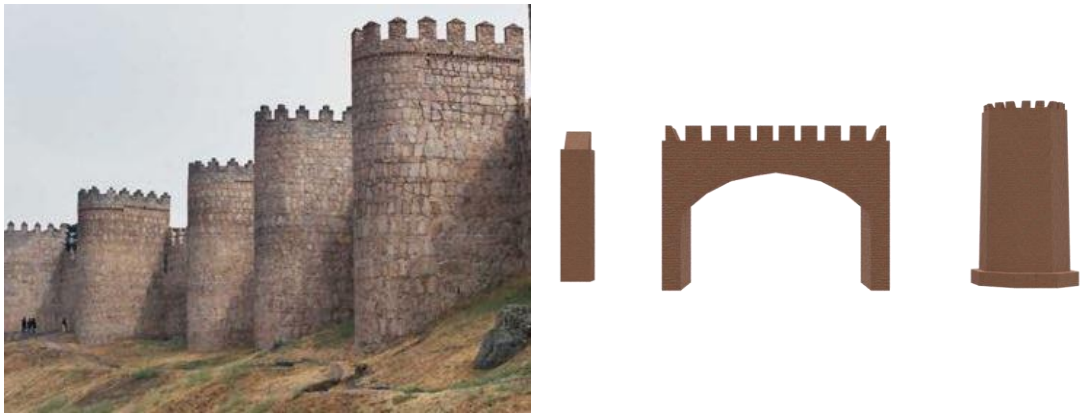


Figure 3.31 Breaking down a complex model in simpler elements

First, you create a tower, a door, a small section of the wall and an arch. To further speed up the process, you can exploit the natural symmetry of these object and work only on a section of the model, applying a mirror modifier.

Thus, it is possible to combine all these elements together to achieve the desired wall structure.

- Create a `path` with a curve object
- Take the small section of the wall and apply an `array modifier` to create a sequence of copies to simulate the walls. Apply a curve-deform modifier so the walls now will be forced to follow the path created previously.
- Create a new plane, enter in edit mode, and merge all the vertices into one. Apply an array modifier to create as many copies as you want and apply a curve- deform modifier with the

same path. Create a new geometry node and add a Point instance node. It will replace each vertex with a specific object, in this case the tower.

- Take the last two elements, duplicate-link them (this will create a new object with all its data linked to the original object) and put them on the desired position.

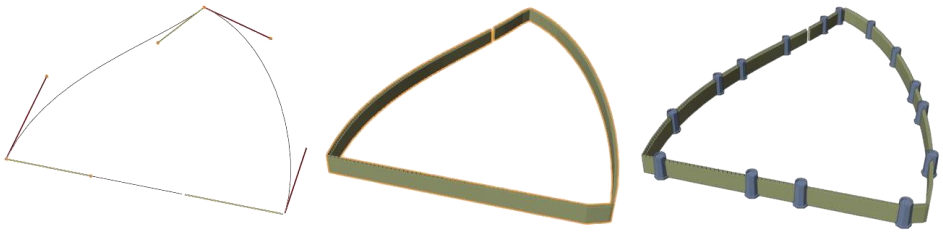


Figure 3.32 Procedural generation of the walls

Once the design process is complete, you can apply all the modifier and work on the details.

3.1.3.1.3 *Neighborhood Pavia Nord-east*

This set of houses was full of problems (non-manifold mesh, missing windows, and doors), so it was deleted and did it from scratch.

Using as reference the old neighborhood, it was easy to create a similar design for the low walls and rearrange the new houses, obtained copying the models spread all over the city, to fill the gaps.

Figure 3.33 Old version of the neighborhood

Figure 3.34 New version of the neighborhood

3.2 Setting up the scene

3.2.1 Match between the model and the maps

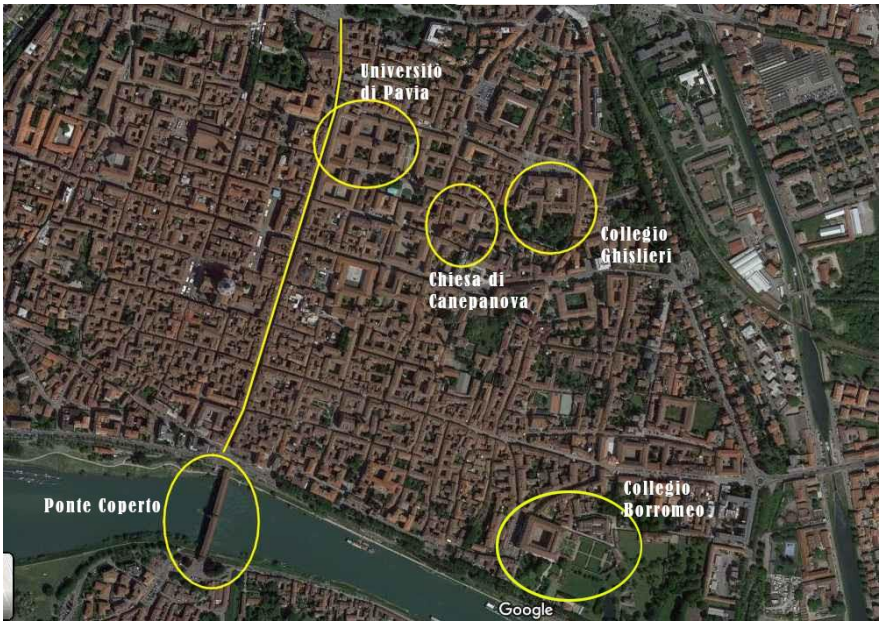


Figure 3.35 Map of Pavia, taken from Google maps.

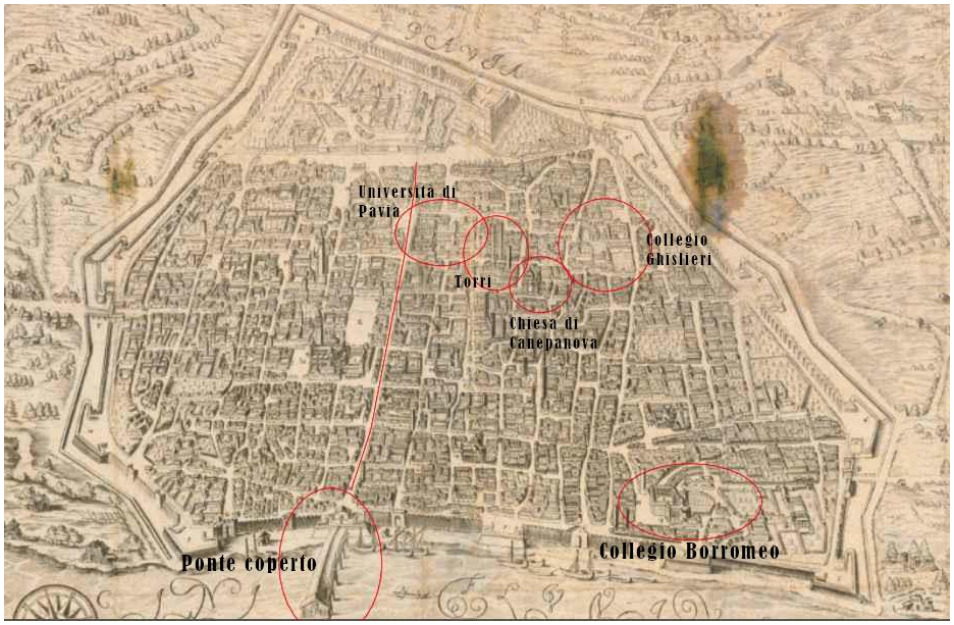


Figure 3.36 An ancient map of Pavia drawn by Ottavio Ballada, 1617.

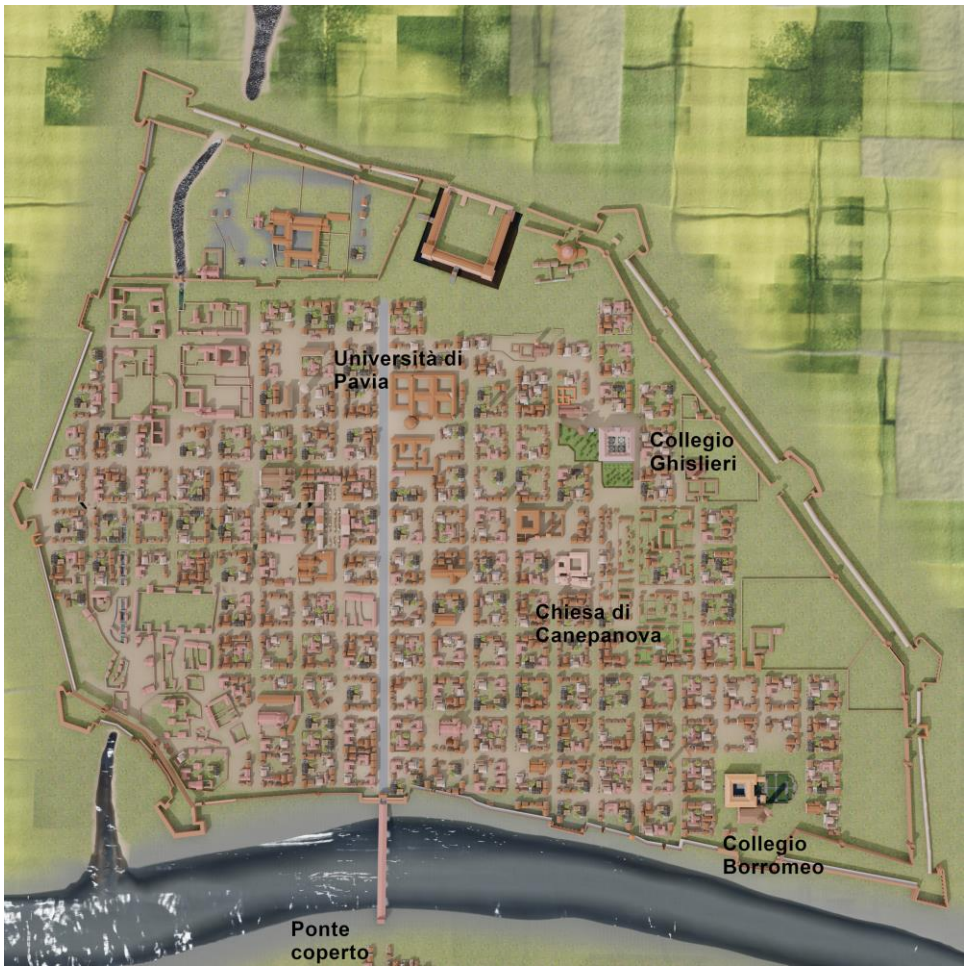


Figure 3.37 3D representation of the mid-1500s city.

3.2.2 Populating the city

Placing characters on a scene is not a difficult problem to solve, you can always manually take each person and place them anywhere and anyway you want, but if you consider populating an entire city, which implicates hundreds or even thousands of people, this could take an exceedingly long time.

So, apart in some cases like characters in the foreground, the focus should be how to find the quickest and effective method to automatically scatter many characters that also gives you access to the largest set of controls to tweak the algorithm.

3.2.2.1 Scatter Object

Object Scatter is an addon installed in Blender that allows you to scatter an object on any surface. By dragging the mouse over the parts on which you want the selected object to be scattered, the drawn lines tell the software where to place the copies of the object.

There are several settings that can be changed to achieve the desired result: number of object instances, rotation, randomness in rotation. Once the operation is confirmed, the objects are not individually editable, and this becomes a problem if you must arrange intersecting objects.

3.2.2.2 Particle system

Particle systems generate a set of particles randomly placed on a mesh. Furthermore, Hair particles systems create strands, which are static by default, can be used as placeholders for our characters on the streets and you can decide what kind of source to use (vertices, faces or volumes) to produce them.

Density weight map helps to establish how the particles are scattered on the mesh. Strands can be rendered as a single object or a collection, for each particle will be chosen an object inside the collection randomly. Rotation can be controlled using the normal of the faces/vertices, so the characters' rotation axis is perpendicular to ground, and adding a value of randomness to affect the final value.

Emitting from faces or volume, the issue of intersecting objects appears when you reach a certain number of particles that depends on the mesh geometry and the size of characters. Vertices can be a solution because the objects will be placed where the vertices of the mesh are in space.

One of the drawbacks of using particles systems is the slow rendering time because the computation is much more complex when dealing with an enormous number of particles that are simulated.

3.2.2.3 Geometry nodes

Geometry nodes are a new feature for Blender, they were introduced in February 2021 with Blender 2.92. In contrast with the other modeling techniques available on this software, geometry nodes use node trees to apply set of operations to a model. This tool does not already have all the features that you can use normally, but it is extremely powerful.

In this case, you can use these nodes in the same way as the particles systems: use the faces or the vertices of a mesh to create a random set of points and substitute them with any object or collection. The advantage is that there are not simulations involved, the algorithm generates the points once speeding up the rendering process.

Furthermore, there are interesting new features:

- Set a threshold distance between the generated points, so that the objects will not intersect one another.
- Replace points with a whole collection, this could be useful for scattering arranged groups of people.
- Use math operation to control the attributes of each point (position, rotation, scale) which can be used either for static geometry or animation.

Figure 3.38 Examples of crowd generated using geometry nodes

3.2.2.4 Diversity

Using a small set of characters to simulate the population of a city can increase the sense of monotony in a scene because, especially when multiple characters appear on the screen, there is a higher chance that some of them are the same model and maybe close to each other, making impossible to not notice it. Monotony is usually boring to stare at and should be avoided.

However, without modeling new characters there is a quick way to create the illusion of diversity.

- Applying a random rotation and scale to all the characters.
- Mirroring poses.
- Randomizing colors for clothes and hair.

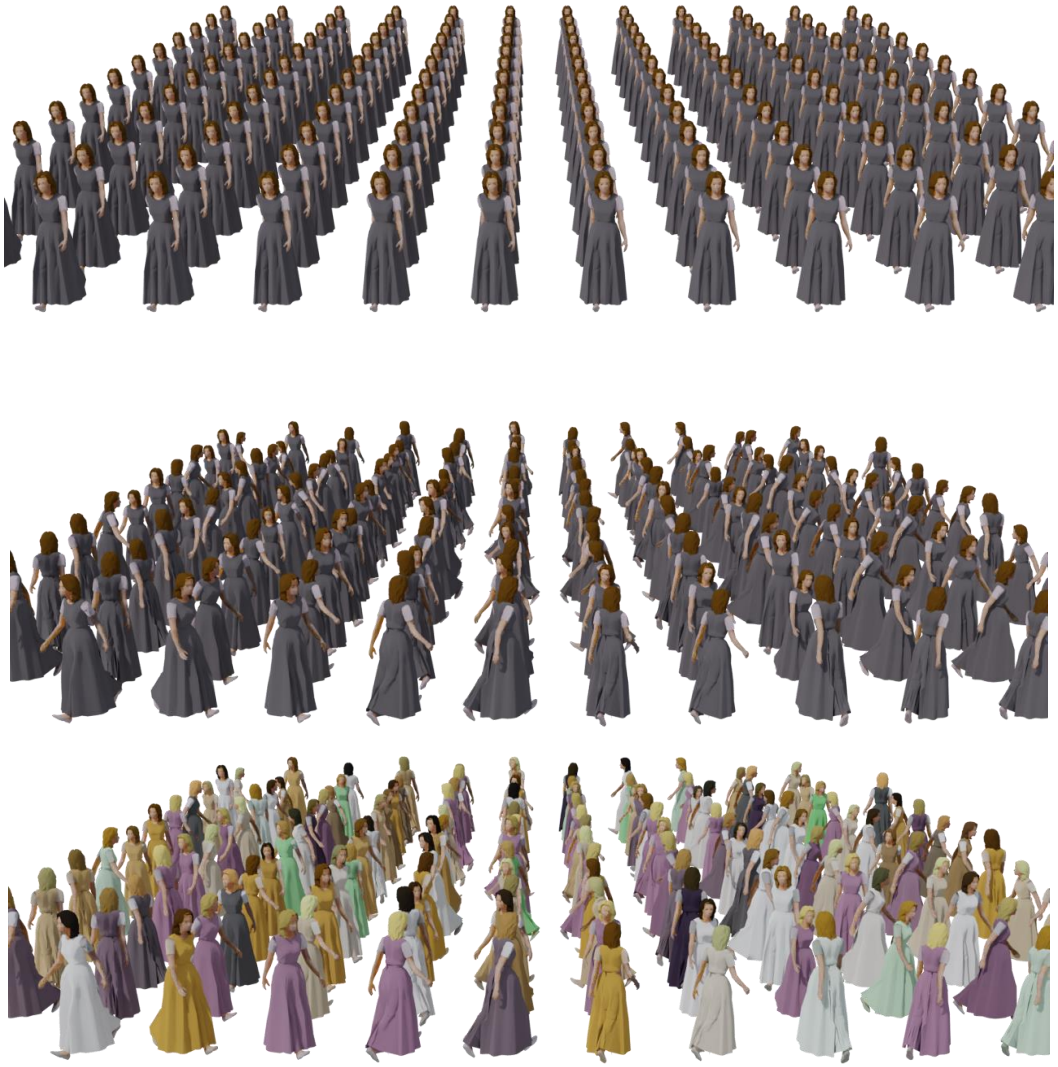


Figure 3.39 Monotony creates the military effect, in which the women seem a troop of soldiers. Randomness in rotation and mirrored poses simulate the behavior of people in crowds. Randomness in size and colors better reflect the inherent randomness of the real world.

3.3 Animation

3.3.1 Animating the camera

Camera motion can be handled easily with keyframing which gives a wide range of possibilities to control the animation.

- Location e rotation can be keyframed, using a key-pose scheme and editing the animation curve to control the interpolation.
- A curve object allows you to create efficiently a curvilinear path, controlling the radius and tilt of a curve. A follow path constraint will force the camera to follow a path. Combining them together, it makes a lot easier to move a camera along a curve than with location and rotation keyframing. It is also possible to create a system of more complicated dependencies to have more degree of freedom. For example, by applying the constraint follow path to an empty object and making it a parent of the camera, the camera movement is still able to follow the curve but has the possibility to move freely in space.
- Track to constraint will force the camera to be oriented towards a target object. It cannot be used to change the position of the camera, but it can be used to control its rotation.
- A camera rig is a skeleton used to control how the camera is positioned. The Root Bone is the parent of the entire rig, and all the children will follow its transformations. The Control Bone is the bone that will translate the camera around. The Aim bone is used to track an object.
- Walk / fly navigation is an operator that allows you to navigate the scene with first person controls. With a combination of keyboard and mouse, it works like any other navigation system available in 3D games (move forward, backward, left, right, jump, rotate camera). This operator can be applied to the camera view and using the auto keyframe function while an animation is playing, the software will record the attributes of the cameras at each frame in which they change value, creating a record of the walk-through.

3.3.1.1 Planning shots

Most of the videos of virtual tours on the web use a single camera moving throughout the city. However, the strategy chosen to film this itinerary is to use different cameras, each with its own function and technical characteristics (focal length, constraint) in order to make the animation system more versatile and easier to handle. Using different shots allows you to focus more on details (what is changed about the city), showing the buildings from different angles, without overwhelming the viewer with too much information at the same time.

3.3.2 Effects of destruction and construction

3.3.2.1 Build modifier

The build modifier is a fast and easy tool to make objects gradually appear or disappear. It takes the faces of a mesh, ordered by an index, and creates an animation based on an interval of frames.

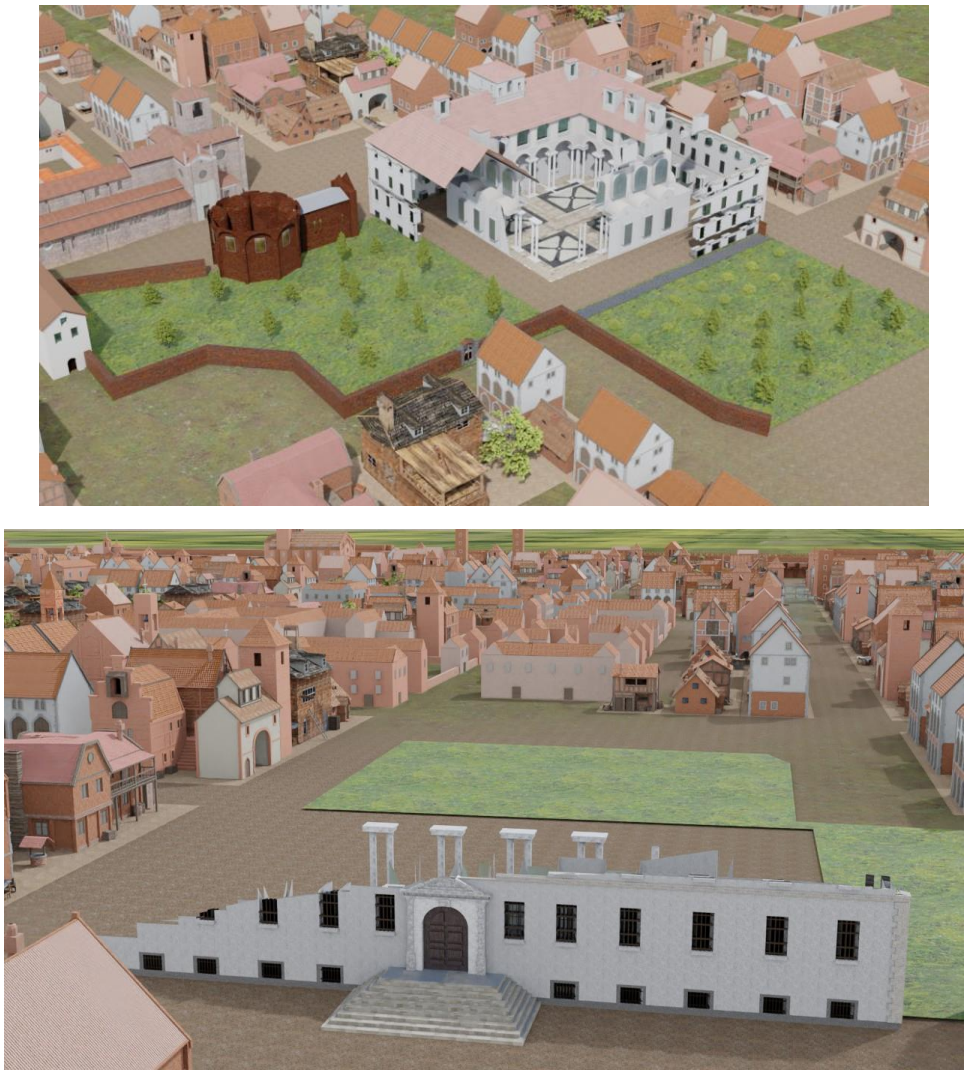


Figure 3.40 The build modifier applied to the Collegio Ghislieri

By default, the sequence of appearance/disappearance of the faces is driven by the order of creation, but Blender offers algorithms to sort the elements.

3.3.2.2 Materials

Materials can be used to control how the object is rendered and can be set to be transparent, so that light can pass through any objects using the material. Transparency establishes how much an object interacts with light and it is controlled using an *alpha channel*, where the value 0 means "fully transparent" and 1 "fully opaque". The alpha channel can be driven using a grayscale mask texture, which is mapped onto the model and can be animated.

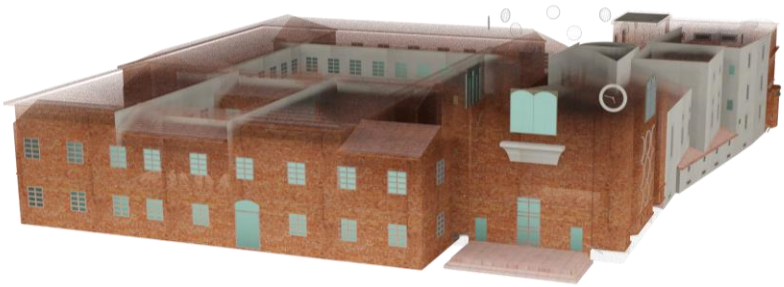


Figure 3.41 Combine a shader with a transparent BSDF and use a gradient mask to control how much the shaders are mixed; an object can transit from fully opaque to fully transparent.

3.3.2.3 Dynamic paint & Mask modifier

The mask modifier takes as inputs a vertex group, which contains a set of vertices and assigns a weight to each one of them, and a threshold which is compared to the weight of the vertices to make them appear or disappear. Weight paint mode allows to paint directly on the mesh and change the weights and animating the threshold value the mask modifier will decide which vertices hide.

Another approach uses Dynamic paint to turn an object in a brush and the object to be masked in a canvas, setting the surface type to *Weight*. The brush can now affect the weights of the vertices according to their position in space and more precisely, the brush will interact with the portion of the canvas, derived from their intersection. Brushes can be animated, moving the object around or changing its shape with shape keys or modifier.

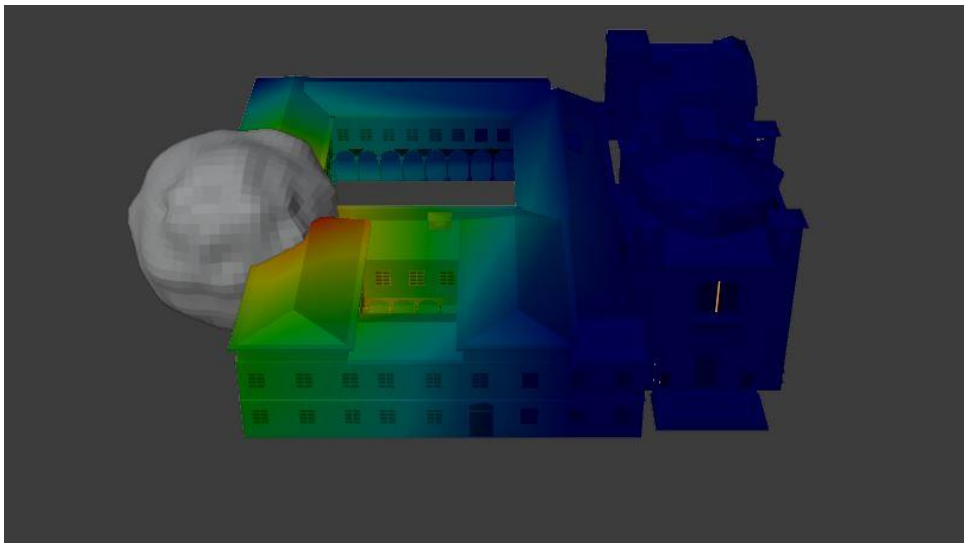


Figure 3.42 An object is used as a brush and paints the weight on the target.

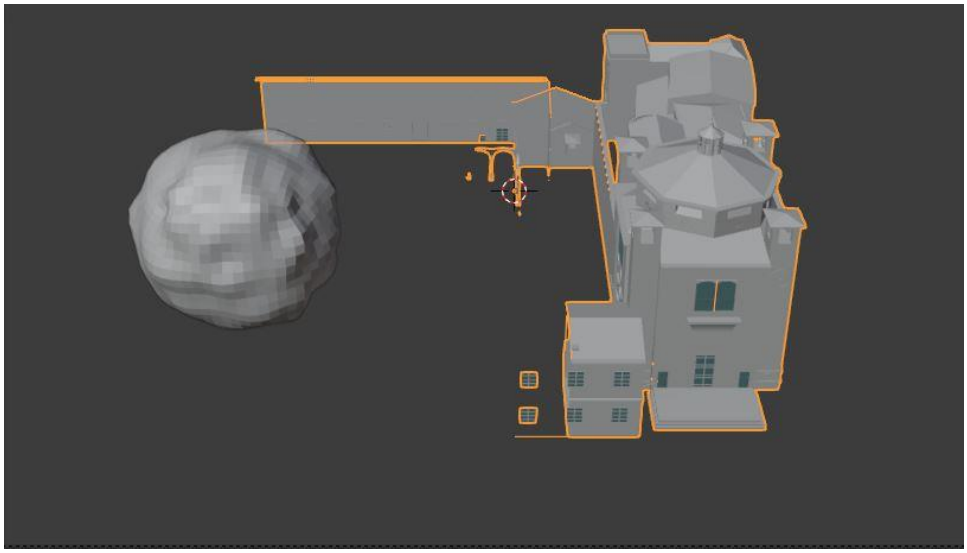


Figure 3.43 The mask modifier analyzes the weights to decide which vertices will be hidden.

3.3.2.4 Explode modifier & Collision

The explode modifier uses a *particle system* to attach the faces of the mesh (either the original faces or the new faces generated by cutting the mesh based on the location of the emitted particle) to the particles. The particles can be controlled with *physics*, so they can have mass and be dragged down by gravity according to Newton's law. Gravity is not the only force that can interact with particles, there are other type of forces (wind, turbulence....) which can be used to create more complex dynamics.

The resulting effect is an object broken into pieces and the pieces will move based on the forces that interact with them or objects that have Collision enabled, such as the ground on which the particles will bounce and stop their motion due to friction.

Figure 3.44 Particles emitted from the walls will break the mesh into pieces and bounce on the ground with collision enabled.

3.3.2.5 Cell fracture & Rigid body

The Cell Fracture operator generates fractures based on the geometry of a mesh. It works using a combination of Boolean operations to divide an object in many parts and it needs a source which can be the vertices of the same object or another object, a particles system, or lines drawn on the model using the grease pencil.

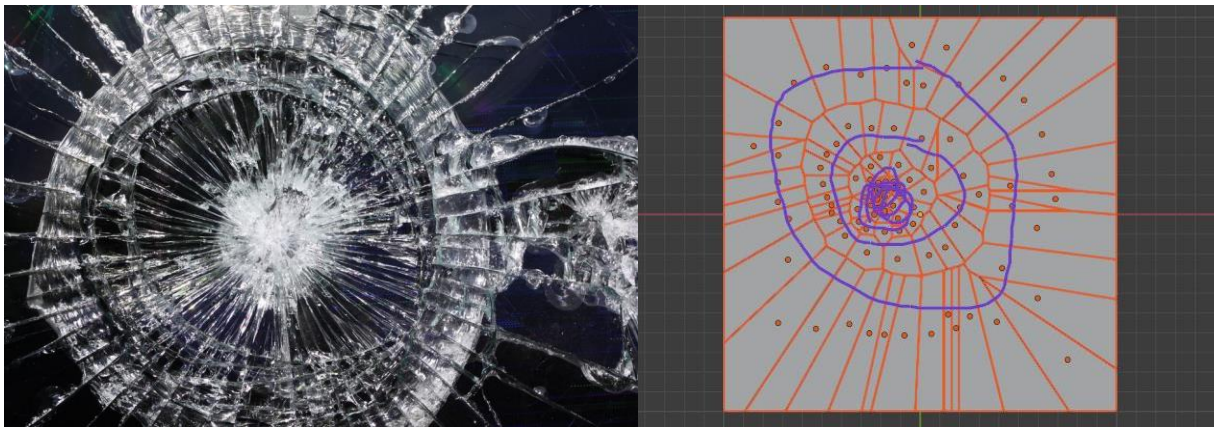


Figure 3.45 Fracture patterns can be achieved drawing lines on the surface and the algorithm will use them as a guide.

Once the object is fractured in pieces, you select all of them and add rigid body physics making them all of type *Active*. It is important to check that all pieces have the origin point in the center of geometry, otherwise they will act strangely. You can also assign a mass based on the volume of the piece and some density preset values (brick, iron, ice...).

Those fractures need to interact with the ground which will have rigid body physics of type Passive, not being affected by any force.

3.3.2.6 Shape keys

Shape keys are data structure that stores information about the position of the vertices inside a mesh. They can be used to assign different shapes to the same mesh, allowing to transit from one shape to another. It is useful to animate faces, which involve specific and frequent muscle movements, and it's faster to use, sliding a value between 0 and 1, than posing the character every time. Shape keys are very powerful and precise, but precision comes at high cost in terms of time.

3.3.2.7 Destruction of the Visconti castle

While the other monuments are mostly animated using the Build modifier (in case of construction) or a simple explode modifier (for the medieval walls), the destruction of the Visconti castle has a more complex approach. These animation starts with an army of soldiers who arrange the cannons in front of the castle, then cannons fire and the castle falls.

Figure 3.46 The shot begins with a close-up of a cannon pushed by a soldier

The cannon is animated using *keyframes*, while the wheels, the soldier and the camera are parented to the cannon so that they inherit its transformation. The wheels rotation is obtained mapping the distance traveled by the cannon to the angle. The camera is still free to move around while the parenting forces it to follow the cannon movement.

Creating a simple rig for character to control only certain parts of the mesh like head, torso, or an arm, and linking the bones to mesh using vertex weight paint, characters can be animated with small movements. Inverse and forward kinematics are the most common bone constraints as they help to produce movements quickly. Bones are usually linked by a chain; a bone can be a parent of multiple bones and/or be a child of another bone. Forward kinematics mean that moving the parent moves the child and its names derives from the fact that the transformations of the bones only influence bones that are further forward down the chain (moving your right forearm moves your right hand, moving your hand does not move your hand). Inverse kinematics is the opposite and allows the

transformation of a bone near the end of a chain to influence bones that are further up the chain (moving your hand moves your arm). For example, to obtain the motion in figure 3.47 forward kinematics needs to pose the arm, the forearm, and the hand, while inverse kinematics needs to know only where the hand is located.

Figure 3.47 The soldier on the left turns his head to the right and raises his arm to give the fire signal

The firing of the cannons can be simulated using smoke and fire simulation, which are created using Mantaflow, the new physically based fluid simulation framework in Blender for gas (smoke & fire) and liquid simulations.

Figure 3.48 Smoke behavior after the firing of cannon is simulated using particles

The first thing that happens when cannon fires is an explosion and then the smoke slowly covers the entire scene. These simulations can be controlled using *particles systems*, since the particles will be used to generate the gas, their behavior can be a preview of how the smoke will behave.

An explosion involves a mass of smoke and fire that develops rapidly and moves with a certain speed, which can be modeled as an emitter that generates thousands of particles in fractions of a second with a high initial velocity along the direction of fire. After that, there will be a cloud of smoke that, once the energy generated by the explosion has dissipated, will be affected by various forces, including wind, turbulence, interaction with the particles of air. The behavior of the cloud can be simulated with particles that are emitted in the direction of the shot with an additional random value that will help the smoke to spread in space and Force field objects that will affect the movement of the particles, which otherwise will only interact to the ground until they stop.

The cannonball can be animated as a *parabolic motion*; to do that, the ball will be marked as an active rigid body, which allows Blender to overwrite its position anytime the simulation is running. In this situation, the ball will be affected by gravity and just fall which is not what is wanted. To create the parabolic motion, the ball needs a force that will interact with gravity and this force can be animated manually. So, at the beginning the ball moves because of the keyframes then the animation system switches from keyframes to the rigid body simulation, generating the parabolic motion. Additional motion can be achieved reusing some parts of the castle as passive rigid body and the cannonballs will interact with them. The last step is *baking* the rigid body motion into keyframes.

Now that the motion of the cannonballs is solved, you can find the spots where the castle is hit. The destruction of the castle is performed using a particles system and an explode modifier, but to make more realistic the parts hit by the cannonballs are isolated from the rest and it is easier to control the effect. Additional particles systems can be used to add to dust to the broken pieces and, as before, smoke. The cannonballs and the ground can be marked as Collision objects and will influence the motion of the particles.

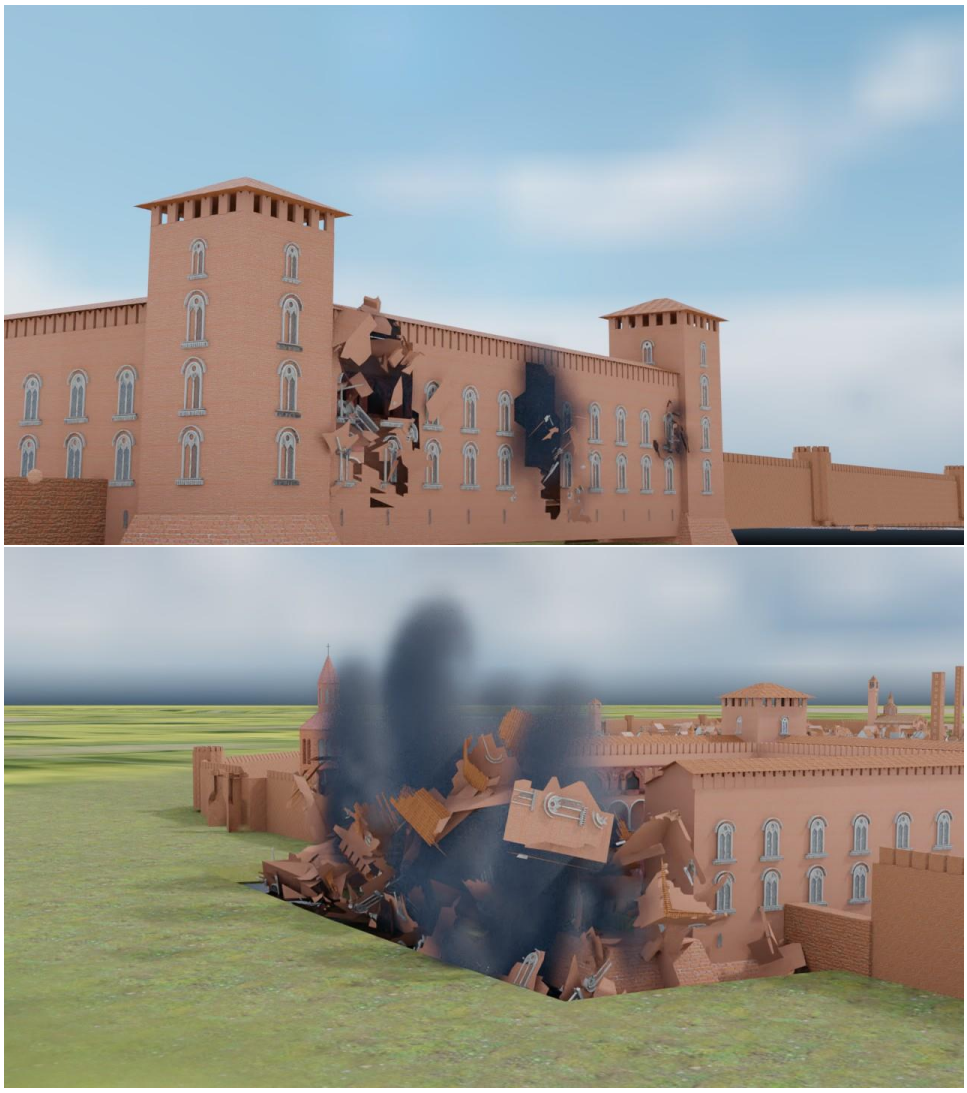


Figure 3.49 The castle hit by cannonballs

3.4 Rendering

3.4.1 Lighting

Before the rendering process, the scene needs a lighting setup. There are various possibilities do this in simpler or more complex ways involving a combination of lamps, emitting objects, images or procedural textures. Being an outdoor scene, the goal is to have as natural light as possible to reflect the lighting deriving from the sun's rays.

The Pro Lighting Skis add-on is a useful resource for outdoor lighting. A sun lamp just illuminates all the objects in the scene from one logical direction, while real lighting is produced by millions of light

sources. The sky shows shades of blue from the atmosphere, reflected light from the clouds and a horizon gradient and light comes from everywhere with different hues, saturations and brightness.

HDR lighting uses a single 360-degree photograph which stores more accurate information about lighting since it is based on a real photo.

Pro-Lighting Skies contains a collection of HDR, including sunny, cloudy, overcast, morning, sunset, nighttime, and evening skies.

It provides controls to change the intensity of the sunlight and its inclination, the intensity and hue of the light of the sky.

3.4.2 Rendering

Rendering is the most delicate part of the pipeline because every decision made in the previous phase of the project will be combined into the final image/video. Rendering is always a trade-off between quality, quantity, and time.

For the creation of the material that will be used to edit the final video, the number of frames will be limited to a maximum of 3000, which at a frame rate of 25 frames per second is equivalent to a 2-minute video. Cycles is the render engine used for the final rendering, and the number of samples used is small (128 samples - some professional renders may have a minimum of 1000-3000 samples).

With this rendering options each frame can be rendered between 3 and 10 minutes; so, to create a 1-second of video it takes between 75 and 150 minutes (about 2 and a half hours), consequently for the entire 2-minute video from 6 to 20 days (about 3 weeks), if the video does not contain any error and some parts of the video need to be rendered again.

3.4.2.1 Noise

Noise is the principal reason to increase the number of samples, if it is too low Cycles does not have enough information to make correct calculation, creating random variation of brightness or color information in adjacent pixels where it should not be present.

Unfortunately, increasing the number of samples to achieve a better computation of light make the rendering time slower. Each sample requires approximately the same amount of computation, so if you want to double the number of samples the expected rendering time will be double.

Luckily, there exists a tool, called denoiser, which analyzes the pixels inside a section of the image (tiles) to detect noise and smooth it. Using denoisers can save a lot of time by keeping the number of samples low (not too low), but it still needs enough information to avoid that the generation of a smear effect where the amount of noise is high.

The denoising process filters the resulting image using information about the objects gathered during rendering to get rid of noise, while preserving visual detail as well as possible.

NLM is a non-local means algorithm, it computes the value of a pixel based on the weighted averages of all the pixels assigning larger weights to pixels that are similar in terms of their local neighborhoods or patches.

Intel Open Image Denoise is an open-source library of high-performance, high-quality denoising filters for images rendered with ray tracing. It filters out the Monte Carlo noise inherent to stochastic ray tracing methods like path tracing with a collection of deep learning based denoising algorithms and it runs on CPU.

OptiX uses GPU-accelerated artificial intelligence to dramatically reduce the time to render a high-fidelity image that is visually noiseless. It works only with Nvidia GPUs.

3.4.2.2 Light path

A ray-tracing engine follows the path of a ray to determine which points it hits and with which intensity. Every time a ray hits an object it will bounce and lose some energy; ideally the number of bounces should be infinite, in practice a smaller number may be sufficient because most of the energy will have already been dispersed, and further bounces will have no visible effect. Intentionally leaving out some light interactions leads to a faster convergence.

3.4.2.3 Resolution & Tiles

Camera view coordinates are transformed into pixel coordinates and consequently, what is seen on the camera preview shows on the 3D viewport will be transformed in an image. Images has a specific number of pixels on the x-, y- axis which depends on its resolution. High resolution images take more time than low resolution ones because more pixel values must be computed.

Currently, Cycles, like many other engines, subdivides the image in sections called tiles. Depending on what device you are using for rendering, different tile sizes can give faster renders. For CPU rendering smaller tiles sizes (like 32 × 32) tend to be faster, while for GPU rendering larger tile sizes give a better performance (like 256 × 256). A smart decision is to use a tile resolution that is a divider of the image resolution, so that the resulted grid has cells with the same size.

Recently, Blender announced the version 3.0 that will replace Cycles with Cycles X, which shows some promising results in terms of rendering time. It is still under development, but one of the new features is that there will be no tiles, so the image will be rendered all at once.

3.4.2.4 Render output

The result of the rendering process is a raster image, a rectangular array of values, called pixel values, all of which have the same type. These pixel values may be real numbers representing levels of gray (a grayscale image), or they may be triples of numbers representing mixtures of red, green, and blue (an RGB image), or they may contain, at each pixel, other information in addition to color or grayscale data.

Image, like other files, are stored in many formats which establish how to store and organize the information about the pixels. For example, in a colored image, a pixel is identified by a triplet of RGB values, each channel with a fixed number of bits. Images are usually compressed to reduce their file size.

- Bitmap stores two-dimensional digital images both monochrome and color, in various color depths, and optionally with data compression, alpha channels, and color profiles. The speed at which images are read or written to disk is faster when compared to other file types, but it consumes a lot of memory space on the disk.
- Open EXR is an incredibly powerful image file format that is both high dynamic-range (HDR) and surprisingly space efficient. It stores in a single file an image sequence corresponding to the render passes (different interactions between objects). It supports different pixel sizes, including 32-bit unsigned integer, 32-bit, and 16-bit floating point values, as well as various compression techniques which include lossless and lossy compression algorithms.
- JPEG is a lossy compression format and has become a de facto standard that is especially appropriate for natural images containing gray values or RGB values. Compression reduces the size of the file (measured in bytes), losing information and consequently, quality. When comparing images, it makes impossible to know whether the images are different or if the difference is an artifact created by the JPEG compression algorithm.
- TIFF is a non-compressed lossless format. It stores multiple channels, each with a description of its contents. For image editing and compositing tools, in which multiple layers of images are blended or laid atop one another, a TIFF image provides an ideal representation for intermediate (or final) results. It preserves more quality at the highest resolution but require more disk space for storage.
- PNG comes close to TIFF in quality and is ideal for complex images. It can support 32-bit images with up to 8 bit per channel for each of the red, green, blue and alpha (RGBA) channels.

3.5 Post-production

Post-production refers to the final stage before creating the final output, in this case a video. Post-production will be done either in DaVinci resolve and Blender, which gives a wider range of tools to edit the video.

3.5.1 Color grading

Blender has a color management option in the rendering panel, which was set to Filmic. Filmic compresses the scene referred linear radiometric energy values and the gamut for high intensity values.

The rendered images are loaded on DaVinci, and you can use the Color Page for color balancing and color correction. It uses a node graph in which each node receives an RGB signal in input and sends an RGB signal as an output making it possible to fragment the color grading process into a pipeline where each node has its own function.

3.5.1.1 Tools

Primaries wheels (and the corresponding bars and log wheels) control the tonal and chromatic values of an image based on three luminance ranges (highlights, midtones, and shadows).

- Lift targets the shadows of the image.
- Gamma targets the midtones of the image.
- Gain targets the highlights of the image.
- Offset affects the entire image uniformly.
- Master wheels are the dark horizontal sliders beneath the Color wheel that control the YRGB values of those respective ranges.

Custom curves give precise control over the chromatic values of an image based on RGB and luminance curves.

- Custom RGBY adjusts luminance and color in relation to tonal areas
- Hue vs Hue changes a certain shade to another.
- Hue vs Sat alters the saturation of any hue in the image
- Hue vs Lum alters the brightness of elements of a specific color.

Scopes provide graphic readouts of the luminance and chrominance values of an image for the purposes of balancing and matching.

- Waveform shows the luminance and color values superimposed on each other in the current frame. Colors appear white if individual channels have the same intensity level.
- Vectorscope shows a circular graph that represents the hue and saturation levels in the image. This is usually useful for checking that skin tones have not taken on unwanted colors, such as green, yellow, or magenta.
- Histogram provides a graphical representation of the tonal distribution of each channel. It is ideal for evaluating hue, identifying overexposed elements in the highlights or flattened elements in the shadows, and for adjusting brightness and contrast.
- CIE chromaticity determines whether the colors reside within the project delivery format limits specified in the settings.

3.5.1.2 Balancing and Shot Matching

The first stage of grading, which is categorized as a primary grade because affects the entire image, must adjust shot luminance and chrominance to create a level starting point for your grade.

It involves techniques known as normalization, balancing, and shot matching.

Normalization and balancing involve creating a neutral starting point for all the clips, where there is not a strong color dominance and normalize the distribution of color values.

Shot matching means creating a link between all the clips through color and luminance. It guarantees the "continuity" in the scene, especially in terms of light and shadow.

3.5.1.3 Setting Tonal Range

The human eye is particularly sensitive to light sources and shadows, which is why it makes sense to begin adjusting them. The waveform scope can serve this purpose. The vertical axis of the scope represents the entire luminance range of the image. The bottom of the display represents the blackest black (0 in a 10-bit depth signal), and the top represents the whitest white (1023 in 10-bit). Everything between represents the full midtones range of the image in a grayscale format.

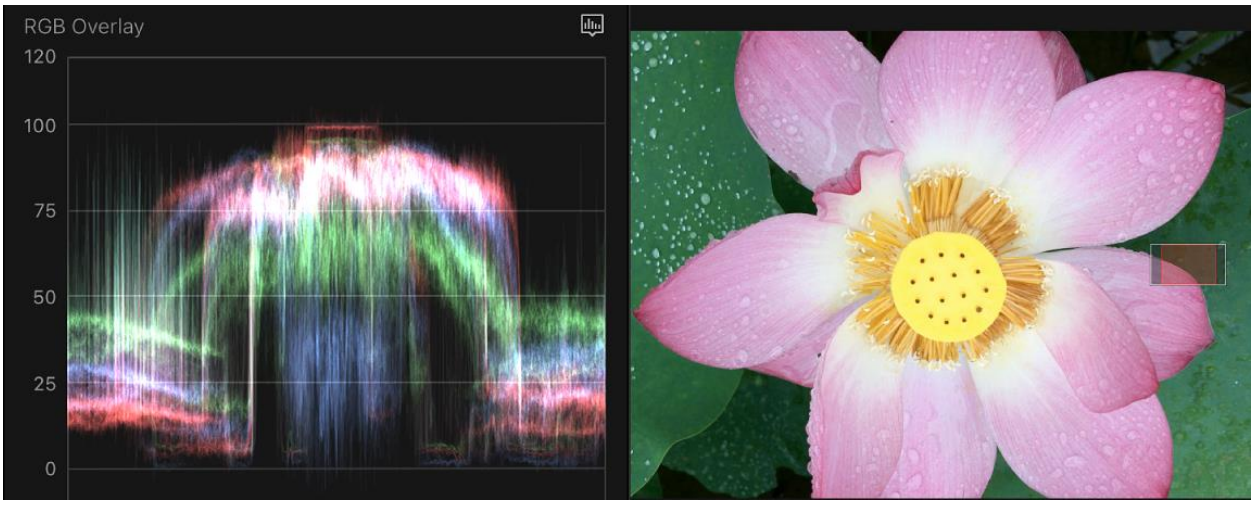


Figure 3.50 The horizontal axis represents the columns of pixels of the image and the vertical axis the luminance of the correspondent pixels.

Each color channel is overlapped in the trace. White in the trace indicates that each channel has an equal intensity. When adjusting just the luminance in an image, the RGB channels in the waveform can be disabled and you see only a single signal.

Any part of the trace that goes below 0 (black point) and above 1023 (white point) in the sRGB gamut will be clipped, which will result in a loss of image detail.

Normalizing a footage implies bring the shadows at around 5-10% above the black point (0) on the scope, while the pure white should stop well under the white point (90%), leaves space to super-white elements such as blown out headlights, lens flares, or metallic specular highlights that can extend beyond the white point. If the image is still too dark or too bright means that the midtones are too compressed, so the gamma wheel can help stretching them.

3.5.1.4 Balancing Colors

After adjusting the tonal range, you should examine the relations between the RGB values on the image. For example, deep shadows, which are supposed to be black, should have the value on each channel, otherwise we will have a color dominance. The same reasoning applies to something that should be white like clouds, this will help to make the image more neutral.

3.5.1.5 Look

A balanced image makes easier to manipulate colors and create a specific look. Color when used correctly can guide the viewers to what is important, adjust the mood or the atmosphere or be used to tell a story.

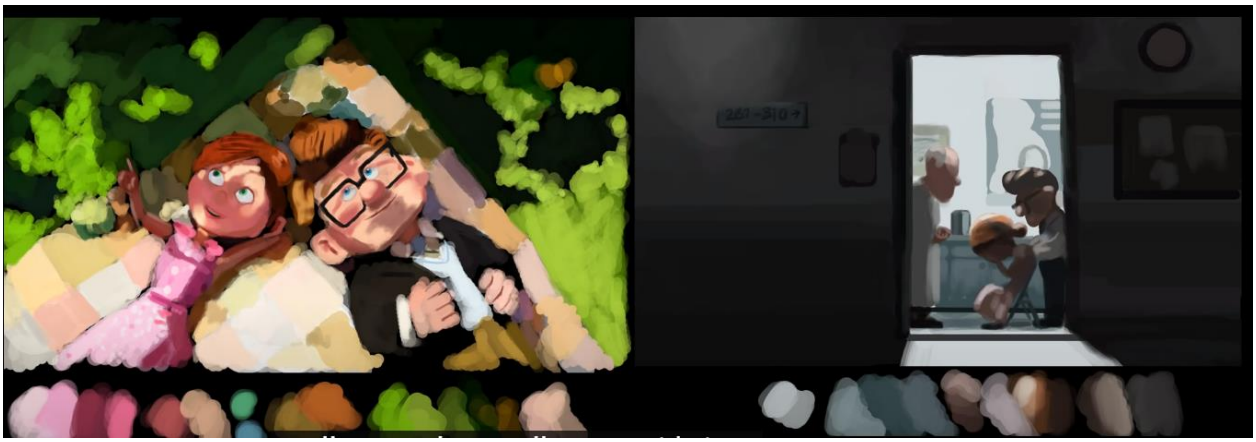


Figure 3.5 l Scenes from "Up". Vibrant colors express a joyous happy moment in their life. Very desatured colors makes the scene sadder.

In this case, we will create a node to adjust the saturation of the overall image to make the colors a little bit more vibrant.

3.5.1.6 Contrast

Contrast describes the relationship between the darkest and brightest parts of an image. Increasing the contrast means deepening the shadows while brightening up the highlights, sharpening the differences between pixels and making object more distinguishable. The pivot control establishes which levels of the luminance scale the contrast will affect more.

Figure 3.52 The contrast between light and dark was the foundation of the chiaroscuro technique, as demonstrated by the masters Caravaggio ("The Calling of Saint Matthew", 1599-1600) and Rembrandt ("The Philosopher in Meditation ", 1632).

Figure 3.53 Progressive look of the image from original to final grade, normalizing the colors, saturation and contrast

3.6 The final products

The virtual tour of the city, as specified in the project presentation, is a recording of a camera moving in the scene and the rendering is done at a resolution of 1280x 720 px, using 128 samples and the OpenImage denoiser. The final step is editing all the sequences into a single video, which can be seen on YouTube at https://www.youtube.com/watch?v=EC-OZi1o-iI&t=25s.

The timeline, instead, is different from the recording because the viewer should be able to travel through time and as the city of Pavia evolves, new buildings rise, and others are destroyed. For convenience, the changes are grouped together into four temporal sequences:

- 1500-1530: the north wing of the castle collapse during the battle of Pavia, the monastery of San Felice is built.
- 1530-1550: the medieval walls are destroyed, and the Spaniards built the new walls, Palazzo Broletto was expanded with the construction of two loggias.
- 1550-1575: the construction Collegio Borromeo and the Collegio Ghislieri has started and ended.
- 1575-1600: the church of S. M. di Canepanova is completed.

The simplest solution is to render the sequences and create the transitions while editing the video, grouping together the sequences and adding new sources to support and make the video more understandable, such as maps that show how the city changes in a sequence and which monuments are involved, titles for each animation sequence, labels and markers (like YouTube allows to add to a video) that indicate in which period the changes take place.

The rendering is done at a resolution of 1280x 720 px, using 40 samples and the Optix denoiser.

3.7 The future….

To create interactive environments, there are different software which can import a scene and provide navigation tools. Most of them are dedicated to real-time immersive 3D architectural visualization, such as Twinmotion, or game development, such as Unity, Unreal Engine, Godot.

In the past, Blender also had its personal game engine, but the Blender Game Engine was removed from Blender in 2018 with the release of the version 2.8 to focus on providing a good support for external game engines like Godot, Armory3D and Blend4Web.

However, there is a possibility to create a real- time walkthrough also in Blender, combining camera movements and animation sequences.

3.7.1 First Person Experience

Widely used in videogames, first person perspective is any graphical perspective rendered from the viewpoint of the player's character. Camera view is linked to the user view, creating an immersive experience in the environment. A person can, then, walk as the camera moves forward, backward, left, or right and move his head as the camera tilts, pans or rolls.

The walk navigation function allows you to move the camera in space using the W, A, S, D keys and rotate it moving the mouse pointer, while using another customizable keystroke you can trigger the next sequence until the last one resets the timeline to the 1500 and the cycle restart.

3.7.2 Real-time render

Eevee is Blender's real-time render engine built within Blender using OpenGL focused on speed and interactivity while achieving the goal of previewing PBR (Physically Based Rendering) materials. Since it uses rasterization, it is faster than a ray tracing engine and focus only on estimating the way light interacts with objects and materials. In this way a user can preview materials and objects in real-time and navigate the viewport while the scene is rendered.

3.7.3 Pavia navigation add-on

Blender is a software based on python language and includes a console to run commands or python scripts. Python scripts are useful to create additional functionalities, called add-on, to the Blender library. To install a new add-on, you need to go the Preferences tab and once it is installed you must enable it.

The objective is to create an add-on that calls the function Walk navigation, which contains all the command to navigate the scene, and plays the sequences in order every time you press a specific key, allowing you shift timeline while you are walking.

3.7.4 Achieving a more realistic look

Eevee is faster than Cycles but the speed increases at the expense of the ability of creating a photorealistic look. Since the render is in real-time, the post-production process is not a solution available.

However, one of the big differences between Eevee and Cycles is the accuracy of light and shadow because Cycles considers random samples of light that bounce around the scene and make accurate computation to calculate the intensity of light on the surfaces. If the position of the light source does not change and effects that depends on the position of the viewer are discarded, maps can be used to fake lighting.

Cycles provides a tool to create images that stores the result of the render passes (normals, albedo, shadows). This process is called baking and it is usually used to create high resolution maps from high poly meshes to apply them on the relative low poly meshes.

Complex object may contain multiple material applied; in this case the baking process can combined all the information gathered during the rendering to create a single material.

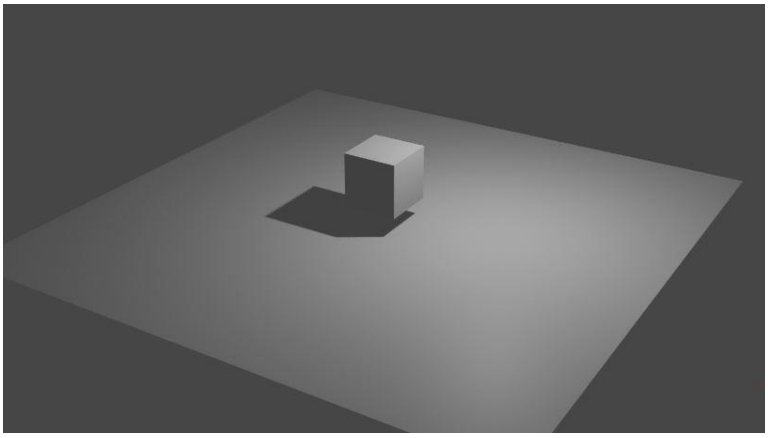


Figure 3.54 Rendering the scene with Eevee

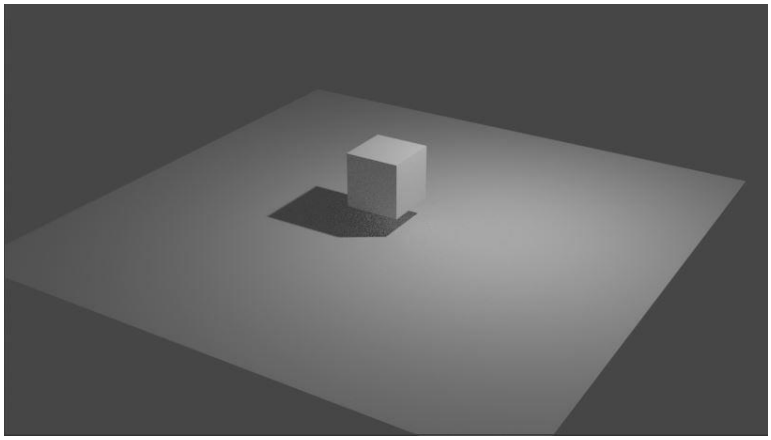


Figure 3.55 Rendering the scene with Cycles

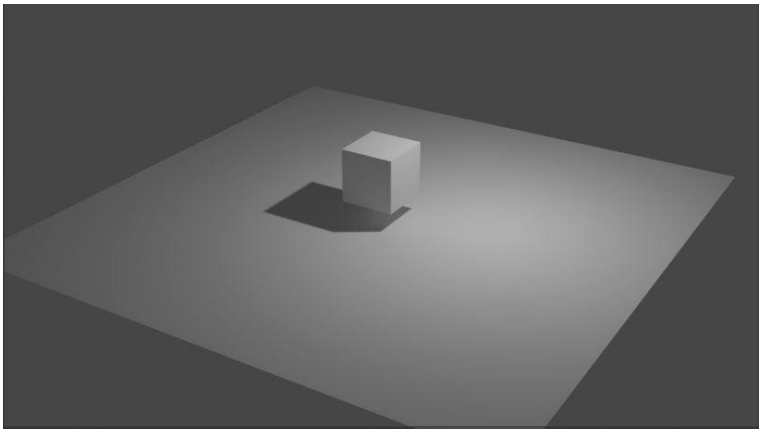


Figure 3.56 Rendering the scene with Eevee after baking

Conclusions

The "Virtual reconstruction of Pavia in the 16th century" project is gradually getting closer to its objectives, but before taking it to the next steps it has extensive room for improvement. If rendering animations is the only current solution available to create a final product to share with the public, the issue here is not just a matter of hardware power. Just to compare some numbers, the PC used during this project has 32 GB of RAM and the average size of the files containing all the models is 7GB, that seems handleable, however, even with no other programs opened, Blender often freezes. hangs or crashes making very difficult to work on the models, which are usually imported in a separate file, modified individually and then reimported in the original file. The rendering process causes the same issues because memory usage is a key factor. Using small number of samples, controlling the number of light bounces and limiting the resolution of images to very low resolution still generates very high peak of memory usage that must be balanced with the tile resolution and other options with the aim of speeding up the otherwise too slow rendering time (at the cost of increasing the memory usage) in order to not make the software crash. The answer seems easy (to blame the hardware), but, before thinking about an upgrade, there is still one thing to try: optimizing the asset. It is not easy, and it is definitely time consuming, but it has a lot of benefits.

The retopology of the models, which aims to reduce meshes of millions of triangles to few thousands maintaining almost the same quality, will improve the performance of the software (the more polygons are rendered, the more computations are necessary to move the vertices when navigating the scene, playing animations, etc.) and helps to set boundaries, especially when the final product will run on a pc or a phone which can have different requirements. Low poly organized mesh not only will allow previewing real time animation, which is a great advantage when you want to animate something, but will improve rigged animation and modifier results, making everything look better.

The optimization of the asset can be achieved also using instancing, which means making copies of an object with all their data linked to the original object or even copies of an entire collection, which saves a lot of memory in relations to just duplicating data blocks.

Blender is a program with many purposes but creating applications or games in it is not its main purpose. In this sense, Blender is working to have a good support for external game engines like Godot, Armory3D and Blend4Web, but in general it is possible to export the model using the standard formats. For example, architectural real time walkthrough software are powerful tools which allow to navigate the rendered scene in real time, add scripts to handle specific events and even transfer the experience from a monitor to a virtual reality visor, transforming keyboard and mouse inputs into a real walk.

Finally, software and hardware are key factors that also depends on the resources available for the project, but keeping the models organized and optimized should be important as well, especially on large projects where small errors spread along thousands of models could impact severely the performance of any machine.

Bibliography

- Itinerari monumentali a Pavia, testi di Donata Vicini, a cura dell'Assessorato alla Cultura del Comune di Pavia e dell'A.P.T. Pavia, Tipografia Popolare - Pavia, 1990-1993.
- FONDAZIONE GHISLIERI - La storia - Fondazione Ghislieri, https://www.ghislieri.it/fondazione-ghislieri/benvenuti/centro-didattica-universitaria-ricerca/la-storia/
- Collegio Ghislieri | Collegi di Pavia, https://www.collegidipavia.it/collegio/2/collegio-ghislieri
- FONDAZIONE GHISLIERI - PALAZZO DEL COLLEGIO - Fondazione Ghislieri, https://www.ghislieri.it/fondazione-ghislieri/luoghi/luoghi/
- Almo Collegio Borromeo Pavia | Storia del Collegio, http://www.collegioborromeo.it/it/almo-collegio-borromeo/storia-del-collegio/
- EX Monastero di San Felice, http://www.paviaedintorni.it/temi/arteearchitettura_file/arteinstrada_file/PAVIA%20VINCOLI%20URBANISTICI_file/PAGINA%20%20ELENCO%20EX%20MONASTERI%20E%20CONVENTI_file/EX%20MONASTERO%20DI%20SAN%20FELICE.htm
- Ottavio Ballada, Mappa della città di pavia, (1617), http://graficheincomune.comune.milano.it/GraficheInComune/immagine/P.V.+g.+5-26
- James D. Foley, Andries van Dam, Steven K. Feiner,John Hughes. "*Computer Graphics: Principles and Practice*", Addison-Wesley, 2nd edition: 2005. ISBN 978-0-201-84840-3
- Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. "*Real-Time Rendering*", Natick: A.K. Peters Ltd., 3rd edition: 2008. ISBN 978-1-56881-424-7
- Peter Shirley, Steve Marschner. "*Fundamentals of Computer Graphics*", A. K. Peters, Ltd, 3rd edition: 2009. ISBN 978-1568814698.
- William Vaughan. "*Digital Modeling*", New Riders Pub, 2011 ISBN 978-0321700896
- E. Catmull, J. Clark. "*Recursively generated B-spline surfaces on arbitrary topological meshes*", (1978).
- Edwin Earl Catmull. "*A subdivision algorithm for computer display of curved surfaces*", (PhD thesis), University of Utah, (1974).
- Ivan Edward Sutherland. "*Sketchpad: A man-machine graphical communication system*" Massachusetts Institute of Technology, (January 1963).
- Henri Gouraud. "*Computer Display of Curved Surfaces*", Doctoral Thesis (Thesis). University of Utah, (1971).
- Bui Tuong Phong. "*Illumination for computer generated pictures*", (1975).
- Arthur Appel. "*Some techniques for shading machine renderings of solids*", IBM research center N.Y., (1968).
- J. T. Whitted. "*An improved illumination model for shaded display*", (1979).
- J. Turner Whitted. "*A Ray-Tracing Pioneer Explains How He Stumbled into Global Illumination*" (2018), https://blogs.nvidia.com/blog/2018/08/01/ray-tracing-global-illumination-turner-whitted/
- A. Michael Noll. "*Computer animation and the fourth dimension*", Bell Telephone Laboratories, Incorporated, (1965).

- The Computer Graphics Book of Knowledge, https://www.cs.cmu.edu/~ph/nyit/masson/history.htm#:~:text=Computer%20Graphics%20(CG)%20was%20first,and%20Boeing%20in%20the%201950s,
- Wayne Carlson. "*History of computer graphics and animation*", https://ohiostate.pressbooks.pub/graphicshistory/chapter/14-3-tron/
- Pixar History Revisited - A Corrective, http://alvyray.com/Pixar/PixarHistoryRevisited.htm
- Our story | Pixar, https://www.pixar.com/our-story-pixar
- Ian Phillips, Ben Nigh. How Pixar's Movement Animation Became So Realistic, Movies Insider, (2021), https://www.insider.com/from-toy-story-to-soul-evolution-of-pixar-character-movement-2021-3
- Nick Romano. A Visual History of Motion-Capture Performances on Film, (2014), https://screencrush.com/motion-capture-movies/
- Michael Burns. VFX: De-aging tech, Industry Trends, IBC, (2019) https://www.ibc.org/trends/vfx-de-aging-tech/5188.article
- Ran Kidron. The History of Aging and De-Aging on Screen, (2020), https://artlist.io/blog/the-history-of-aging-and-de-aging-on-screen/
- Devin Coldewey. "*How 'The Mandalorian' and ILM invisibly reinvented film and TV production*", (2020), https://techcrunch.com/2020/02/20/how-the-mandalorian-and-ilm-invisibly-reinvented-film-and-tv-production/?guccounter=1
- Olena, Altumea. A Brief History of GPU, https://medium.com/altumea/a-brief-history-of-gpu-47d98d6a0f8a
- Thomas Dreher. "*History of Computer Art*", http://iasl.uni-muenchen.de/links/GCA_Indexe.html
- Vashi Nedomansky, ACE. "*The ultimate history of CGI movies*", (2019), https://vashivisuals.com/the-ultimate-history-of-cgi-movies/
- Ian Phillips, Ben Nigh. 10 movies that changed CGI this decade, (2019), https://www.insider.com/movies-that-changed-cgi-decade-2019-12
- Gibson, Ian, and Jorge Bártolo, Paulo. "*History of Stereolithography",* Stereolithography: Materials, Processes, and Applications. (2011)
- Richard Trenholm. "*Visual effects company MPC reveals modern movie magic*", (2018), https://www.cnet.com/news/visual-effects-company-mpc-reveals-modern-movie-magic/