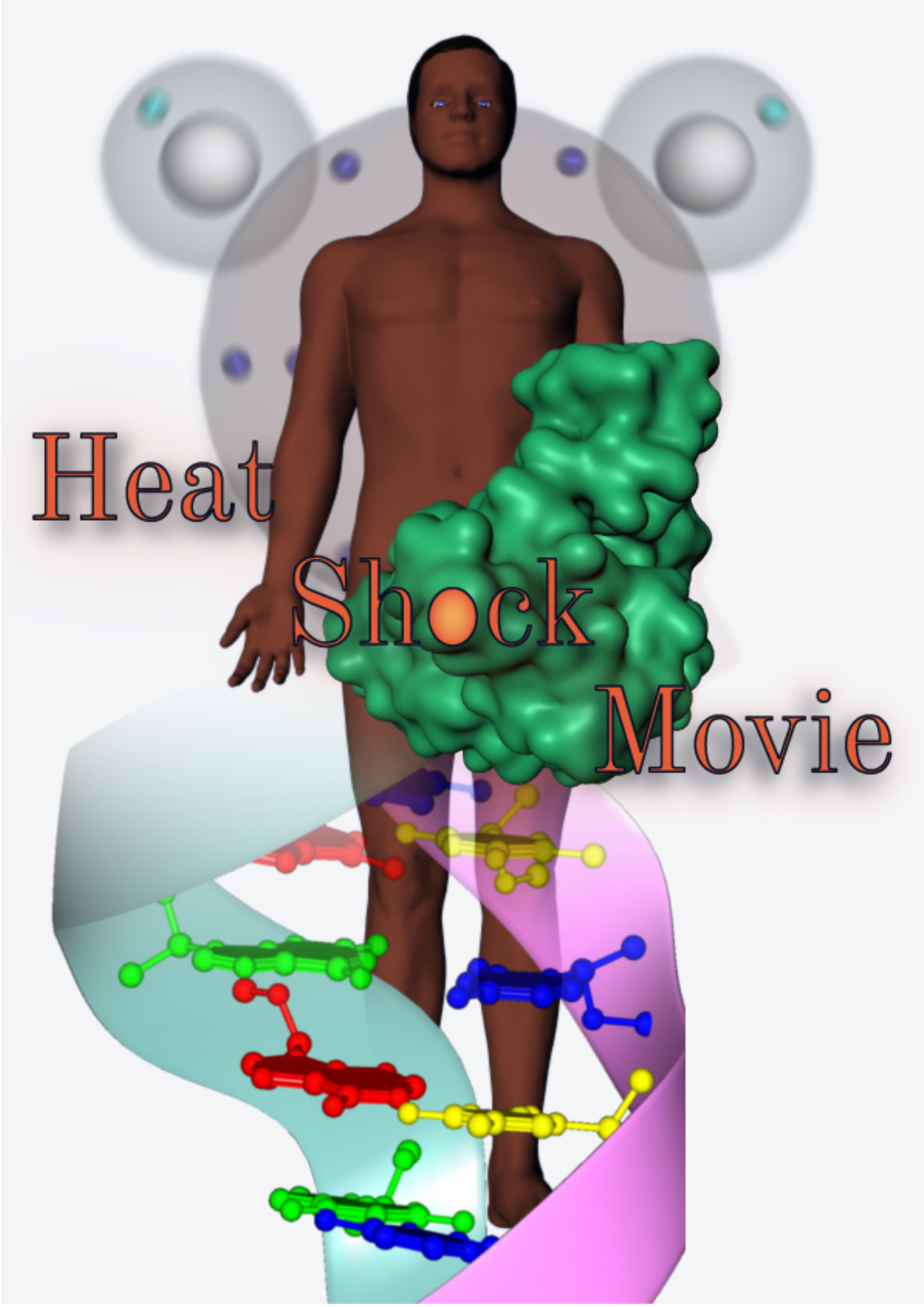

Heat
Shock
Movie

Abstract

This book describes how scientific visualization and storytelling techniques can work together for explaining a complex topic in simple and entertaining way via an immersive CGI movie. Film language has been utilized for conveying scientific data, acquired from PDB files (protein structures) and other scientific resources. After importing them into Autodesk MAYA via “Molecular Maya toolkit” (mMaya), they were animated using artistic license when appropriate for maintaining a dramatic structure. Animation workflow included traditional keyframing, as well as procedural animations (MASH networks and physically based simulations). Multidisciplinary approach allowed, after compositing rendered images within non-linear editing software (Adobe Premiere Pro) and adding an audio commentary, perfecting whole educational movie merging science, pedagogy and arts into a pure EDUTAINMENT.

Keywords

Scientific visualization, edutainment, CGI, multidisciplinary, storytelling, Autodesk MAYA, immersive experience, digital cinematography, artistic license, film language, dramatic structure, compositing, non-linear editing, animation workflow, physically based animation, protein structure, PDB, scientific data, light drawing, three-point lighting.

Bc. Martin Matulík

555Production © 2017

Matulik.Martin@gmail.com

Content

INTRODUCTION **10**

1 COMMUNICATING SCIENCE **12**

1.1 FUNDAMENTAL COMPETENCIES **14**

1.1.1 INTEGRATING ABILITIES 15

1.1.2 OPTIMIZING RESULTS 15

1.2 VISUAL LANGUAGE **16**

1.2.1 DEPTH PERCEPTION CLUES 16

1.2.2 FOCUS & SUBTEXT 17

1.3 ENHANCING REALITY **18**

1.4 DRAMATIC STRUCTURE **19**

1.5 MULTIDISCIPLINARY CAPTURING METHODS **21**

1.5.1 CINEMATOGRAPHICAL FILM MILESTONES 21

1.5.2 FAST EVENTS IMAGING 22

1.5.3 MOTION CAPTURE & VIRTUAL REALITY 24

1.6 SCIENTIFIC APPROACHES **25**

1.6.1 RECONSTRUCTING MEDICAL IMAGES 26

1.6.2 DETERMINING MOLECULAR STRUCTURES 27

1.6.3 PROTEIN DATA-BANK (PDB) 28

1.6.4 BASIC VISUALIZATION TOOLS 29

1.7 DIGITAL MODELLING **30**

1.7.1 ANIMATION SOFTWARE 30

1.7.2 IMAGE CREATION WORKFLOW 31

1.7.3 ANIMATION TECHNIQUES 31

1.8 BEST OF ALL WORLDS **32**

1.9 EXAMPLE: BLOOD FLOW ANIMATION **33**

2 AUTODESK MAYA® 37

2.1 SOFTWARE INTERFACE 38

2.1.1 OVERVIEW WINDOWS 39
2.1.2 MENUS & LINES 41
2.1.3 NAVIGATION 43
2.1.4 COMMUNICATING WITH MAYA 45

2.2 MODELLING TOOLS 46

2.2.1 NURBS VS. POLYGONS 46
2.2.2 POLYGONAL MODELLING 47
2.2.3 ADJUSTING ATTRIBUTES 48
2.2.4 COMPONENT SELECTION MODES 48
2.2.5 SOFT SELECTION 49
2.2.6 SHAPING GEOMETRY 49
2.2.7 NURBS CURVES 50
2.2.8 CURVES INTO SURFACES 51
2.2.9 TROUBLESHOOTING COMMON ISSUES 52

2.3 SHADERS & MATERIALS 53

2.3.1 LAMBERT SHADER 53
2.3.2 BLINN SHADER 54
2.3.3 ANISOTROPIC SHADER 55
2.3.4 HYPERSHADE WINDOW 56
2.3.5 CREATING PROCEDURAL TEXTURES 57
2.3.6 BUMP MAPPING VS. DISPLACEMENT MAP 58

2.4 CAMERAS 59

2.4.1 CAMERA SEQUENCER 59
2.4.2 SIMULATING REAL-WORLD CAMERA 60
2.4.3 FINALIZING CAMERA SETUP 61
2.4.4 RESOLUTION GATE 62
2.4.5 SHOT SIZES 63
2.4.6 COMPOSING SINGLE SHOT 65

2.5 LIGHTING 66

2.5.1 STANDARD LIGHT ATTRIBUTES 66
2.5.2 AVAILABLE LIGHTS 67
2.5.3 BASIC ILLUMINATION & AESTHETIC UTILIZATION 68
2.5.4 THREE-POINT LIGHTING 69

2.6 ANIMATION APPROACHES 70
2.6.1 PLAYBACK PREFERENCES 70
2.6.2 KEYFRAMES 71
2.6.3 MOTION PATH 72
2.6.4 EXPRESSIONS 73
2.6.5 PHYSICALLY BASED SIMULATIONS 74
2.6.6 APPROXIMATING REAL-WORLD BEHAVIOR 74
2.6.7 INSTANCING GEOMETRY 76
2.6.8 PROCEDURAL ANIMATION 79
2.6.9 DEVELOPING COMPLEX BEHAVIOR VIA MASH 80
2.6.10 COMPARING ANIMATING APPROACHES 83
2.7 RENDERING IMAGES 84
2.7.1 BUILT-IN RENDERERS 85
2.7.2 SELECTING SUITABLE RENDERER 86
2.7.3 SETTING COMMON RENDERING ATTRIBUTES 87
2.7.4 ADJUSTING RENDER-SPECIFIC ATTRIBUTES 88
2.7.5 TWEAKING MAYA HARDWARE [2.0] RENDERER 90
2.7.6 LAYERS VS. PASSES 91
2.7.7 RENDER SETUP WINDOW 92
2.7.8 COLLECTIONS AND OVERRIDES 93
2.7.9 RENDERING IMAGE SEQUENCE 94
CONCLUSION 95
LITERATURE 97

LIST OF FIGURES

Figure 1) Three basic visual metaphors are widely used to display molecular structure [19] .. 12
Figure 2) Comparing various approaches targeting different audiences [26] 13
Figure 3: Finalizing partial results 15
Figure 4) Great paints utilize principles of human perception [41] 17
Figure 5) Stylization of journal cover [27] 18
Figure 6) Narratology terms relationships 19
Figure 7) Modern dramatic structure 20
Figure 8) Image capturing milestones in cinematography 21
Figure 9) High-speed camera principle 22
Figure 10) Different technological solutions for capturing fast events 23
Figure 11) Timeline of medical imaging modalities 26
Figure 12) From electron microscopy to structural study [57] 27
Figure 13) PDB entries 28
Figure 14) Zika virus inside NGL viewer from various distances: 29
Figure 15) CGI creation workflow 31
Figure 16) From experimental data to visually appealing content 32
Figure 17) Selected stages of diegesis creation 34
Figure 18) Blood flow final images selection rendered via Maya's native software renderer 35
Figure 19: MAYA 2017's Graphical User Interface 38
Figure 20: Four-view Viewport layout 38
Figure 21: Tool Box 39
Figure 22: Channel Box 39
Figure 23: Attribute editor 40
Figure 24: Main menu ribbon 41
Figure 25: Menu Sets 41
Figure 26: A Shelf 41
Figure 27: Help Line at the very bottom of MAYA Interface 41
Figure 28: Animation Controls (green) and Animation Preferences (yellow) 42

Figure 29: Frame selection fits selected object to Viewport *43*
Figure 30: Quickly accessing menu commands via a Hot-Box *44*
Figure 31: NURBS shape (left) provides more curvature than Polygonal one (right) *46*
Figure 32: Creating a Polygonal sphere Primitive *47*
Figure 33: Main menu contains additional options *47*
Figure 34: Basic Primitive's attributes *48*
Figure 35: Selecting object's components *48*
Figure 36: Vertex mode *48*
Figure 37: Tool settings for organic selection *49*
Figure 38: Shaping Polygonal Sphere (left) into Red blood cell geometry (right) *49*
Figure 39: NURBS Patches *50*
Figure 40: Bezier Curve Tool provides most flexibility *50*
Figure 41: Extrude command requires selecting Profile first (white) and Path last (green) .. *51*
Figure 42: Pre-setting Extrude options *51*
Figure 43: Normals determine which side will be shaded *52*
Figure 44: Extrude tab allows tweaking some attributes after creating the Surface *52*
Figure 45: Lambert Shader attributes *53*
Figure 46: Customized Lambert-type Material *53*
Figure 47: Blinn shader provides decent Specular Shading controls *54*
Figure 48: Anisotropic Shader allows directionally customize Specular Highlights *55*
Figure 49: Customized shaders may resemble various materials *55*
Figure 50: Hypershade window for building Shading networks *56*
Figure 51: Shading network created automatically for Bump Mapping input node *57*
Figure 52: Noise1 driving material's Bump Map *57*
Figure 53: Displacement mapping is accessible from shading-Engine node *58*
Figure 54: Bump Mapping (left) vs. Displacement Map (right) *58*
Figure 55: Sequencing different cameras into a single clip *59*
Figure 56: Standard camera attributes *60*
Figure 57: Film back allows simulating real-world camera *60*

Figure 58: Device aspect ratio should match Film Aspect Ratio *60*
Figure 59: Orthographic View allows position camera precisely *61*
Figure 60: Gimbal mode prevents weird rotations *61*
Figure 61: Resolution Gate allows establishing composition accurately *62*
Figure 62: Conventional shot sizes *63*
Figure 63: Complex shot featuring unfolded protein (Long Shot) and HSP70 (Close-Up) *64*
Figure 64: Tissue-analog sphere illuminated by Default Lighting *66*
Figure 65: Virtual Lights allow overcoming physical boundaries of real-world lighting *66*
Figure 66: Available MAYA standard Lights *67*
Figure 67: Four-point Lighting setup established by only three Lights *69*
Figure 68: Viewport's Playback interface *70*
Figure 69: Playback Preferences are crucial for making timing-related decisions correctly *70*
Figure 70: Graph Editor allows graphically customize interpolation between Keys *71*
Figure 71: Attaching Red blood cell to Motion Path *72*
Figure 72: Setting Motion Path attributes to obtain expected animation results *72*
Figure 73: Expression Editor allows to establish Expression between selected attributes *73*
Figure 74: Assets should be ready before developing nParticle Simulation *74*
Figure 75: Pulsing blood requires only three Keyframes *75*
Figure 76: Essential Field's attributes *75*
Figure 77: nParticles travelling with pulsations inside a Blood vessel *75*
Figure 78: Rotation Per-Particle enables computing rotation for each particle individually *76*
Figure 79: Baking an Instancer *77*
Figure 80: Editing Expression for assigning different shapes to nParticles proportionally ... *78*
Figure 81: White blood cell flowing within Pulsating blood stream inside an artery *78*
Figure 82: MASH is most convenient tool for creating procedurally based animations *79*
Figure 83: Setting volumetric Distribution for attached geometrical object (Mesh) *80*
Figure 84: Blue cones dispersed within Tissue-analog sphere *80*
Figure 85: Signal node allows controlling random motion over time comprehensively *81*
Figure 86: Falloff Object for art-directing procedural animation manually *81*

Figure 87: Importing geometry by MMB *82*

Figure 88: Exporting MASH Network as Alembic Cache for ensuring proper rendering *82*

Figure 89: Randomly floating Chromosomes animated procedurally via MASH *83*

Figure 90: Project Management Triangle in "Pick any two" form *84*

Figure 91: Common Render settings *87*

Figure 92: Sampling determines image quality and rendering time *88*

Figure 93: Arnold allows precisely manage individual feature's quality by sampling values *89*

Figure 94: Reasonable settings for rendering images in production quality *90*

Figure 95: A Beauty Pass image of Blood artery immersive shot *91*

Figure 96: Three Layers are optimal for even complex scenes *92*

Figure 97: Render Setup Window *92*

Figure 98: Expression adds new objects automatically *93*

Figure 99: Rendered images from FG (top left), MG (top right), BG (bottom left) Layers and Final composite image from resulting Movie after NLE operations (bottom right) ..*94*

LIST OF TABLES

Table 1: Monocular and binocular clues for depth perception [53] *16*

Table 2: Key applications of selected imaging techniques *25*

Table 3: Complementarity of common molecular structure determining methods *28*

Table 4: Animation software comparison *30*

Table 5: Autodesk MAYA® hardware requirements *37*

Table 6: Quick but powerful navigation shortcuts *44*

Table 7: Comparing MAYA's essential GUIs *45*

Table 8: Selecting appropriate type of Lens for common purposes *65*

Table 9: Assorting MAYA standard Lights *67*

Table 10: Achieving various purposes by different lighting *68*

Table 11: Comparing different animating approaches *83*

Table 12: Comparing MAYA integrated renderers by PMT *85*

LIST OF SYMBOLS & ABBREVIATIONS

CGI	Computer generated imagery
ADP	Adenosine di-phosphate
ATP	Adenosine tri-phosphate
2D	Two-dimensional
3D	Three-dimensional
4D	Four-dimensional
LoD	Level of detail
fps	Frames per second
24p	24 Progressive
HD	High Definition
DVD	Digital versatile disc
GB	Giga Byte
CPU	Central processing unit
GPU	Graphical processing unit
RAM	Random access memory
CCD	Charge-coupled device
AR	Augmented reality
VR	Virtual reality
HMD	Head-mounted display
CAD	Computer aided design
CNC	Computer Numeric Control
RTG	X-ray radiography
CT	Computed tomography
PET	Positron emission tomography
SPECT	Single-photon emission computed tomography
NMR	Nuclear magnetic resonance
MRI	Magnetic resonance imaging
fMRI	Functional MRI
MRS	Magnetic resonance spectroscopy
XRC	X-ray crystallography
MPM	Multi-photon microscopy
AFM	Atomic force microscopy
EM	Electron microscopy
DNA	Deoxyribonucleic acid
tRNA	Transfer ribonucleic acid
H-bond	Hydrogen bond
PDB	Protein data-bank
wwPDB	Worldwide-PDB
RCSB	Research Collaboratory for Structural Bioinformatics
LT	Low cost
DICOM	Digital Imaging and Communications in Medicine
GUI	Graphic user interface
MEL	MAYA embedded language
NURBS	Non-uniform Rational Basis Spline
PP	Per particle
VFX	Visual effects
LUT	Look up table
PoV	Point of View
NLE	Non-linear editing
PMT	Project management triangle

INTRODUCTION

Following project seeks to guide scientific workers in biomedical field through process of creating visually appealing content from their research findings, allowing thus vivid understanding of their work by their colleagues, as well as people with limited knowledge in their area of expertise.

Various approaches for creating digital imagery are briefly discussed to get insight and overall idea of currently utilized methods, followed by showing, how these approaches may be combined together to produce astonishing results, formerly infeasible.

Even better, these techniques are available to any researcher with access to personal computer equipped with proper software, without necessity of cumbersome and painstaking obtaining necessary digital models for portraying physically accurate biological structures, as well as they function.

We are then introducing fundamental modelling and animating approaches, along with other tasks (establishing *Cameras*, *Lighting* and *Rendering* setup), whose understanding is necessary for creating visually appealing images that can be subsequently composed into animated movies.

Example files are created in Autodesk® MAYA®, which had been chosen as most comprehensive software environment for creating computer generated imagery. Substantial part of following publication is aimed at understanding of how to communicate scientific topics effectively, especially for educative and outreach audiences.

Current medical and bio-molecular research data acquisition methods, whose outcomes can be cost-effectively utilized within modelling and animating software solutions, are discussed and overall workflow, from data acquisition up to producing final images, is outlined. This allows reader to quickly acquire necessary skills for creating visually appealing content, supporting medical and biotechnology research findings.

This publication thus shall be brief, but comprehensive guide, allowing scientists to produce visually appealing content, optimized for conveying core message to targeted audience effectively, in as physically accurate manner, as possible.

Part I:
Conveying scientific knowledge via visually appealing content

1 COMMUNICATING SCIENCE

One of essential objectives, interconnecting all natural sciences, is to get better **understanding** about how world around us works. When some interesting observation of previously unknown phenomena takes place, we might be formulating hypotheses, based on our observations, in order to comprehend principles leading to such observations. As we are creating models, simplifying overwhelmingly complex nature of physical reality, decision must be made, which aspects of real world observations are crucial for understanding principles driving them, as well as excluding those diverting attention from the core ideas. These models should allow us to not only effectively understand some basic principles, on which world around us works, but also help other people understand quintessence of our discoveries.

Substantial part of this everlasting process consists of **conveying** our current understanding to other people – our colleagues, peers in scientific community and also folks with limited knowledge in particular scientific field, or natural sciences in general. Therefore, we use different representations for same things, each suitable to convey particular aspect of interest most clearly.

a) Bond diagram (balls & sticks) **b) Space-filling diagram** **c) Ribbon diagram**

Atomic bonding *Molecular interactions* *Structural research*

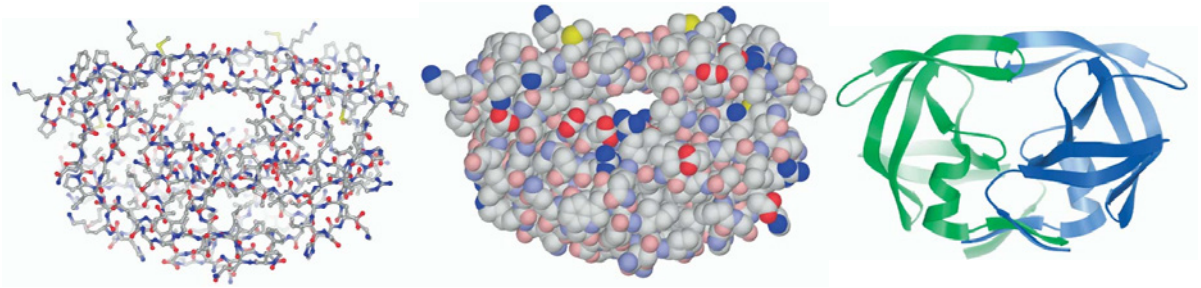


Figure 1) Three basic visual metaphors are widely used to display molecular structure [19]

The second crucial thing to consider is **targeted audience,**
for which our message is primarily intended:

Expert researchers in molecular biology will probably be interested in accurate and detailed descriptions, allowing them to e.g. replicate our experiment, so they can confirm its results in their laboratory, or utilize acquired knowledge to advance their own research.

High-school students will likely appreciate more simplified, conceptualized models, facilitating their growing knowledge by clear representation of core idea, meanwhile communicating important scientific breakthrough to the *world outside* scientific community (outreach audience) effectively would require catching, attractive visuals, conveying the message via convenient metaphor, or appealing story.

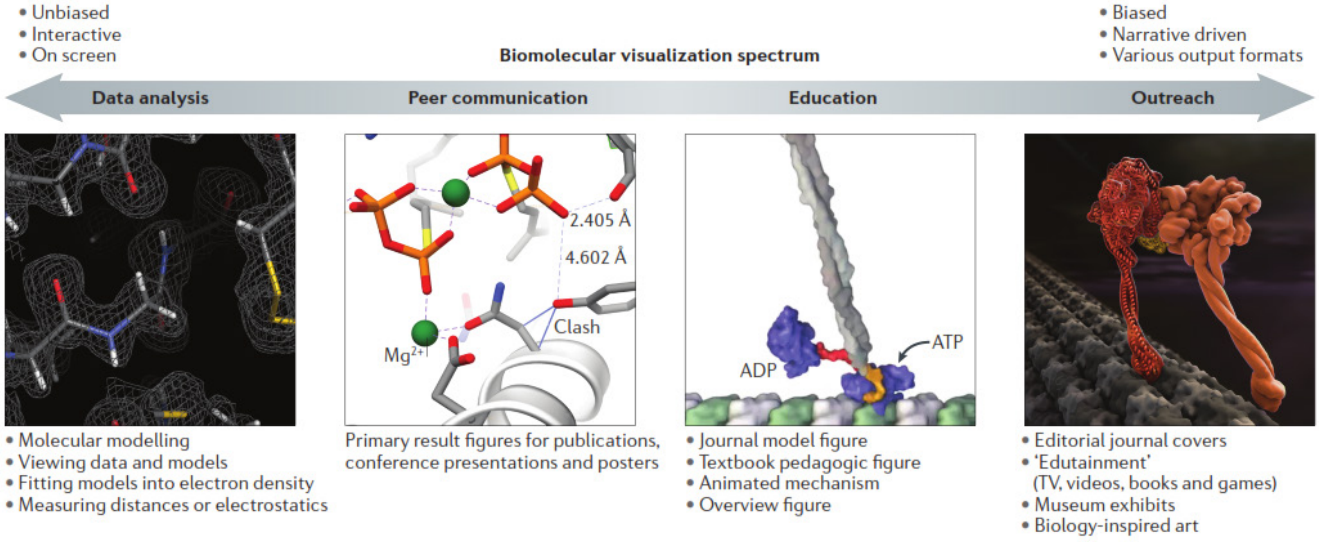


Figure 2) *Comparing various approaches targeting different audiences* [26]

Defining our targeted audience, allows us to competently **determine appropriate visual representations** and subsequently select most suitable tools for creating them, e.g. visualizing motor protein via a *molecular viewer* (Chimera), exporting picture in *data format* convenient for following work (.png), composing it with other visual elements (ADP, ATP and hydrogen atom visualizations, along with arrows depicting chronological chain of their interaction) into *single image*, which is then placed in power-point based *presentation slide* with supplemental materials (text boxes, geometrical shapes) and appropriately formatted (duplicating slides, fading different image regions for step-by-step process explanation).

Creating content conveniently for targeted audience facilitates understanding of core ideas, ensuring its effective communication and probability of reusing work by other people (researches, educators, media), giving an opportunity to increase discussed topic awareness, promote our research, or monetize its results in a short time effectively.

1.1 Fundamental competencies

Communicating complex scientific topics to less educated (students) and even outreach audiences (world outside scientific community) is a great challenge, as for doing it properly, a single person has to acquire competencies in three distinct areas, requiring completely different way of thinking and often demanding contradictory approaches (scientific precision vs. artistic licence[1]):

Science – Any idea we, as scientists, want to share with other people, needs to be based on solid data, rigorously acquired from findings of reliable scientific studies. To design and execute such studies, or at least comprehend results from studies performed by others, is necessary to have profound understanding of particular area we would like to talk about (i.e. molecular biology). At utmost importance is not only understanding new discoveries and their impact on current state of scientific knowledge, but also our ability to evaluate credibility of these discoveries. Furthermore, we need to realize, why exactly this particular discovery is so important, that we intend to devote our time and personal effort for conveying this idea in a way understandable to wider audience.

Pedagogy – It is said, that most difficult part of being teacher isn't teaching itself, but rather deciding what NOT to teach. Likewise, when explaining scientific knowledge (sometimes accompanied by abstract concepts), we need to carefully select ideas, most suitable for conveying the core ideas, while excluding others, which are not essential for understanding the topic by targeted audience. Selected ideas then need to be processed and cultivated into a comprehensible form, ingestible by average attendant. Pedagogical education is recommended, although understanding fundamental didactic principles can provide us essential tools for doing so (i.e. *organizing* know-how into *logical units*, proceeding *from simple to complex*, give *example* → deduce *law* → outline *practice*).[2]

Arts – Resulting visual style can dramatically improve understanding to particular idea. Appropriate design and composition can drive viewer's attention towards main area of interest, meanwhile allowing supplemental material subtly provide context, reinforcing thus viewer's overall idea and allowing integration with previously gained knowledge. Furthermore, imagery inciting emotional responses can strengthen neural connections responsible for memorizing acquired information, while stronger, and especially pleasurable, emotional responses lead to better and longer lasting memorization.[37] Appealing visual style greatly enhances attractivity of whole product to end user and can be the difference between rejection and adoption of conveyed ideas, as well as product itself.

[1]Fact distortion in service of a purpose [22]

1.1.1 Integrating abilities

Ordinary person may usually have profound understanding and necessary qualification in one of above-mentioned areas, as they all require extensive study, along with experiences, that can be gained only by utilizing acquired knowledge in real-world situations. Therefore, it is rather rare for single person to have necessary qualities in all areas and integrating them into mutually complementing unit, allowing him to skillfully utilize most suitable aspects of those areas for achieving primary purpose effectively. At the same time, it is essential to realize, that **these three pillars are absolutely necessary** for comprehensibly conveying complex scientific topics to students and outreach audience.

Although science and arts usually require contradicting demands, carefully assigning appropriate weight to each aspect in certain situation may lead towards finding most suitable solution for conveying particular idea (artistic license used for completing unknown parts of cellular machinery with generalized form of probable structure). Pedagogical competencies there may serve as **bridging element**, interconnecting both worlds and provide clear answer, when deciding, which aspect should be given priority in a particular situation.

1.1.2 Optimizing results

For defining next steps, in order of achieving work's primary purpose, **asking appropriate questions** can lead us to suitable solutions (*will be the product comprehensible without approximating unknown areas? Will that amount of artistic license elucidate incorrect visual representations? What else needs to be done in order to properly convey the core idea?*). These questions must be answered before finalizing work's individual segments (chapters in book, slides in presentation, scenes in movies).

Actions taken subsequently should be aimed for correcting all serious mistakes, as well as improper idea displaying solutions (unsuitable for conveying particular message), along with adding supplemental content, throwing out distractive elements, and enhancing overall appearance.

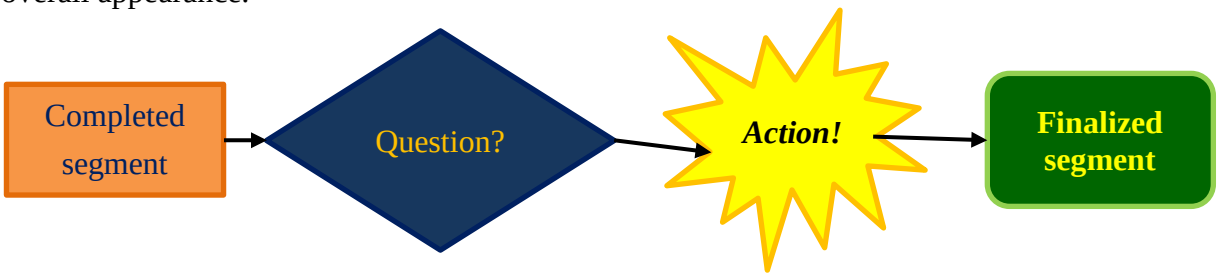


Figure 3: Finalizing partial results

By evaluating partial results of our work this way, we are efficiently improving their comprehensibility and attractivity and consequently achieving our primary goals more accurately, by utilizing available resources in cost-effective manner.

1.2 Visual language

Throughout the human history, knowledge has been passed on by various, increasingly sophisticated means. From formulating and vocalizing our ideas, putting them on paper via handwriting, to creating **illustrations**, understandable by anyone, without necessity of intentionally learning syntax and spelling rules any human-made language requires. The **visual language** is universal, although for using it effectively to convey a particular message, it is necessary to learn its grammar, which is based on principles of human perception. Studying **cognitive psychology** findings allows us being aware of them and purposefully utilize those principles, when communicating via visual forms (e.g. exaggerating light source intensity within scene by realizing Weber-Fechner's law, telling us that *increasing stimulus intensity appears to be lower subjectively, than its corresponding increase expressed in physical or chemical units*[49, 15]).

1.2.1 Depth perception clues

Composition decisions should respect those principles and utilize them for conveying our message most effectively. Since our brain is observing 2D image representing 3D environment (scene), it needs some reliable clues to correctly determine depth and figure out spatial relations among objects. As we are consciously arranging visual elements in a scene, monocular and binocular clues can help us establish these relations naturally.

Table 1: Monocular and binocular clues for depth perception [53]

Clue ***(monocular)***	**Object is near**	**Object is far away**
Surface gradient	↑ granularity at ↓ distances	↓ granularity at ↑ distances
Relative size	Bigger	Smaller
Occlusion (interposition)	Overlapping another object	Being overlapped by another object
Linear perspective	Parallel lines diverge towards observer	Parallel lines converge towards horizon
Aerial perspective	Objects appear clearer, strong outlines	Objects appear vague, weak outlines
Horizontal plane	Obj. under horizon placed lower, obj. above horizon placed higher	Obj. under horizon placed higher, obj. above horizon placed lower
Motion parallax	Obj. moving towards observer grows quicker	Obj. moving away shrinks quicker
(binocular)		
Convergence	Eyeballs turn to nose tip	Each eye turns to ear on its side
Disparity (stereopsis)	↑ difference in same image observed by each of both eyes	↓ difference in same image observed by each of both eyes

Figure 4) Great paints utilize principles of human perception [41]

1.2.2 Focus & Subtext

When looking at some particular object within our field of view, we tend to put in sharpest vision center, so that light rays, reflected from the object, converge right into macula lutea – pigmented area inside our eyes with largest concentration of cone cells. As a result, other objects (especially at significantly different distances) appears to be blurred. This phenomenon is simulated by **depth of field**, usually referring to, in case of optical devices, the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. [13] We can achieve similar results artificially, by selectively blurring image areas based on its object's spatial relationships.

People perceive surrounding reality not only from sensoric information, but evaluate them with higher cognitive processes, utilizing previous experiences. Using **associations,** when composing scene, therefore represent powerful tool for enlightening unobvious facts and revealing deeper meanings (*by depicting ATP-Synthase model on energy bar wrapping, viewer instantly understands, that it has something to do with energy replenishing, without any knowledge of molecular machinery and even without explanatory text!*).

Substituting generally unknown genuine objects with artificially added ones, well known to viewers from their everyday life, can clearly convey their actual purpose, function, or other desired aspect. **Metaphors** thus represent another powerful tool, useful especially during initial learning phases (topic introduction), as they may facilitate memorizing essential representations long-lastingly (subsequent physically accurate models will be associated with this emotionally-loaded core memory, so they may be recalled more easily from long-term memory). Other frequent application takes place, when communicating complex ideas to outreach audience in extremely short time effectively is required (commercials), albeit as merely rough approximation of factual state of affairs.

1.3 Enhancing reality

First step, when creating visuals, is **depicting subject in as physically accurate manner**, as possible (creating models from data obtained by imaging techniques and approximating currently unknown structures, based on current scientific knowledge), often in context with other relevant entities, facilitating better understanding of subject's purpose. This may be achieved, in case of biological structures, by techniques discussed in following chapters.

Second step is based on **enhancing selected image properties** in a way, which improves conveying core messages to a viewer. Focusing viewer's attention towards main subject may be achieved by putting supplementary structures out of focus (blurring), reducing their contrast, saturation and level of detail (LoD). Reciprocally, we can also sharpen main subject, increase its contrast in brightness, introduce complementary colors [63], boost saturation, and, if possible, reasonably maximize LoD in parts crucial for understanding core idea.

> Well-premediated and carefully implemented stylization is able to convey main message clearly, while preserving necessary contextual information:

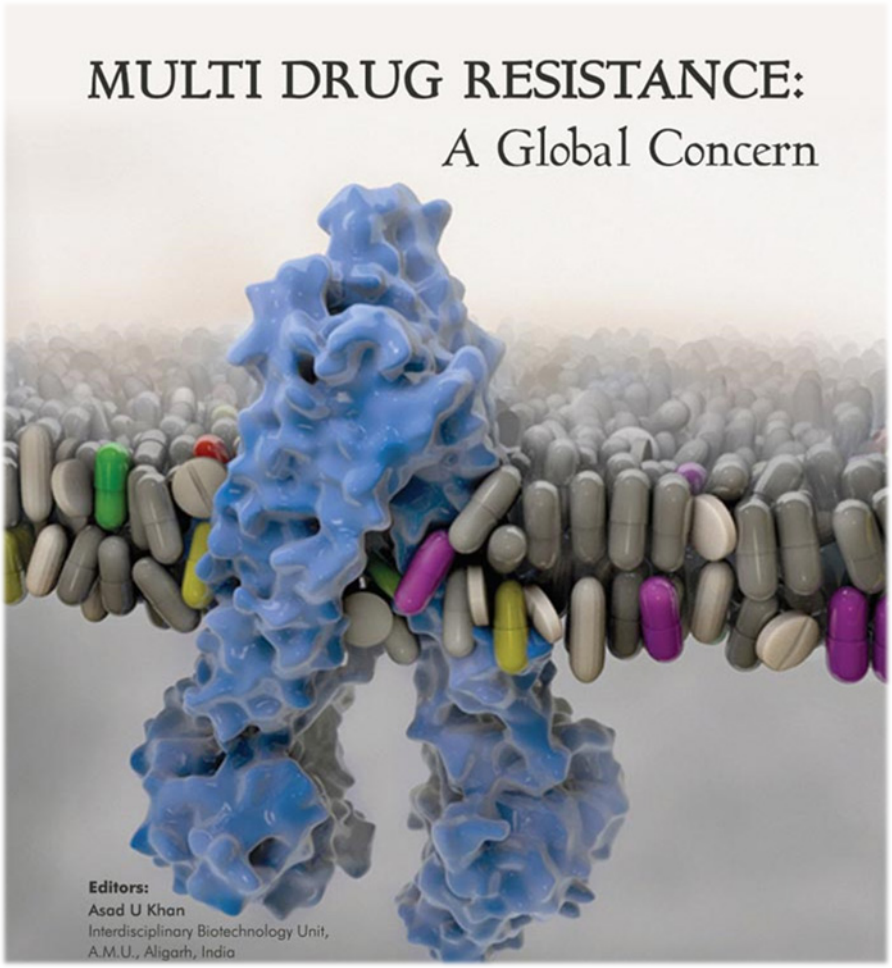


Figure 5) Stylization of journal cover [27]

1.4 Dramatic structure

Core ideas often require broader discussion, establishing context and clarifying individual steps, leading to final state. Some ideas may be clearly explained within a single step, but when intended for an outreach audience, attractive form and catching element needs to be added to capture viewers' attention and possibly pave the way for acceptance of the idea (aka infusing vitamins into lovely chocolate bar, or designing child's vitamin pills to look and taste like a candy).

Meanwhile **narration** merely presents series of events, **story** can serve as supreme vehicle, delivering the message naturally in digestible form. People are accustomed to accept ideas presented in this way, as it has been utilized for sharing them throughout the whole history (painting on cave's wall – telling stories – writing manuscripts). **Storytelling** is therefore essential skill for anyone, who strives for conveying important ideas effectively in a form acceptable by recipients.

When telling a story, either verbally, or via other medium (book, picture, film), we are establishing artificial world (*diegesis*), with its own rules, where actual action is taking place. Chronological collection of raw diegetic story material (*fabula*) is organized by *syuzhet*, including also other, non-diegetic elements, which are not part of established world (e.g. background music). **Syuzhet** may be therefore considered as total sum of all elements presented to a viewer [7].

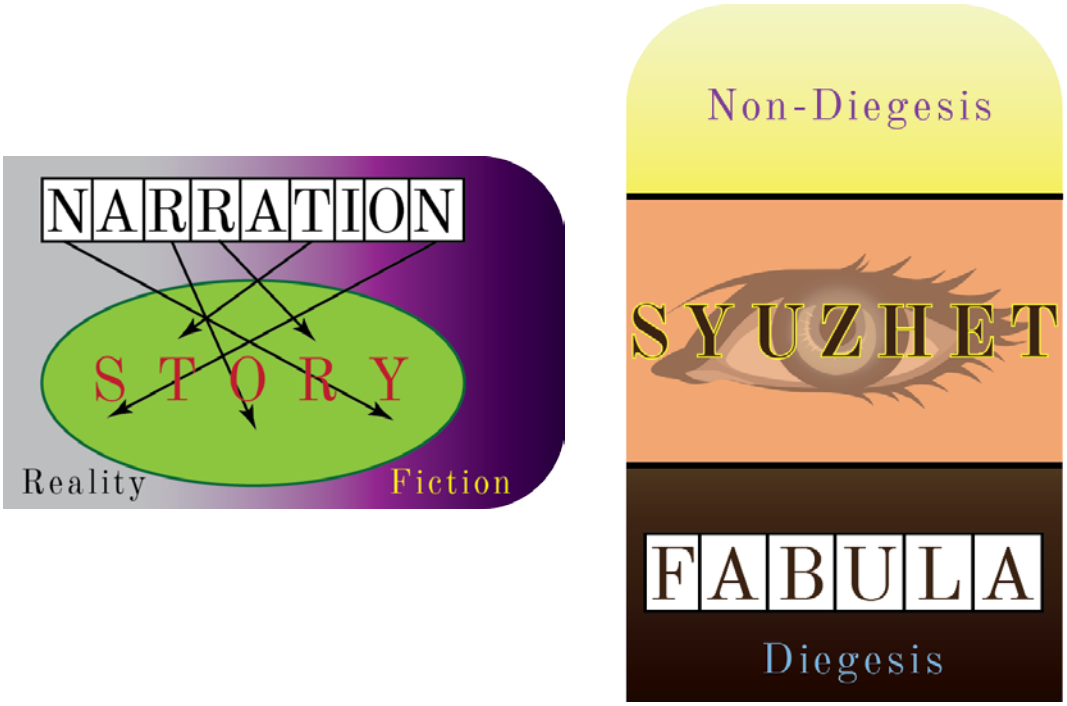


Figure 6) Narratology terms relationships

All good stories consist of four essential parts: *Introduction*, *main body*, *climax*, *closure:*

Introduction (exposition) should *introduce* viewer into the story (establish world, characters, narrative style). Occasionally, this step is skipped, throwing thus confused viewer directly into unknown water.

Main body (rising action) develops plot, shows important aspects of character's personality and slowly builds tension. May include falling action segments (releasing tension) to increase relative height of climax.

Climax draws together consequences of all previous events, confronting them, creating thus turning point of whole dramatic unit. It may unveil formerly hidden motivations and personal traits of main characters, reveal their true nature and provide answers to viewer's questions, emerging during main section.

Closure (resolution & denouement) shows consequences and outcomes of such confrontation, allowing viewer to discover main message and comprehend it in context to real-world situation. It can give him peace of mind, serving as emotional reward, or facilitate thinking process by raising another question, directly related to viewer's life.

This pattern can be utilized at multiple scales [43], e.g. when designing individual dramatic units co-creating whole work. After introducing viewer into diegesis (story's world we have created), there is usually some *inciting incident*, causing chain of events leading to climax - moment with maximum tension, which is consequently released quite quickly, when action is falling down to the *resolution* point, where main character definitely wins/loses. During closure (*denouement)*, emotion settles, status quo is established, but main protagonists are developed (changed by their experiences gathered during plot's events) – as dramatic work ends, time for viewer's inner contemplation follows.

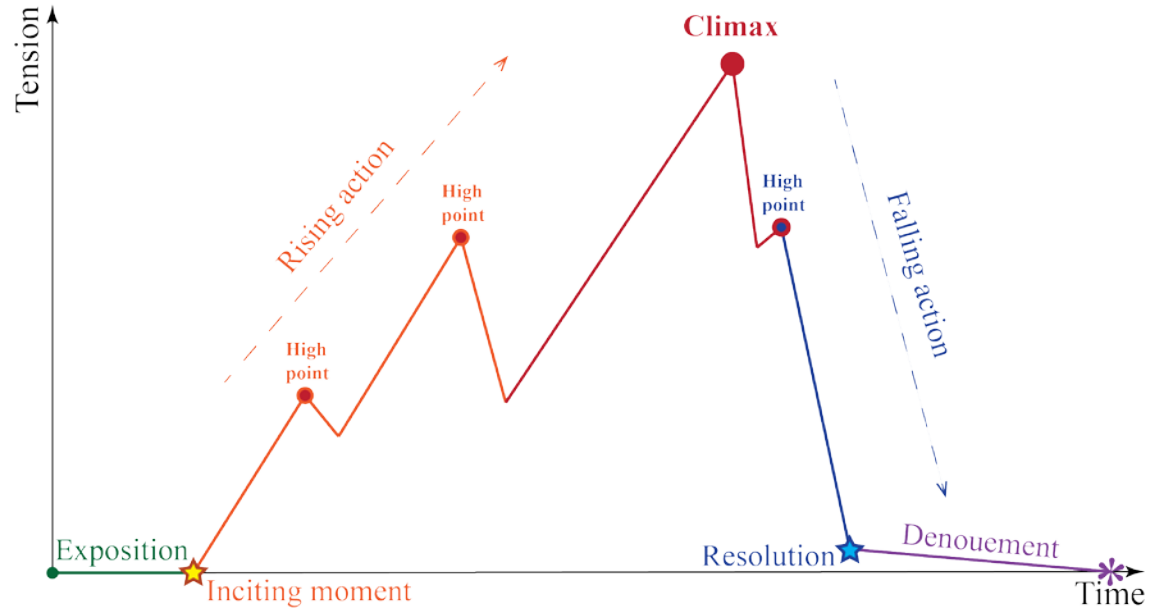


Figure 7) Modern dramatic structure

1.5 Multidisciplinary capturing methods

During human history, observing and consequently capturing reality was achieved by various means based on **optics.** Even now, when using other than optical phenomena (ultra-sound sonography) for obtaining necessary data, harvested information are usually presented in visual form and consequently processed by irreplaceable optical system – **the eyes.**

1.5.1 Cinematographical film milestones

Presenting real-world 3D objects and events as 2D images originally meant **painting** them, so drew items were an approximate illustration of real ones, although very good one, when skillfully crafted, taking into account real object dimensions, as well as human perceptual peculiarities (e.g. monocular clues). Capturing their state in given moment of time directly have been utilized since 1840's by using visible-spectrum light in daguerreotype **photographic** process (first efficient, commercially available way of processing photographs) [Peres]. Portraying reality thus have been achieved by means of light, in comparison to brush with physical colors, therefore photographic craft is sometimes referred to as *light drawing art* [1].

At the end of 19th century emerged first instruments capable to produce multiple images during sufficiently short time period for giving a continual movement illusion (more than 12 images per second, which human eyes register as individual images). Finally, these **motion picture** cameras provided grayscale images at standardized rate of 24 frames per second (fps), stabilized during 1927-1930, as it was slowest frame rate possible for producing intelligible sound (lowering production costs) [44]. During 1930s, true 3-color chemical processing have been available (3-strip Technicolor), even though extremely costly until 1950s (Eastman color negative), albeit hand-tinted motion pictures existed since 1895[8].

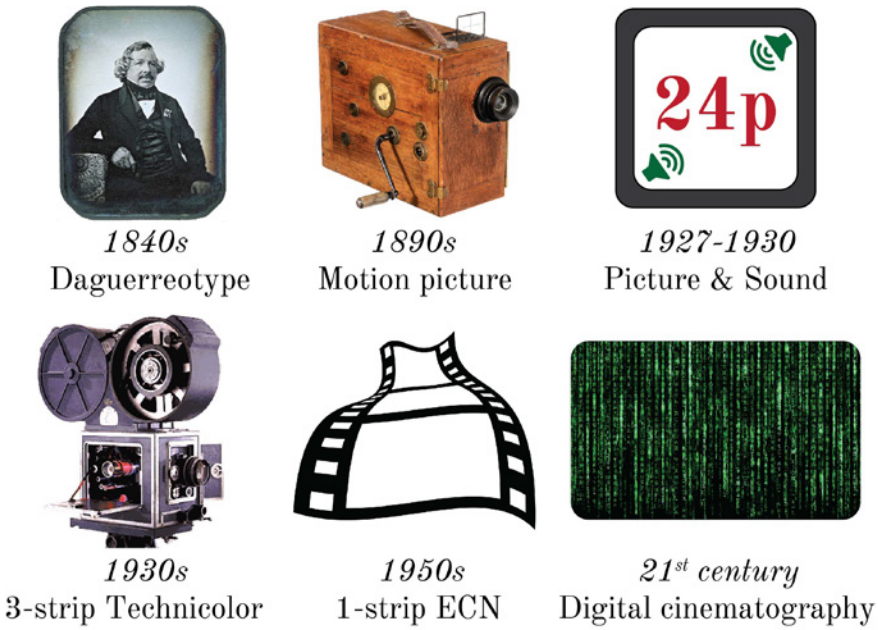


Figure 8) Image capturing milestones in cinematography

Digital CCD-based high-definition cameras gained considerable attention at the end of 20th century and currently most movies are produced either entirely digitally, or via *digital* intermediate process, as digital projections hold major market share [38] (traditional celluloid film projection share is rather marginal).

All these techniques allowed capturing reality as we might see with our bare eyes, but observing phenomena at other scales (e.g. μ-meter sized objects during μ-second events) may require specially constructed optical devices, or utilizing **other modalities**, than light medium in visible-spectrum range (x-rays, ultra-sound, electron beam), while obtaining other ones would be practically infeasible (after obtaining human tomographic slices by photographing them directly, there would be no human anymore, just bunch of slices…).

1.5.2 Fast events imaging

Observing fast happening phenomena may be achieved via **slow motion** techniques. Since standard film shooting process and subsequent projection uses 24 still frames per second (fps), fast action happening in single second would lack time-related details, or whole phenomena may be completely missing (bullet coming out of a gun). Proper way to solve this issue is increasing number of pictures taken during same time period (sampling rate) sufficiently.

Commercially available **high-speed cameras** (such as Vision Research *phantom* products) can continually capture images, providing up to 25 000 frames per second, although specialized research devices can achieve much higher sample rates (25 Mfps with Brandaris-128[18] and even 0.5 Tfps with streak camera, allowing analysis of light propagation via a femto-photography process [56]).

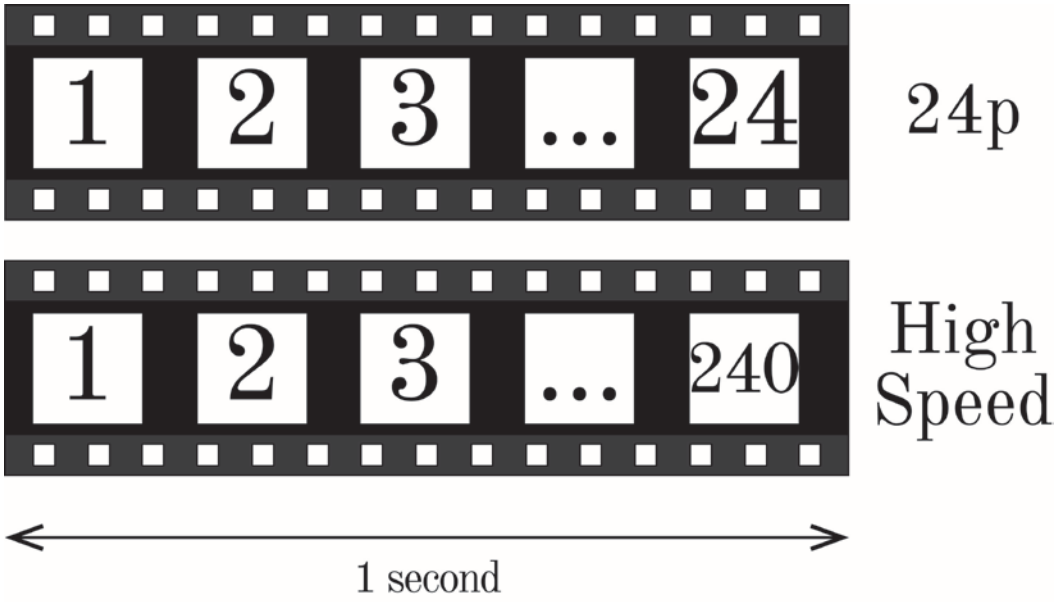


Figure 9) High-speed camera principle

All images, created within single second (e.g. 240 frames), subsequently played at standard speed (24 fps) evoke illusion of entire action happening slowly during much longer time (10 seconds). In this case, all images reflect real scene in given moments; therefore, we may reliably display and confidentially analyze phenomena taking place.

This approach allows us to capture fast-happening phenomena from single point of view (camera standpoint), but when we need images from different standpoints, electronically synchronized multi-camera rig, where each camera is triggered after precisely defined delay with respect to previous one, may be convenient. This way, motion picture appears to have detached space-time continuum, allowing virtual camera moving around the scene at a normal speed, while actual events are slowed down, effect commonly referred to as **bullet-time** [29], but still, physical accuracy is retained, as each frame reflects real scene in given moment (although from different viewpoint).

When lacking equipment capable of capturing sufficient number of pictures during given time period, slow motion effect may be, in certain cases, achieved by **interpolating images** in-between physical ones via **optical flow** algorithms, figuring out apparent speed of moving objects and consequently estimating their position in subsequent moments. Although giving an impression of slow motion, physical accuracy is no longer retained, as multiple images are computationally approximated. This technique is convenient for "slowing down" object on solid background, as in detail-rich environments, simple algorithms produce substantial distortion and serious artifacts [31].

Figure 10) Different technological solutions for capturing fast events

1.5.3 Motion Capture & Virtual reality

For digital analysis and study of animal/human locomotion, **motion capture** technology is invaluable tool. Typically, multiple markers (e.g. infrared) are attached to a person directly, or mounted on special suit. Signal received (reflected) from those markers is collected over time and consequently processed with algorithms calculating body part's movement. This technique proved useful in gait analysis, military combat training simulations and digital character movement matching [12]. Although digitally modeled character's movement may be animated, imitating fine nuances of real-world movement (secondary motions, weight and physical forces exchange) is rather complicated, time-consuming matter, therefore digital character's key points (joints) can be assigned to live action performer's ones, so digital character can perform long-lasting, complex moves, which would be quite expensive to animate, or even infeasible (physically accurate facial expressions of specific person, whose face was digitally modelled multiple times, reflecting various age periods[54]).

Providing supplemental information about our surrounding reality, or augmenting their characteristic features, is currently being accomplished by **augmented reality**[17] (AR) gadgets (eyeglass, head-up display, or contact lenses), e.g. superimposing MRI data directly over patient's body[11, 14]. Facilitating immersive, interactive experience within artificially created environments, on the other hand, can be accomplished via **virtual reality** (VR) devices, utilizing multimodal technologies like stereo-vision (3D) capable head-mounted display (HMD), quadrophonic earphones, and haptic gloves, combined into a single system [55]. In contrast to *hypothetical* **simulated reality** experience, which could be achieved by putting users into sensory deprivation state, meanwhile providing carefully designed stimuli, allowing experience artificial world indistinguishably from the real one, users of VR devices are completely aware, that they are experiencing artificial reality, although very convincing.

Most effort is currently put into developing medical skill's **learning simulators**, significantly improving knowledge, skills and behaviors, positively affecting patient's outcome [48, 23]. **Military** applications exhibit impressive adaptive abilities, customizing trainee's experience and optimizing its development [42]. Other **pedagogical** areas, such as improving driver's motor and reaction abilities, evolves at slower pace. **Medical** application include psychotherapy (one of primary treatments for post-traumatic stress disorder [50, 46]), where it has already proven very effective [45] and rehabilitation, as well as tele-medicine applications (remotely controlled robotic arms in robot-assisted surgery [58]). **Entertainment industry** experiences booming in use of this technology for digital content delivery (interactive movies, video games, advanced scenography concerts), although it penetrates into other commercial areas as well (CAD engineering, architectural and urban design, retail product view).

1.6 Scientific approaches

Imaging phenomena, unobservable by human eye with the aid of optical instruments, can be accomplished by various sophisticated solutions, developed for particular scientific fields.

Clinical imaging techniques allow digital reconstruction of anatomical compartments (CT/MRI), along with biochemical changes not only at organ level, but currently also on molecular scale (Magnetic resonance spectroscopy, PET, SPECT). Other, more frequently employed, imaging modalities offer sufficient diagnostic data for a reasonable price in common clinical practice (ultra-sound, infrared, X-ray radiography).

Scientific research imaging approaches allow investigation of molecular structures, as well as their visualization on different scales (various microscopy techniques). Most of these techniques are *destructive* to a given specimen, as either require sample preparation in way excluding their viability (X-ray crystallography), or may harm it during data acquisition phase (mechanical damage in atomic-force microscopy). [28]

Table 2: Key applications of selected imaging techniques

Imaging technique	Abbrev.	Key application
X-ray radiography	RTG	Bone fracture
Magnetic resonance imaging	MRI	Anatomy
Functional magnetic resonance imaging	fMRI	Organ activity
Magnetic resonance spectroscopy	MRS	Detecting metabolites
X-ray crystallography	XRC	Protein atomic structure
Multi-photon microscopy	MPM	Visualizing cell structures
Atomic force microscopy	AFM	Mapping cell surface
Electron microscopy	EM	Discerning proteins

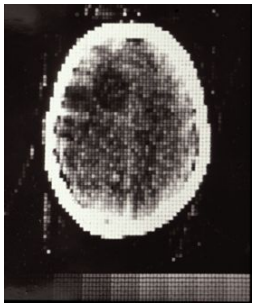
1.6.1 Reconstructing medical images

Obtaining physiologically relevant images have come through great revolution during past few decades. At the beginning of 20th century, bone anatomy could had been observed via simple **X-ray radiography** (RTG**)** images. In 1970, first **computed tomography** (CT) system was used for displaying brain tissue, layer by layer. These systems allowed depiction of bones into great detail, along with making clear distinction between bones and soft tissues (lungs), but differentiating soft tissues one from another was almost impossible. Therefore, completely different systems emerged during 1980s to fill need for detailed soft tissue images.

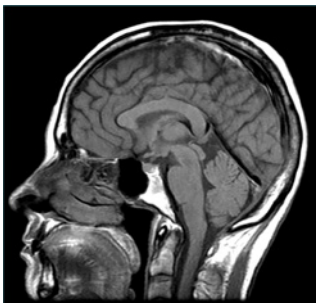
Nuclear magnetic resonance (commonly referred to as magnetic resonance imaging – MRI) was able to display clearly distinguishable soft tissues and even pathological changes in their morphology. Although those systems could display anatomy accurately (currently also compute precise 3D models with changes over time in some cases – e.g. heart cycle), along with physiology of whole organs in past two decades, physicians were still craving for devices unveiling chemical changes on a molecular level, allowing them to noninvasively perform biochemical studies in vivo.

While **Single-photon emission computed tomography** (SPECT) and **Positron emission tomography** (PET) devices were evolving since 1960s, merging them with CT devices at the end of 21st century and MRI few years later (first commercially available whole-body PET-MRI system emerged in 2011[10]) into **hybrid systems** elevated possibilities and diagnostic precision of medical imaging techniques to a whole new level, being so far the most superior diagnostic tools ever made [3].

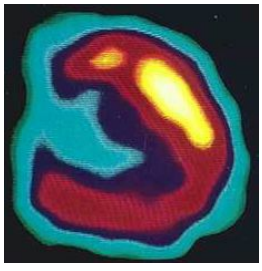
1895 (RTG)

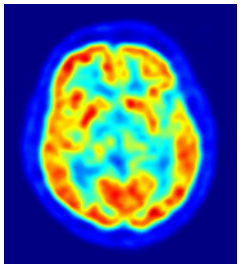


1970 (CT)

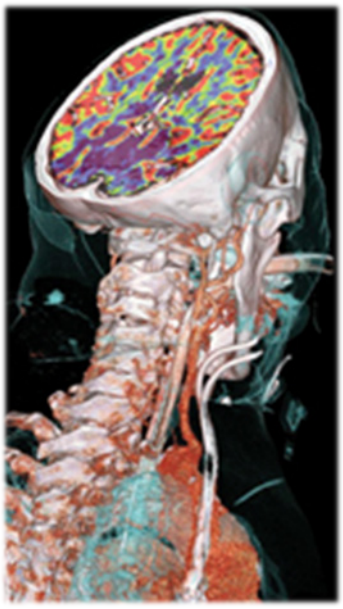


1980 (MRI)





1960 (SPECT & PET)



2001 (Hybrid systems) [9]

Figure 11) Timeline of medical imaging modalities

1.6.2 Determining molecular structures

For better understanding of physiological changes within human body, we need to study various biologically active structures, especially proteins, on a molecular level. Determining atomic structure of such entities is crucial for comprehending their role within an organism and predicting possible behavior under various conditions (e.g. conformational changes). Several methods are currently used in order to get experimental data with proper spatio-structural relations.

X-ray crystallography is probably most employed technique for obtaining protein structures, as it provides very detailed atomic information, but can be successfully applied only to rigid proteins, that form well-ordered crystals. Protein is usually purified, crystalized and beamed with X-rays, creating diffraction pattern, from which can be determined electron densities within given protein (electron density map) and consequently locations of individual atoms. [60]

NMR Spectroscopy analyzes frequency spectra obtained via putting purified protein into strong magnetic field and probing it with radio-frequency signal. Large proteins exhibit overlapping peaks in NMR spectra, therefore this technique is usually employed, when studying small and medium sizes proteins. Since specimens within solution provide better results, NMR spectroscopy is favorable for determining flexible proteins structure. [16]

Electron microscopy proves particularly useful, when examining large macromolecular complexes [30]. In case of cryo-electron microscopy, electron beam passes through a sample towards camera sensor. Acquired data are digitally processed to build a 3D model and consequently render high-resolution image. Although usually unable reveal atomic-level data, electron microscopy provides very good structural data of larger scale assemblies and macromolecule's overall shape. By combining with data from methods discussed above, we can create multi-modal images of multi-molecular structures (tRNA, protein factors).

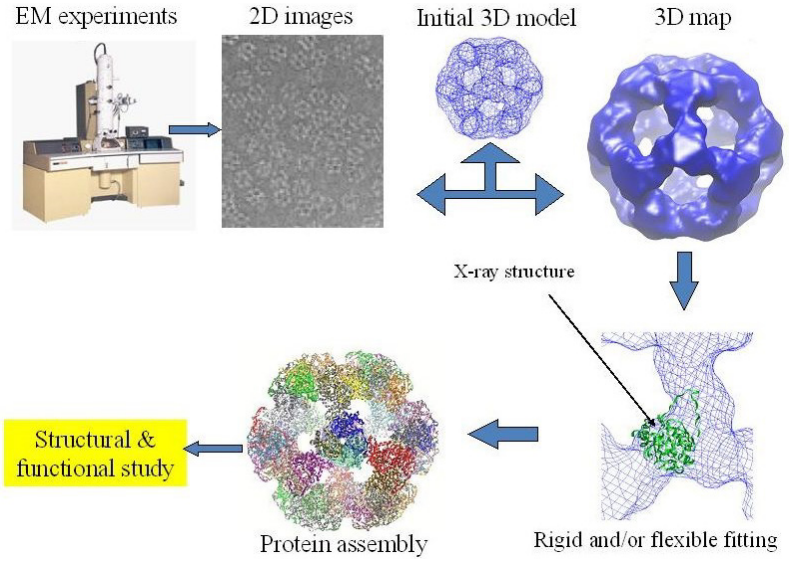


Figure 12) From electron microscopy to structural study [57]

Table 3: Complementarity of common molecular structure determining methods

	X-ray crystallography	NMR Spectroscopy	Electron microscopy
Suitable entities	Rigid proteins	Flexible proteins	Macro-molecules
Level of detail	Atoms	Atoms	Structures
Signal type	Electron diffraction	Spectral peaks	Electron diffraction

1.6.3 Protein data-bank (PDB)

After putting considerable effort and money in obtaining structural data of individual samples by all those techniques, it would be wise to store acquired information in a further utilizable form, allowing quantitative analysis and visualization of these structures via different representations (bond/space-filling diagram).

PDB data format is popular and widespread coordinate file type, providing data uniformity for biological macromolecules structure obtained by various molecular structure determining methods. *Header* section summarizes protein, citation and supplemental information, followed by *atoms sequence*, accompanied with their spatial coordinates (XYZ). For building a full-blown atomic model, we need not only sufficient experimental data, but also additional information about molecular structure of studied assemblies (e.g. preferred geometry of atoms in a typical protein) [5].

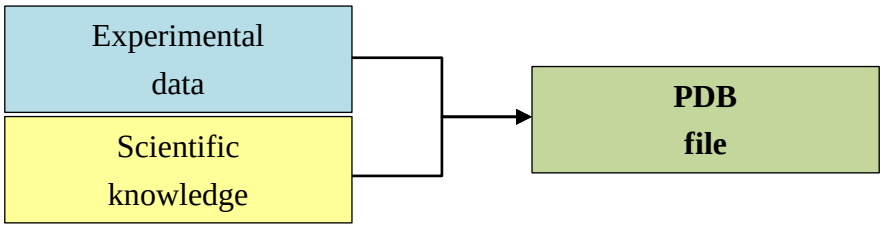


Figure 13) PDB entries

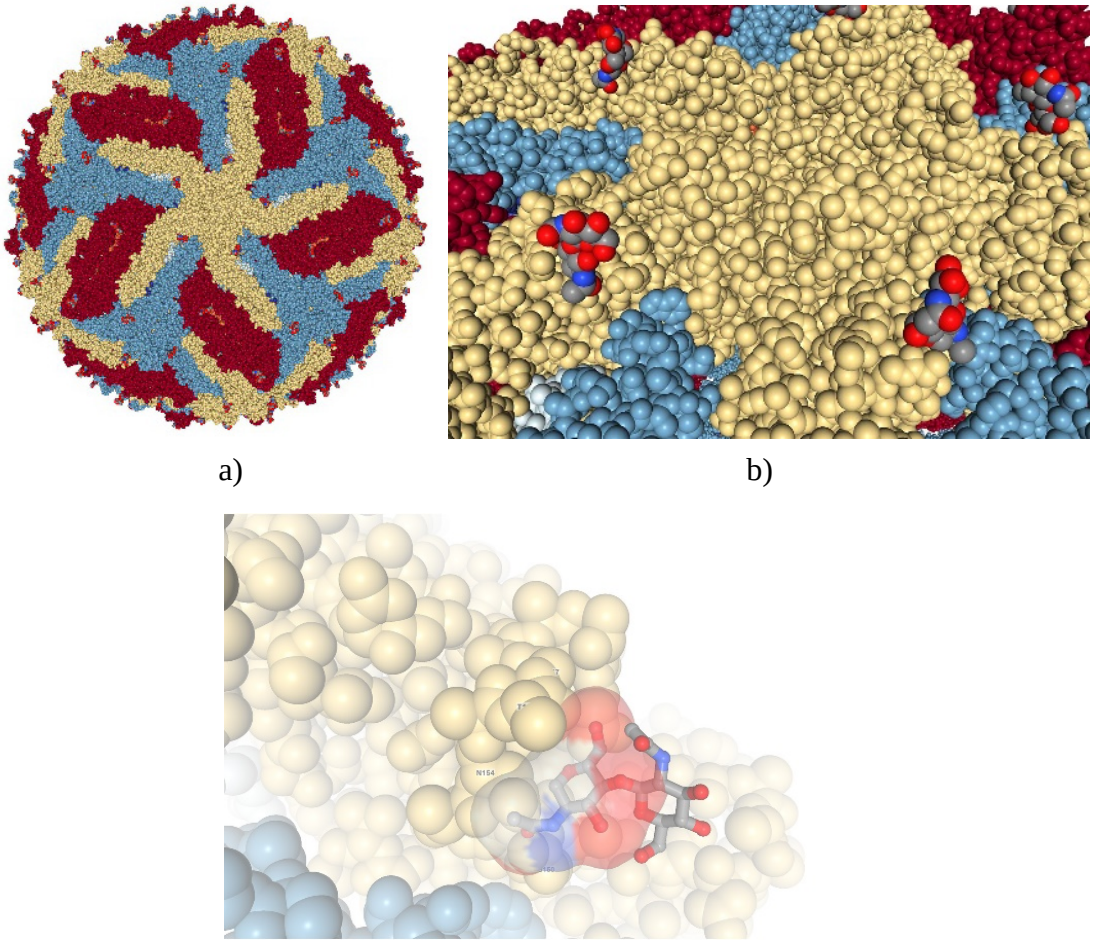
For storing all the data from laboratories focused on structural research, ensuring data validity and uniformity, along with their consequent distribution and accessibility, **worldwide Protein Data Bank** (wwPDB) was established in 1971 as single repository of information about the 3D structures of proteins, nucleic acids, and complex assemblies. This organization manages the PDB archive and ensures that the PDB is freely and publicly available to the global community. As of January 1, 2016, entire wwPDB server contains more than 1,400,000 files. [62]

Research Collaboratory for Structural Bioinformatics (RCSB) site, governing PDB informational portal, seems to be immense source of structural research data, educational materials, as well as data visualization tools.

1.6.4 Basic visualization tools

Even though all the data are currently freely accessible, we still need appropriate tools for processing them, so that they can serve our purpose properly. Visualization tools allows us to interactively inspect various macromolecular assemblies in form of 3D models. Meanwhile different software solutions offer different additional functionalities (e.g. displaying H-bonds, distances and angles among individual atoms, conformational changes), all have common ground in simple navigation throughout the scene (three mouse button control for rotating, zooming and panning) and offer multiple graphical representations of viewed structures (bond/space-filling/ribbon diagrams).

In contrast to general visualizing tools (NGL, JSmol, Protein viewer), various specialized software instruments provide narrowly focused toolset for performing specific tasks like ligand interaction analysis (Ligand explorer), correspondence between the human genome and 3D structure (Human gene view), or explore metabolic pathways with PDB structures (Pathway view).



a) b)

c)

Figure 14) Zika virus inside NGL viewer from various distances: a) Overall view in space-filling representation b) Medium close-up with ligands c) Close-up of ligand binding

1.7 Digital modelling

3D modelling is a process, where mathematical representation of physical object is being developed via appropriate digital modelling tools [61]. Resulting mathematical *model* can be displayed as 2D image through *rendering* process. Models can consequently serve as blueprints for object manufacturing (CNC machining, 3D printing), photorealistic visualization (retail products, architectural design, anatomical parts), or animated movie creation, conveying particular message (educational videos).

For all those various purposes, different software solutions offer optimized tools to solve challenges inherently associated with diverse tasks. Therefore, CAD programs are usually more suitable for object manufacturing purposes, meanwhile creating visually attractive content (educational and commercial images), efficiently communicating complex ideas, may be more conveniently achieved by reasonably chosen animation software.

1.7.1 Animation software

Although same tasks can be achieved by virtually any decent animation software, different packages are usually optimized to suit particular user group's needs. Considering most popular solutions, **Blender** is open-source based 3D modelling and animation software, offering virtually same powerful toolset, as commercial solutions do, making it convenient for *individual* artists and small studios. **Cinema 4D** is appropriate for *commercially* oriented objectives (e.g. motion graphics in advertisement), with plethora of preset options, saving time and money in consumer-oriented production. **Autodesk MAYA** is considered industry-leading standard, with most extensive toolset and strong rendering capabilities. *Collaborative environment* oriented workspace, most suitable for big studios, may seem bit overwhelming for single user at a first glance. Since LT package is more affordable than comparable commercial solutions, MAYA may fit needs of smaller creative groups as well.

Table 4: Animation software comparison

Software	blender	CINEMA 4D	AUTODESK MAYA
Core use	Animation, Games	Motion graphics	Animated movies
Workspace	Well-balanced	Light interface	Extensive toolset
Suitability	Individual	Small studio	Big corporation
Price	Free	Expensive	Affordable

1.7.2 Image creation workflow

Object within these tools are usually created as **polygonal** meshes, although **curve modelling** is suitable for smooth, curved surfaces, which may be consequently *tessellated* (transformed into polygonal objects), prior to rendering. **Surfaces** are defined either by texture mapping (assigning bitmap image to object parts via UV coordinate system), or via *procedural materials*, with parameters (including appearance and light interaction qualities) configured within modelling software's material toolbox. **Lighting** then provides scene's fundamental *illumination* (for proper "exposition", contrast among objects, image depth illusion), as well as *visual subtext* (directing viewer's attention, establishing atmosphere, revealing character's mood). Appropriately setting **cameras** should compose final images for providing desired experience to the viewer (establishing spatial context, following biochemical cascade clearly, totally immersing into artificial world). **Rendering** creates final images from models according to data relationships established in previous steps. Scene parameters should be optimized (e.g. setting LoD appropriately for different objects) and available computational resources considered (single CPU render times will be considerably longer in comparison to cloud computing solutions, when maintaining same quality and feature's fidelity).

Figure 15) CGI creation workflow

1.7.3 Animation techniques

Motion illusion of static objects (*animation*) may be achieved by traditional **keyframing** approach, allowing precise movement control over time. For particle-based animation, where very large number of *primitives* compose resulting "object" (liquid sprays, water elements, blood cells in vein), **particle systems** allow their effective management. Utilizing physical fields, emulating real-world forces (such as gravity), may approximate their apparently realistic behavior (based on scientific knowledge compliant with observed phenomena).

Although carefully handcrafted models and artificially simulated phenomena may seem very convincing, it is essential to bear in mind, that **physical accuracy cannot be achieved** this way. Creator thus shall compose scenes forethoughtfully to assure, that depicted phenomena are displayed in sufficiently physically accurate manner, as errors and inaccuracies may be replicated multiple times, by reproducing art-work by other people in their works, contaminating current level of scientific knowledge.

1.8 Best of all worlds

Standard modelling and animation methods lack experimental data driven functionality features, therefore creating physically accurate scenes directly would require enormous time expenditures and very profound understanding of molecular biology processes, as well as comprehensive modelling skillset for a particular application. Special toolboxes have been created for popular animation software packages (including molecular Maya toolkit and BioBlender), which allow *importing* experimental data driven models directly into particular software, visualizing them in commonly used molecular representations (bond diagram, space-filling diagram, volumetric surface) in case of molecular structures, and **working with them in physically accurate manner.**

Molecular structure based models may be usually imported directly into PDB format, while CT/MRI data require more sophisticated approach. Surface reconstruction of anatomical model needs to be created from CT images (DICOM format) via appropriate software (3D Slicer, Fiji) and exported in Maya importable format (.STL). These structures, while retaining physical accuracy, may be worked with as other Maya natively created objects.

As physically accurate structures may be directly imported and consequently animated to e.g. depict signaling pathway, whole workflow simplifies significantly, allowing creator to concentrate more on core message conveying aspects (object appearance features, image composition and camera's movement, incorporating supplemental content, such as Brownian motion), considerably contributing to cost efficiency of creating final images.

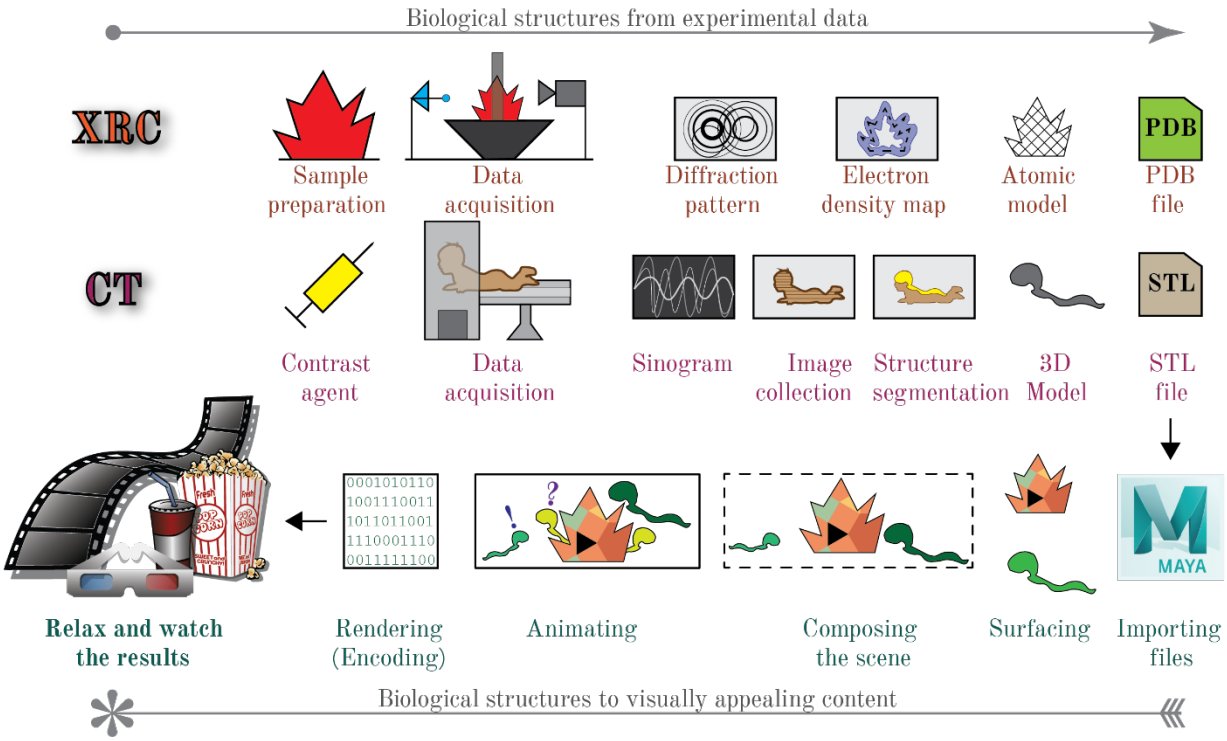


Figure 16) From experimental data to visually appealing content

1.9 Example: Blood flow animation

Red blood cell and vein geometry were modelled with curve modelling tools and subsequently tessellated into polygonal meshes. Soft tissue and white blood cell geometry were created as polygonal objects directly.

Blood flow simulation utilizes native *nParticle* system with physical field (pink circles visible in figure 16c) along vein axis to drive particles motion (speed, random particle rotation, growing viscosity towards vein walls, flow pulsations). Another field, with adjusted parameters, was applied to left branch (referred to as “empty”) for diversifying flow strength in both branches. Blood cell geometry was attached to generated nParticles via an *instancer*. Custom attribute “IndexPP” was added in nParticle geometry node as per particle attribute, driven by custom script assigning binary value [0, 1] to each generated particle randomly, with specified probability (0,3%). Based on this value, either white or red blood cell geometry is attached to corresponding nParticle.

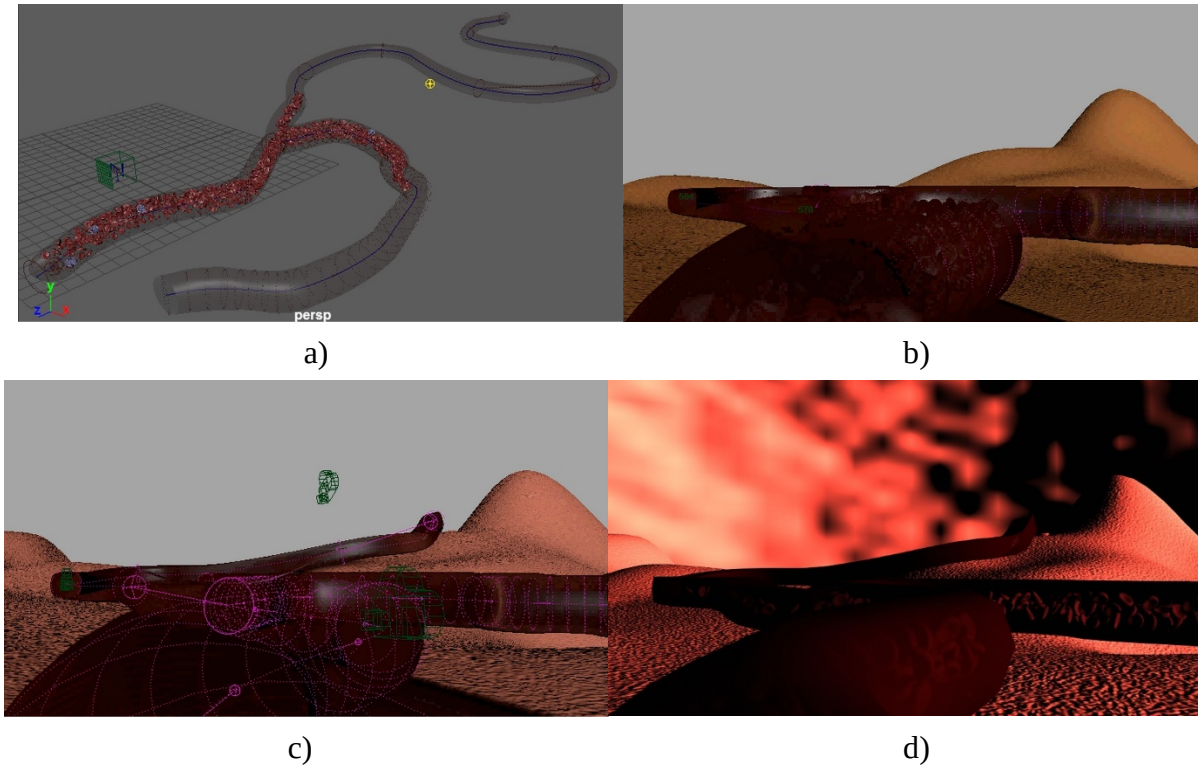
Surface materials are based on *Blinn* material type. Bump mapping node, driven by volume noise, creates an illusion of depth with minimal computational costs. Two distinct materials were attached to vein walls: Opaque material is bump mapped with *perlin* noise quite strongly; meanwhile transparency node of semi-transparent material is driven by camera’s point of view via conditioning process, utilizing sampler’s node facing ration value. Each material visibility is complementarily turned on/off, based on camera’s point of view, assuring that during interior shots, vein walls stay opaque, while exterior shots utilize semi-transparent material.

Seven distinct cameras were created for taking individual shots. Their spatial coordinate values change over time (translationally rotational movement) either by keyframing them manually (assigning values at given frames), or via *motion-trails* for immersive shots, imitating other than camera’s Point of View. Their temporal switching was performed within *camera sequencer* editor.

Backdrop environment is basically polygonal sphere with enormous radius, enclosing whole scene, requiring light source without spatial intensity attenuation, therefore *directional light* serves as main light source for exterior scenes, providing sufficient basic illumination. *Ambient light* with very low intensity value (0.025 in contrast to 1-5 for other lights) softens darkest areas in visually plausible way. Each camera is accompanied by its own *area light*, providing very soft illumination, albeit requiring most computational time of all available basic light types. For a given shot, only necessary lights are enabled (ambient light and corresponding area light within interior scene, where directional light is disabled), saving thus computational resources.

Final images were rendered with Maya's native software renderer as series of 960x540 PNG images with 150 pixels/inch spatial resolution. Renderer settings were empirically balanced for providing best quality at reasonable rendering times on particular computational machine.

Unfortunately, Maya does not allow image rendering from different PoVs, established within camera sequencer, directly in any of prevailing software renderers, therefore customized script, creating batch-rendering file harvesting camera sequence-related information, was utilized. Batch file was consequently executed in windows command line, initiating thus *background-rendering process*, utilizing available system resources most effectively of all rendering approaches.

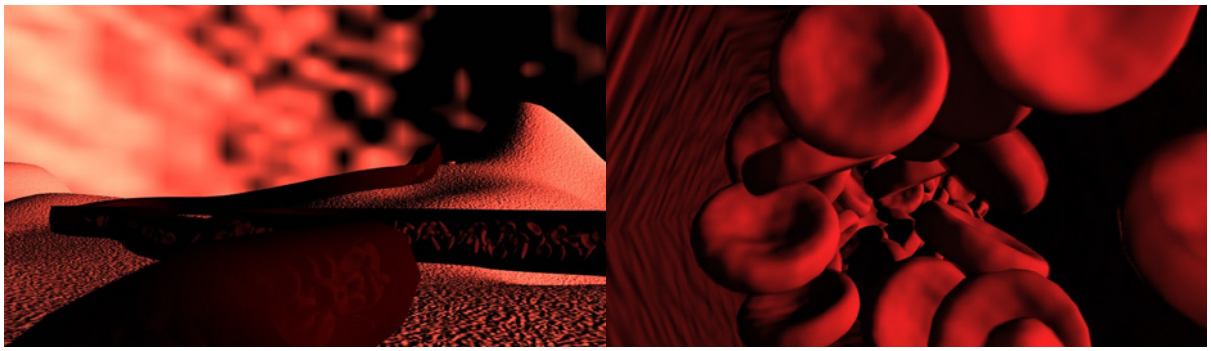


a) b)

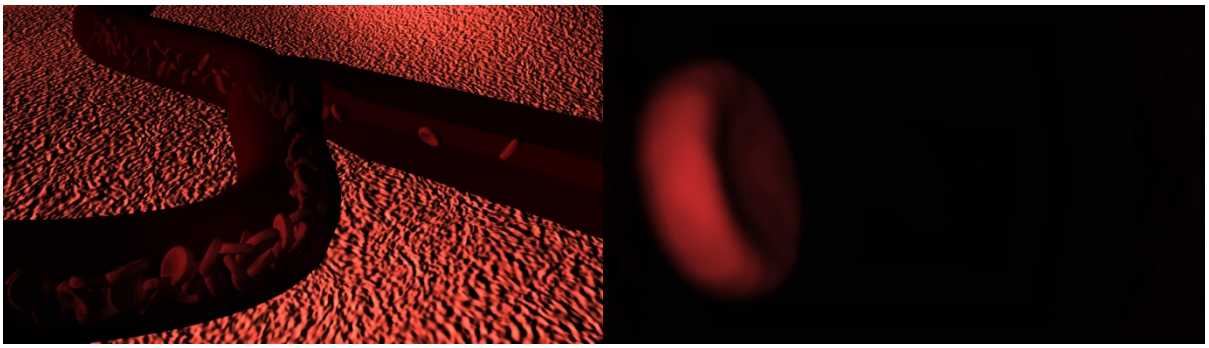
c) d)

Figure 17) Selected stages of diegesis creation

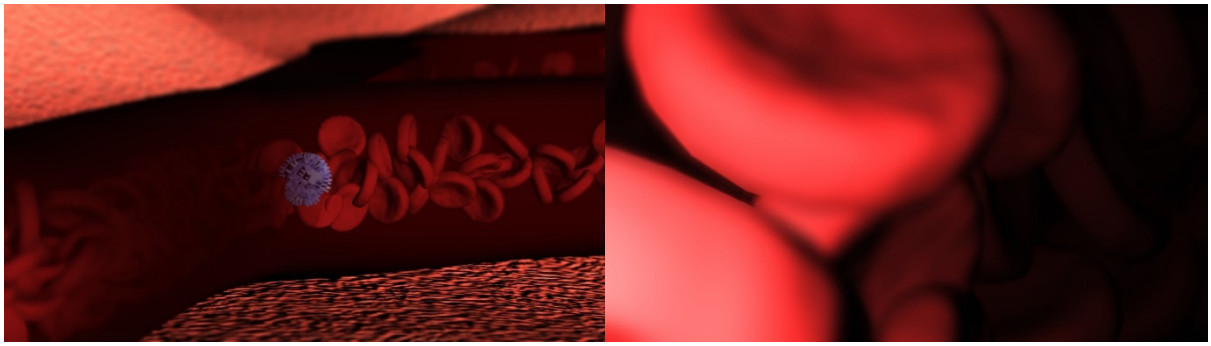
a) Finished blood flow simulation in Maya's viewport
b) Molded tissue and semi-transparent walls provide visual context
c) Enhancing surface material properties and establishing cameras (composition & movement)
d) Adding backdrop environment and lighting setup for improving overall visual plausibility



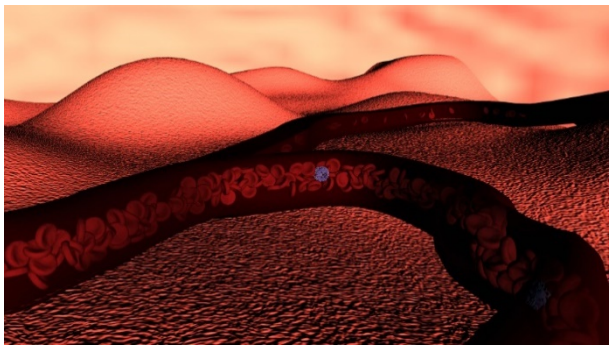
a) b)



c) d)



e) f)



g)

Figure 18) Blood flow final images selection rendered via Maya's native software renderer

a) Establishing shot giving overall idea of blood cells flowing throughout the vein

b) Camera intersects vein's walls to see blood flow directly (notice opaque vein walls)

c) Exterior shot reveals vein branching (d) Upstream motion in empty branch creates tension

e) Rare white cell (f) Immersive cell's PoV by "flowing camera" (g) Closing (summarizing) shot

Part II:
Modelling and animation of biological structures

2 AUTODESK MAYA®

In Hinduism and Vedic texts, there is a term “MAYA” denoting a magic show, an *illusion*, where things appear to be present, but are not what they seem on the first glance. Hence the title of Autodesk’s 3D computer graphic software should imply its extraordinary power and capabilities, when wisely and skillfully utilized.

This software package can run on all commonly employed operating systems (Microsoft Windows, Apple Mac OS, Linux), although software installation on Linux can be a little bit more tedious, than on other systems.

From hardware point of view, 2017 and newer versions require 64-bit processor architecture (both Intel and AMD), at least 8 GB space in random access memory and 4GB on hard-disk for installation itself, but additional content, as well as cache files and final images, would demand much more space. Our personal recommendation is 128 GB at least, dedicated to MAYA related files. List of certified graphic cards can be accessed via Autodesk knowledge network website (https://knowledge.autodesk.com/support/maya/troubleshooting/caas/simplecontent/content/maya-certified-hardware.html), although any decent card with at least 1.5 GB RAM memory should get the job done.

Table 5: Autodesk MAYA® hardware requirements

Hardware	Minimum requirements
CPU	64-bit multi-core architecture
Graphic card	Maya certified Hardware
RAM	8 GB
HDD space	4 GB

Various licenses with slightly different capabilities are available for satisfying specific end-customer’s needs, ranging from single users, through small and medium sized CGI creating companies, up to big Hollywood studios. 3-year Student & Educator License is also available, but external renderers, or utilizing some advanced rendering capabilities of integrated ones, may require separate licensing process.

Additional information, regarding MAYA, can be accessed via its dedicated website: https://knowledge.autodesk.com/support/maya. Particularly useful are official forum threads, when resolving issues within MAYA workspace, or having a hard time during CGI production.

2.1 Software interface

MAYA interface is organized logically, but provides so much functionality that it may appear a little bit overwhelming at a first glance, therefore is usually good idea to get familiar with it, before jumping right into modelling and animating process.

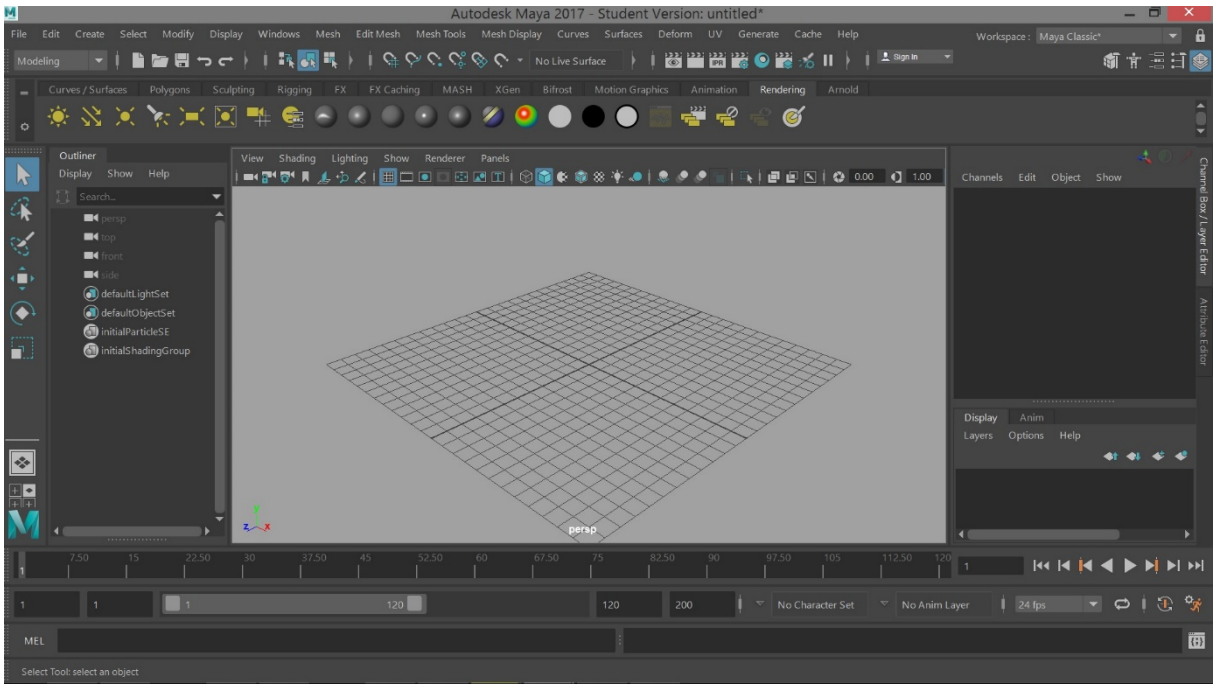


Figure 19: MAYA 2017's Graphical User Interface

We have few different windows, menu sets, lines and helpful graphical icons indicating specific functionality, at hand. The largest area is occupied by a **Viewport**, which can be considered as our worktable, where all created object will appear and can be subsequently worked with. In addition to current *single 3D perspective view*, we have also available other layouts, which are useful for performing various tasks with great precision.

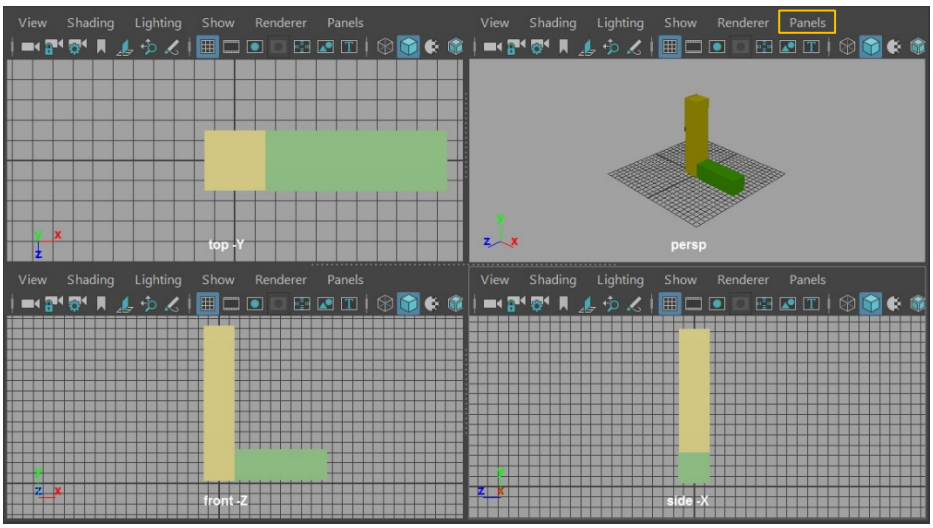


Figure 20: Four-view Viewport layout

2.1.1 Overview windows

We can toggle between those two layouts simply by swiftly pressing and letting go *spacebar* on the keyboard. Displayed content can be accessed via ***Panel*** menu. Other way of choosing predefined viewport layout is selecting them from bottom-side of ***Tool Box***.

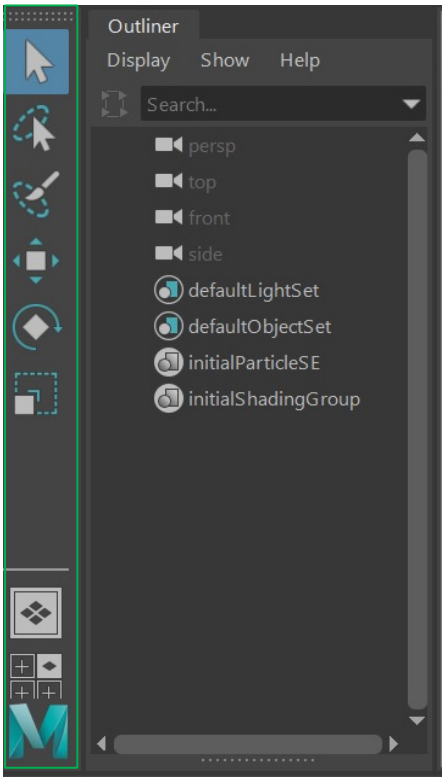


Figure 21: Tool Box

In the upper side are localized *Tools*, most frequently utilized during modelling and animating stages, where currently selected one is being highlighted. They include three different selecting methods (*Select tool, Lasso tool, Paint select tool*), *Move tool* for moving object freely in 3D space, or positioning it along one of three axis (X / Y / Z) at the time.

Outliner window can be accessed via main menu hierarchy *Windows* → *Outliner.* We can find there all objects and nodes within our scene, defaultly containing four *View cameras*, through which we "see" inside individual *Viewport* windows. *Default LightSet* provides basic illumination to the scene, when no light has been manually created, otherwise *Viewport* would appear as pure black (no photons reflecting from objects, consequently passing into *Viewport's* camera).

Initial Shading Group provides default *Lambert-type Material*, telling a renderer, how object's surface should interact with present lighting, so we can actually see it in Viewport without necessity of setting lights manually.

On the right side, there is a ***Channel Box / Layer editor***, which allows us to quickly access object's attributes, commonly edited during modelling and animating stages. In the upper part are located attributes common for all objects (*Translation, Rotation, Scale*) within 3D space, along with *Visibility* accepting only binary values, so that object can be either turned on, or off (hidden). Lower part (INPUTS) contains object-type specific attributes, such as *Width* and *Height* for e.g. polygonal cube, or *Diameter* in case of a polygonal sphere.

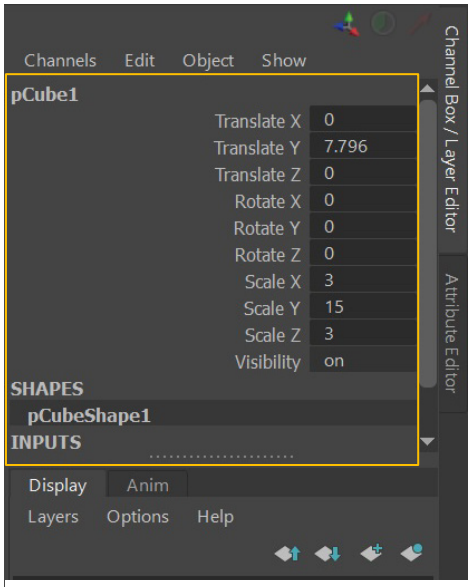


Figure 22: Channel Box

Underlying tabs *Display* and *Anim* allow putting objects onto different layers, proved themselves to be useful for setting rendering passes in previous versions, but since 2016 updated versions, there is much more comprehensive *Render Setup* window, which will be discussed in chapter dedicated to rendering process.

Attribute Editor is basically a list of all commonly utilized attributes, compartmentalized into multiple corresponding sections. Upper *tabs* allow jumping among different *nodes*, controlling different aspects of one object (e.g. *geometry*, *shading*, *applied deformers*).

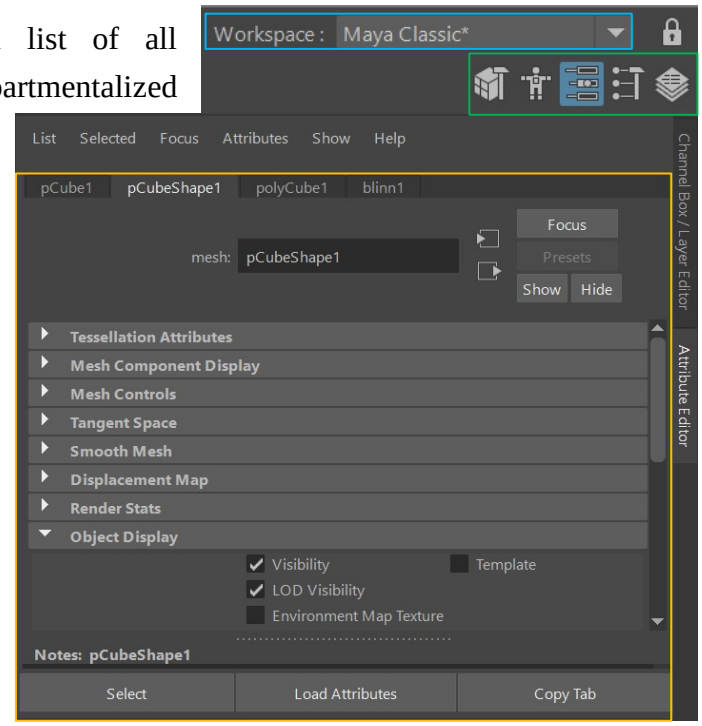


Figure 23:Attribute editor

Both windows (*Channel Box*, *Attribute Editor*) can be switched between via *tabs* on very right side, or showed / hidden by ***Toggle Buttons*** within upper-right menu corner, along with few other windows, optimized for handling specific tasks associated with modelling and animating operations.

It is also the place, where you can find *Molecular-Maya* toolkit, which allows importing PDB files directly into MAYA *Viewport* and selecting their visual representation, after installing it as a *Plug-in* (running an installer and subsequently checking corresponding *mmaya* checkboxes within MAYA's *Plug-in Manager*).

Workspace pull-down menu allows toggling among predefined sets of windows and tools arranged together for performing distinct tasks in different stages effectively, with variants fitting diverse CGI production needs. Creating a *Custom Workspace*, respecting our way of working and personal taste, is a pretty neat new feature, as we can arrange multiple windows among two or more physical screens.

2.1.2 Menus & Lines

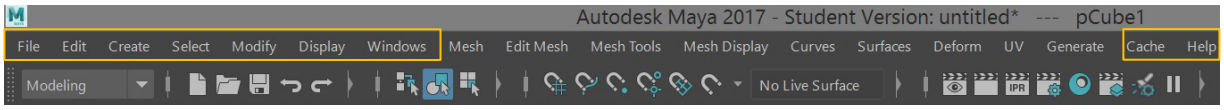


Figure 24: Main menu ribbon

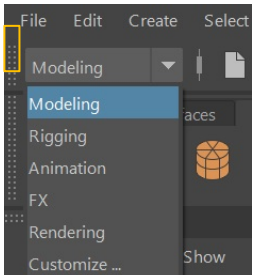


Figure 25: Menu Sets

Main menu may be divided into ***common menu tabs***, always visible at hand, and *custom menu tabs*, containing set of tabs with specific tools, relevant for corresponding production stages. *Menu Sets* are available from pull-down menu just underneath main menu tabs.

Double dotted lines allow you to separate corresponding interface element into its own window, which can be subsequently positioned anywhere on screen, or merged with other separated elements, appearing there as a single tab.

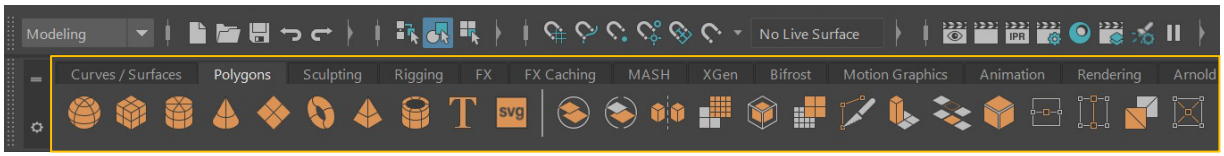


Figure 26: A Shelf

Next to menu sets (currently *Modeling*), are located some control icons for common tasks useful during whole production. These range from file handling operations (*New file, Open file, Save file, Previous step, Next step),* through object selection methods (*select by: hierarchy / object type / component type)* and *Snapping* object variants, up to *Rendering* related tasks.

The ribbon populated with multiple colorized icons is a ***Shelf***, making most commonly used commands, usually accessible from corresponding menu set tab, available at hand. Hovering over individual objects will invoke yellowish help box containing command's label, associated hotkey and information about its purpose. Step by step instructions for using particular tool can be found at the very bottom, within a ***Help Line***, along with various hints, depending on what we are actually doing at the moment.

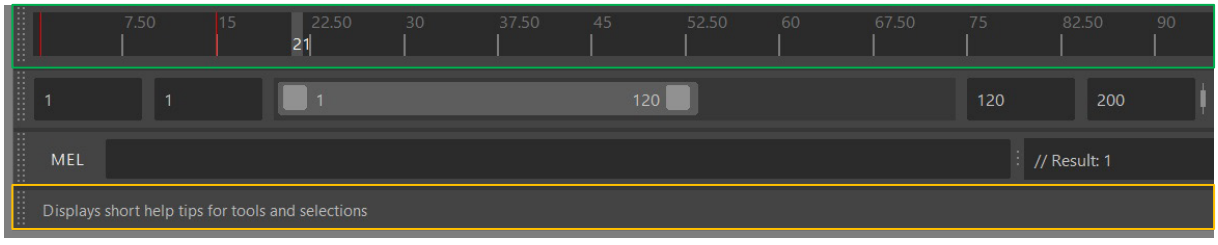


Figure 27: Help Line at the very bottom of MAYA Interface

Timeline gives us an overview of where within animation we currently are (frame 21) and selected object's keyframes temporal position (red vertical lines). We can adjust its range

by *Range slider* located underneath, or numerically typing whole integer numbers into boxes on both sides, where outer ones stand for first and last frame of the entire animation, whereas inner boxes correspond to displayed range (e.g. frames 1 – 120 from 200 frames long animation).

Third line contains a *Command line*, where we can type MEL (Maya Embedded Language) or Python commands, after switching to desired language by clicking on its left-sided label. Next to it is a ***State line*** giving us brief info about results of our last action. Right icon with semi-colon sign opens a *Script editor*, where we can write and run our own scripts.

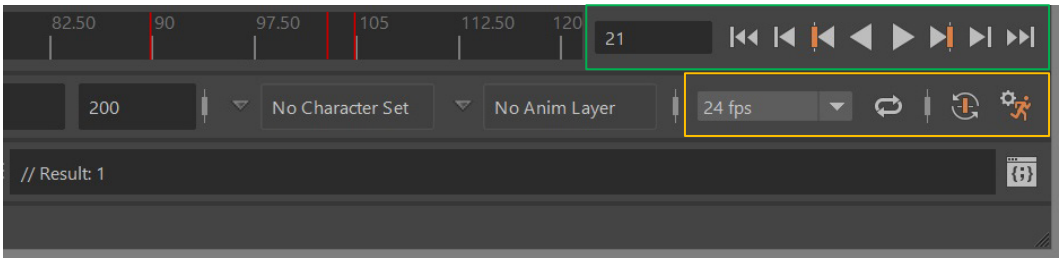


Figure 28: Animation Controls (green) and Animation Preferences (yellow)

Bottom right corner contains ***Playback speed*** pull-down menu (safe bet being 24 fps, but may be changed, depending on particular production needs), ***Continuous loop*** for looping animation, so it keeps playing over and over again, until we manually stop it (useful for procedural animations), ***Auto-Keyframe* toggle**, creating a keyframe for whatever attribute we are currently changing, and a running man icon opening ***Animation Preferences***, as a quick way to access MAYA's preferences (long way being via *Main menu*: *Windows → Settings / Preferences → Preferences).*

Next to the *Timeline* is located a ***Set the current time*** box, through which we can jump to desired frame by typing its number there numerically, and animation ***Playback controls*** for re-playing animation, or stepping on frame-by-frame basis.

As we can see, there are many graphic user interface (GUI) elements, but their design and layout is rather intuitive, aiming for getting our job done comfortably in most effective manner. Informative elements (*Help box*, *Help line*) provide additional hints for using various tools, or give us information about what is currently happening within our project (*State Line*), allowing us to successfully cope with emerging challenges and making potential debugging tasks a lot easier.

2.1.3 Navigation

Navigation within *Viewport* requires a three-button mouse. By holding **Alt** key (**Option** on Macintosh) and **Left mouse button** (LMB) at the same time, we can rotate around the place, where we have clicked. **Alt + Middle mouse button** (MMB) allows *Viewport's* camera moving from side to side, as well as up & down. Third combination **Alt + Right mouse button** (RMB) provides dolly in & out sliding, depending on mouse's physical movement. Similar effect can be achieved by rolling MMB wheel, which moves camera forward or backward in steps, acting as a stepped dolly.

In addition to this essential navigation, we can also set predefined view via *Viewport's* own menu: *View → Default View* (**Alt + Home**) will just restore original view to the scene center that we see whenever creating a new scene. *View → Look at Selection* would re-position the camera, so that selected object appears in the center, but more useful way of doing it is going *View → **Frame Selection (F)***, which effectively sets the camera to let object fit *Viewport's* window. *View → Frame All* (**A**) is handy in more complex scenes with multiple objects that will consequently fit the *Viewport*, allowing us to clearly see all objects within the given scene.

We can also *Undo View change (***Alt + Z***)* and *Redo View change* **(Alt + Y)** via same menu, which may be useful when using carefully set *Camera* and inadvertently changing its attributes by using above-mentioned essential navigation approaches.

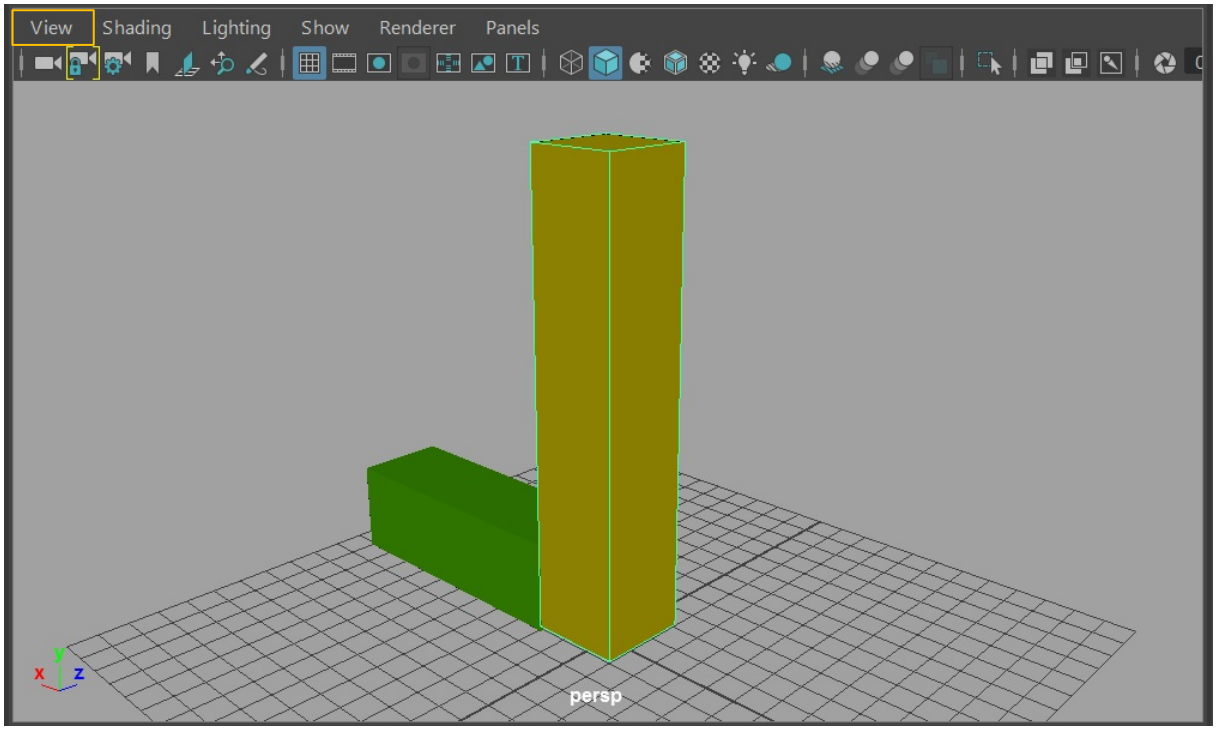


Figure 29: Frame selection fits selected object to Viewport

There is one more way for quickly selecting pre-defined view: Holding **Space-bar** will reveal a ***Hot-Box*** menu, which gives us instant access to all our *Main menu's* items. Subsequently holding any mouse button on "***Maya***" option, while still holding Space-bar key, lets us choose one of pre-defined orthographic views, or the *Perspective View*.

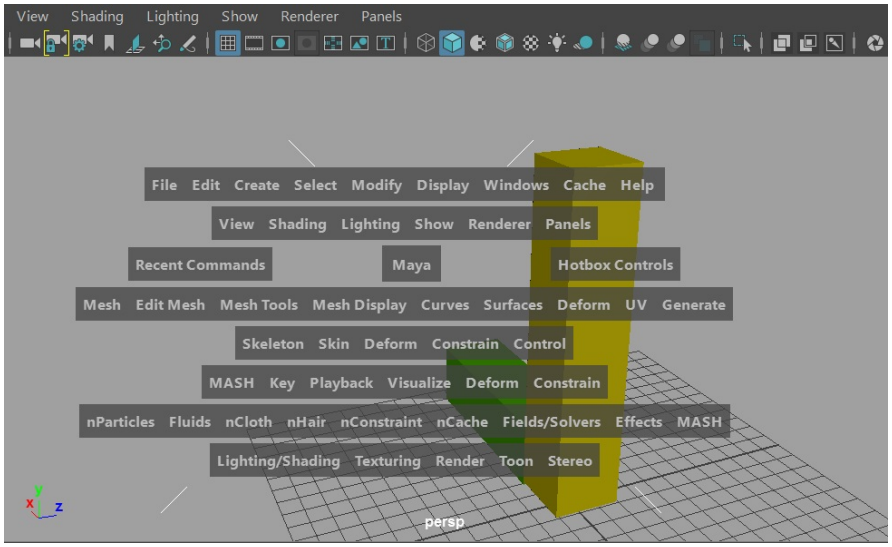


Figure 30: Quickly accessing menu commands via a Hot-Box

When modelling and surfacing, object-specific operations can be also readily revealed within a *Marking menu* by holding RMB, which may save us a lot of time during these production stages.

Table 6: Quick but powerful navigation shortcuts

Modifier key	**Standard key**	**Action**
Alt [Opt]	LMB	Rotating View
Alt [Opt]	MMB	Moving side to side, up & down
Alt [Opt]	RMB	Dolly in & out
-	MMB	Stepped Dolly
Alt [Opt]	Home	Default View
-	F	Frame Selection (Fit window)
-	A	Frame All
Alt [Opt]	Z / Y	Undo / Redo View Change
-	Space-Bar (hold)	Hot-Box
-	RMB	Marking menu

2.1.4 Communicating with MAYA

There are multiple ways of working with MAYA, depending on what we are currently working on, our skills and specialization (e.g. performing only one productional stage tasks), as well as our personal taste making us feel comfortable.

Using ***Menus & Shelf*** is a very simple method, when searching possibilities and most suitable tool for performing the task we are currently dealing with. Graphical icons, as well as *Help Line*, gives us hints about various features, therefore it is a great approach when learning to work within MAYA environment, although faster methods for achieving most tasks can be adopted, after getting familiar with our favorite features.

Hot-Box provides basically the same functionality as menus, integrating *Main menu* and *Viewport* menu at one place. Since it appears exactly, where our mouse cursor is currently located, this way is much faster, than digging through menus, after we accustom to its layout. ***Marking menu*** may be one step quicker, containing only pre-selected modelling and shading task's commands.

Abovementioned approaches allow us to **execute MEL commands** via a GUI, but these can be also written straight into *Script Line*, or more of them within *Script Editor* as an executable script file. Common production and creative tasks would be extremely cumbersome and time consuming by approaching this way, but using scripts for automating routine technical tasks may save us a lot of time as well.

They are also handy as ***Expressions***, when creating some advanced object dependencies during animating stage, which would be nearly impossible to achieve any other way.

Table 7: Comparing MAYA's essential GUIs

Interface	Best use	Knowledge
Menus & Shelf	Learning MAYA	Basic PC skills
Hot-Box & Marking menu	Working productively	Knowing MAYA
Script Line & Script Editor	Automating processes	Advanced expertise

2.2 Modelling tools

In order to produce any computer-generated images, digital *models*, representing objects of our interest, needs to be created in the first place. And although we may *import* some *models* from external sources, they usually require additional work, before can start serving our purpose. Furthermore, situations may come, in which we basically cannot import desired geometry, or we need *models* specific to a given scene only, therefore understanding how to effectively utilize basic modelling approaches is essential for performing any work within CGI software competently, as some decisions made during this stage are crucial to resulting visual output.

2.2.1 NURBS vs. Polygons

Autodesk MAYA offers two rather complementary approaches for creating geometrical shapes. ***NURBS*** objects are shaped by *splines* rather than straight *lines*, as in case of ***Polygonal*** ones, hence the acronym standing for "*Non-uniform Rational Basis Spline*". They are appropriate, when modelling smooth, naturally curved surfaces, but since fine details can be much easily tinkered in polygonal ones, our general modelling workflow should be based upon polygonal shapes, as they will comprise most resulting objects entering the renderer.

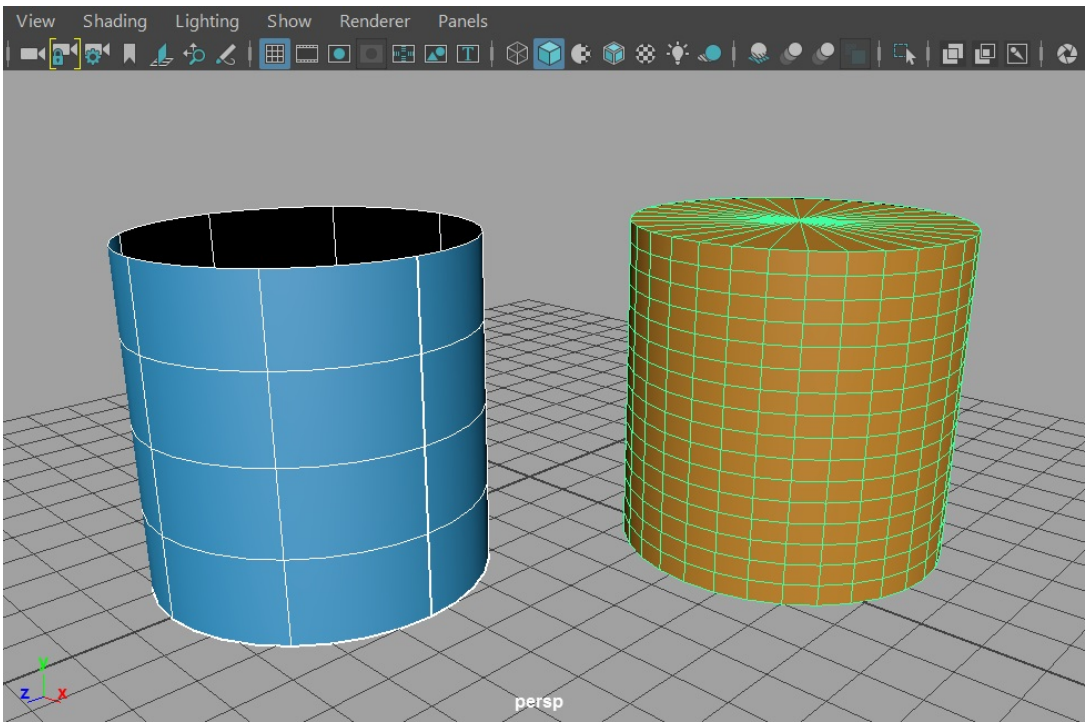


Figure 31: NURBS shape (left) provides more curvature than Polygonal one (right)

2.2.2 Polygonal modelling

Modelling polygonal object starts with creating a ***Primitive*** (basic pre-defined shape), which can be subsequently shaped by various polygonal modelling tools. Instant creation can be accomplished by clicking on corresponding primitive icon and adjusting default shape attributes afterwards.

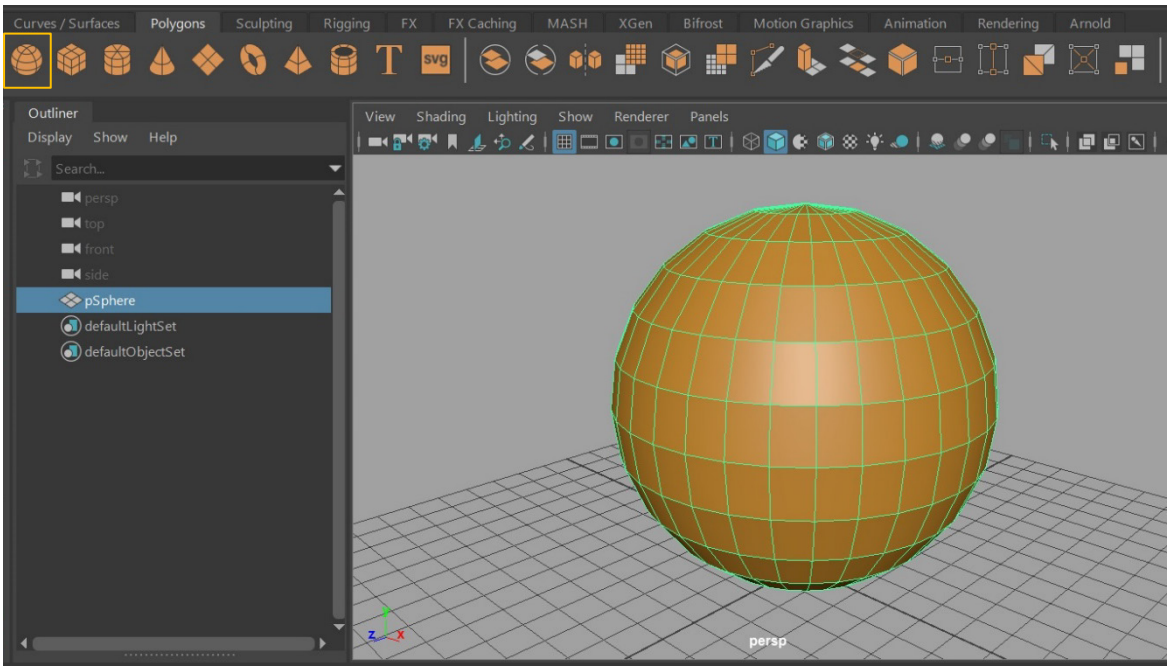


Figure 32: Creating a Polygonal sphere Primitive

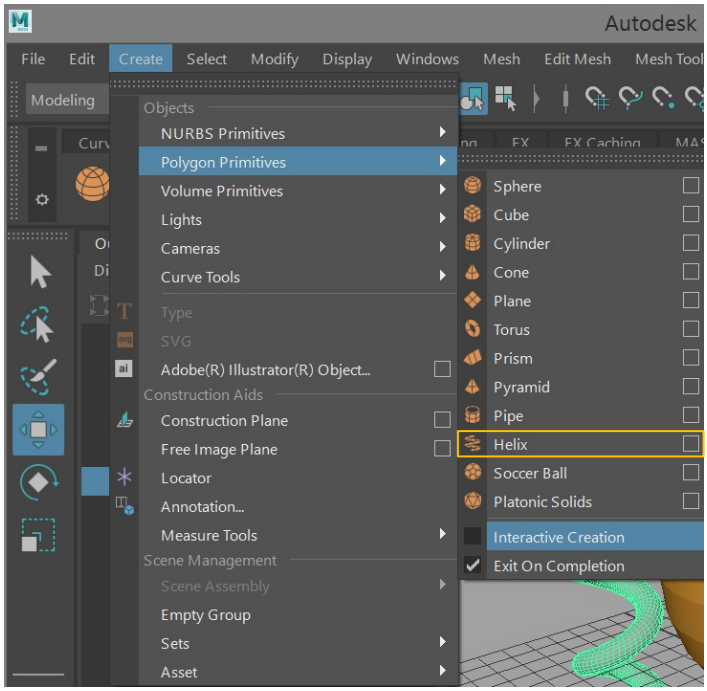


Figure 33: Main menu contains additional options

A few more shapes can be found by going *Main menu: Create → Polygon Primitives → [**Primitive**]*. From biological point of view, a ***Helix*** is very interesting one, allowing us to set up DNA backbone representing shapes rather quickly.

That little ***square boxes***, next to each primitive type, are opening optional window, where object's attributes may be set before creation, but we can usually adjust them later in the *Attribute editor*.

Interactive Creation checkbox provides alternative way of creating polygonal primitives by "drawing" shape's essential attribute value (*Width, Height, Diameter, etc.*) on the grid, so we can assess its proportions in context with other objects during creation.

2.2.3 Adjusting attributes

Whichever way we have created a polygonal *Primitive*, next step is adjusting its attributes:

Attribute Editor contains multiple tabs, corresponding to different aspects of selected object. ***PolySphere1*** contains basic *Polygonal sphere Primitive* related attributes. ***Radius*** determines model's actual physical size, whereas ***Subdivision Axis*** and ***Subdivision Height*** allow us to assess its quality (how many polygons form an actual object) – higher values mean more details, but higher computational requirements, therefore should be set to reasonable values.

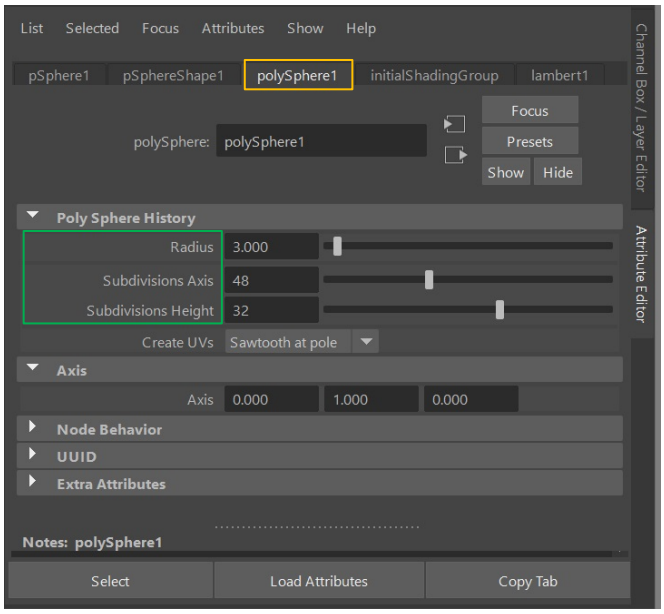


Figure 34: Basic Primitive's attributes

2.2.4 Component selection modes

For actual shaping, we can use object's ***Components*** (elementary sub-objects forming into geometry), accessible by Marking menu. In case of polygonal models, these are ***Vertices*** (1D), ***Edges*** (2D), and ***Faces*** (3D), where an *Edge* represents straight line between two *Vertices* (points in 3D space), whereas *Face* corresponds to surface area bordered by four *Edges*. After finishing adjustments, ***Object Mode*** allows selecting object as an individual entity (default selection behavior).

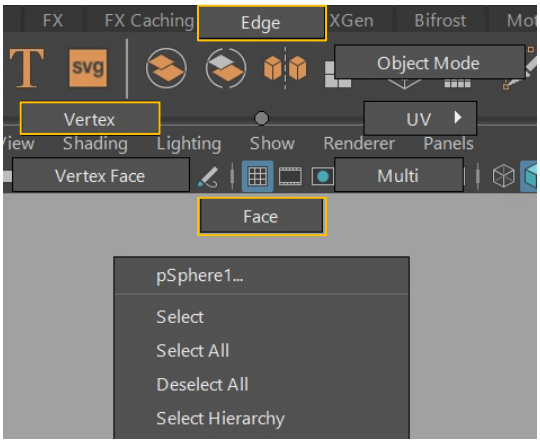


Figure 35: Selecting object's components

By choosing ***Vertex*** mode, magenta dots appear all over the object. These represent individual *Vertices*, whose position can be adjusted with standard *Move Tool*. To effortlessly shape our polygonal sphere *Primitive* into a red blood cell resembling geometry, we need to prepare our *Selection* and *Move* tools accordingly.

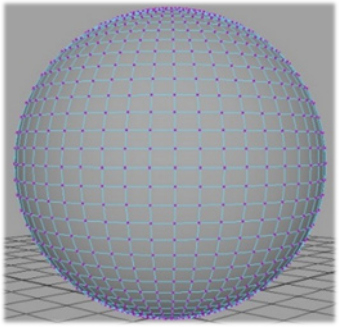


Figure 36: Vertex mode

2.2.5 Soft Selection

Tool Settings can be accessed via toggle buttons in top-right corner of MAYA's interface. ***Soft Selection*** semi-automatically selects multiple *components* in the vicinity of manually specified ones, based on current tool settings. For shaping objects organically, ***Falloff mode*** should be either *Volume*, or *Surface*, the latter one being usually a little bit more reliable. Similarly, ***Interpolation*** should be either *Smooth*, or *Spline*, depending on our taste. ***Symmetry Settings*** allow selecting corresponding *components* "mirrored" over specified axis. Adjusting tools this way should provide reasonable results for shaping polygonal objects rather organically.

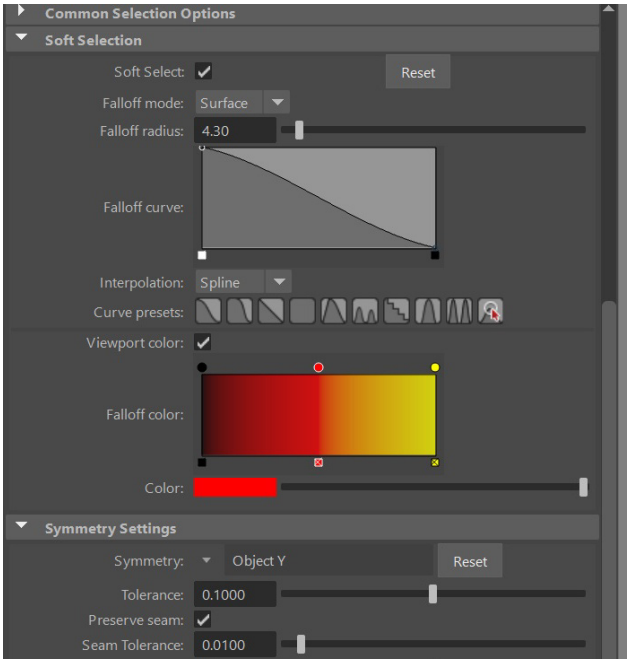


Figure 37: Tool settings for organic selection

2.2.6 Shaping geometry

After setting tools to suit our needs, we can select desired components and move them to shape the object at will, in this case selecting upper-most vertex and subsequently moving it downwards along "Y" axis. To achieve optimal result in a particular situation, we may tinker with tool settings for some time, adjusting especially *Falloff radius* and *Falloff curve* shape.

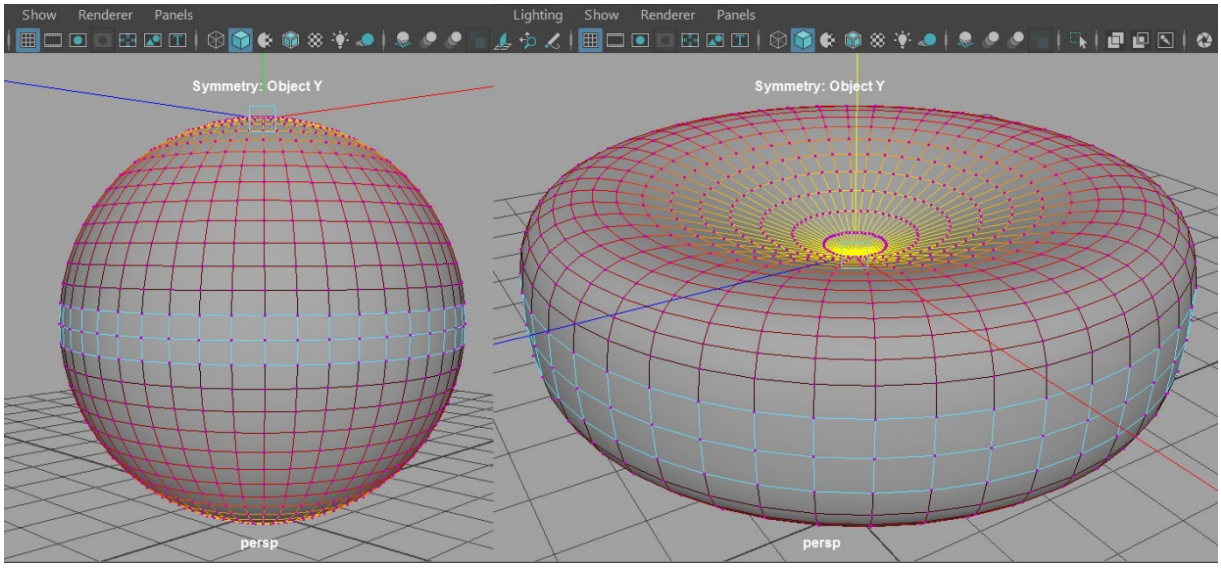


Figure 38: Shaping Polygonal Sphere (left) into Red blood cell geometry (right)

2.2.7 NURBS curves

NURBS surfaces may be also created from simple *Primitives*, but except *Sphere*, these constitute of multiple planar "*Patches*" that form final shape. Due to this, their *components* are different from polygonal ones (*Control Vertex, Isoparm, Hull, Surface Patch*), suitable for handling curvature seamlessly, but making details refinement rather tedious task.

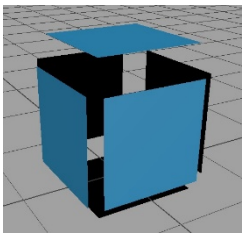


Figure 39: NURBS Patches

More powerful way to modelling NURBS objects is utilizing ***Curves*** and turning them into ***Surfaces*** subsequently. Big advantage, in comparison to polygonal models, is that fairly complex customized shapes can be developed much easily, when organically curved surface is required and limited details are an acceptable trade-off.

Bezier Curve Tool seems to be most comprehensive of all curve-making tools within MAYA environment, as created points are part of the curve itself and hard corners can be easily arranged just by single click to desired place within 3D space. Magenta *handles* provide great flexibility for modifying curvature, and when curve is completed, individual vertices stays in place, so their position may be directly adjusted afterwards.

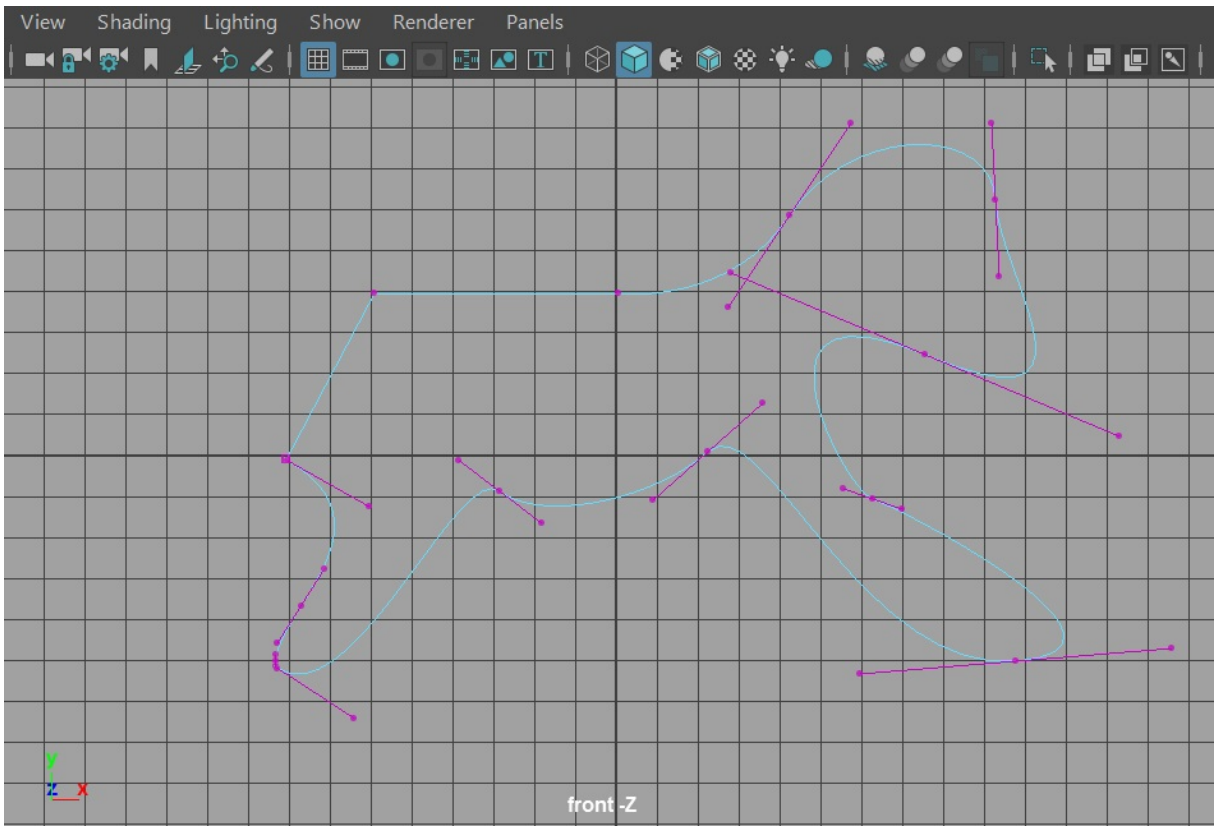


Figure 40: Bezier Curve Tool provides most flexibility

2.2.8 Curves into Surfaces

Curves can be turned into *Surfaces* by pre-defined NURBS surfacing tools. Some can work with only one curve (e.g. *Revolve)*, others require two or more curves / components (e.g. *Loft*). For creating a Blood vessel via *Extrude* surface tool, we need two objects: A *Profile* and a *Path*.

Path is basically a ***Bezier curve***, along which we *Extrude* desired shape, determined by two-dimensional ***Profile*** object (represented here by NURBS circle).

For extruding properly, input objects need to be selected in correct order: *Profile* first, *Path* last. We might change it afterwards, along with other attributes, but doing it properly is just more time-efficient (at these moments, *Help Line* hints are simply a blessing).

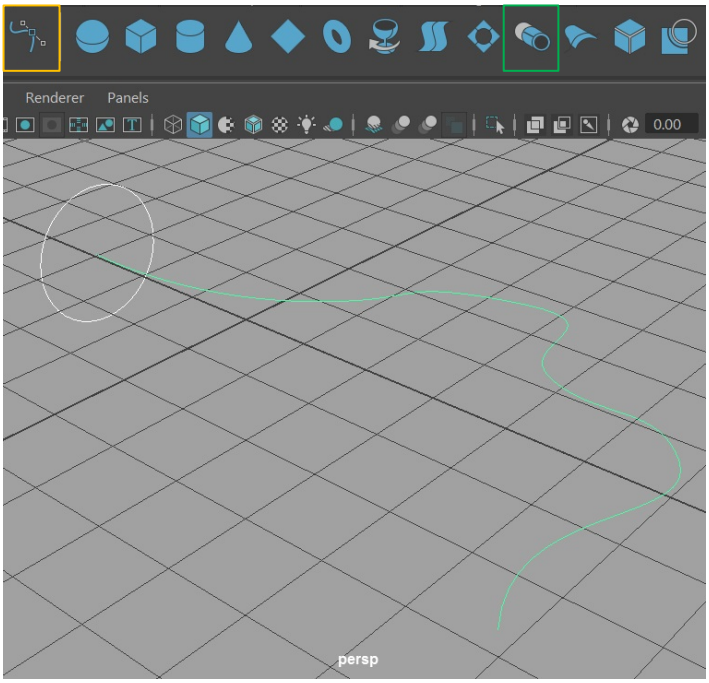


Figure 41: Extrude command requires selecting Profile first (white) and Path last (green)

Clicking on **Extrude** command consequently would create a whole new *Surface*, but via menu *Surface → Extrude [box]* we may open its options and pre-set some absolutely essential attributes, saving our time in the long run.

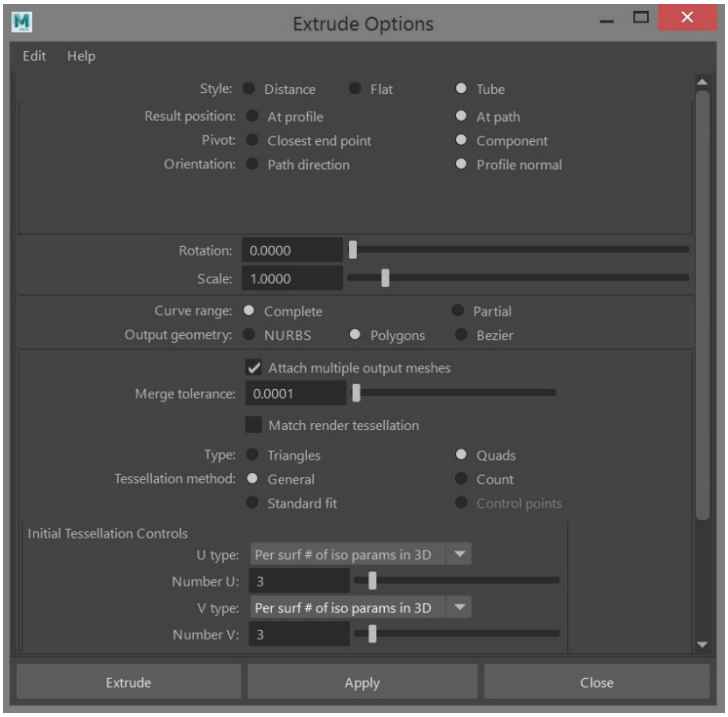


Figure 42: Pre-setting Extrude options

Style set to ***Tube*** will create 3D shape, while *Output Geometry* allows to choose type of resulting object. If we want detailed shape, outputting to ***NURBS*** would require conversion via *Modify → Convert →* ***NURBS to Polygons.*** By selecting ***Polygons***, we save ourselves one step and can fully control quality of resulting geometry by ***Initial Tessellation Controls***, enabled after checking ***General*** *Tessellation Method.*

2.2.9 Troubleshooting common issues

All 3D geometrical objects in MAYA have two sides, whose orientation is determined by *Normals* (lines perpendicular to the given surface). When any model's surface appears as pure black, it is basically "inside out" and its *Normals* need to be ***reversed*** for displaying object correctly in most cases. This may be handled quite easily by one-click solution, requiring one of two dedicated commands, depending on geometry type we are currently using:

For *NURBS* objects, there is *Surfaces* → ***Reverse Direction*** command.
Polygonal objects utilize similar procedure under *Mesh Display* → ***Reverse***.

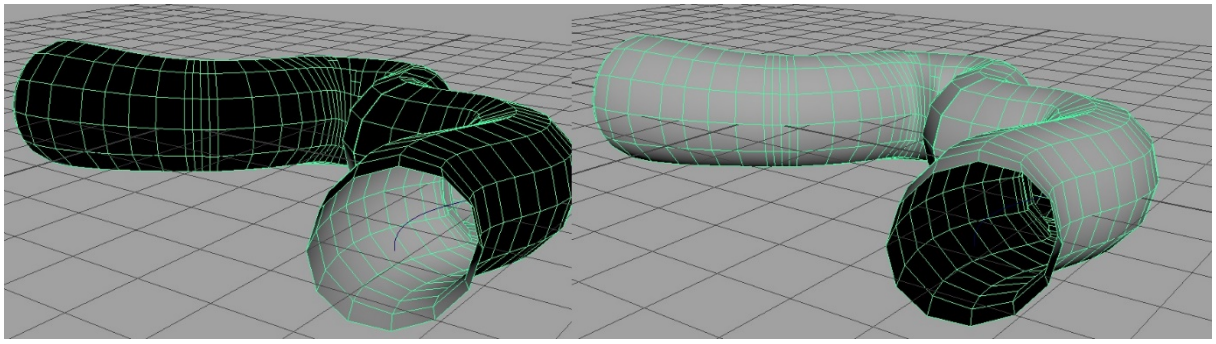


Figure 43: Normals determine which side will be shaded

If resulting tube does not appear exactly along previously selected *Path*, checking *Fixed Path* checkbox, under corresponding tab within *Attribute editor*, may solve it quite easily. In other case, we can revise pre-set parameters inside *Extrude Options* window (*Style*, *Result position*, *Pivot*, *Orientation*).

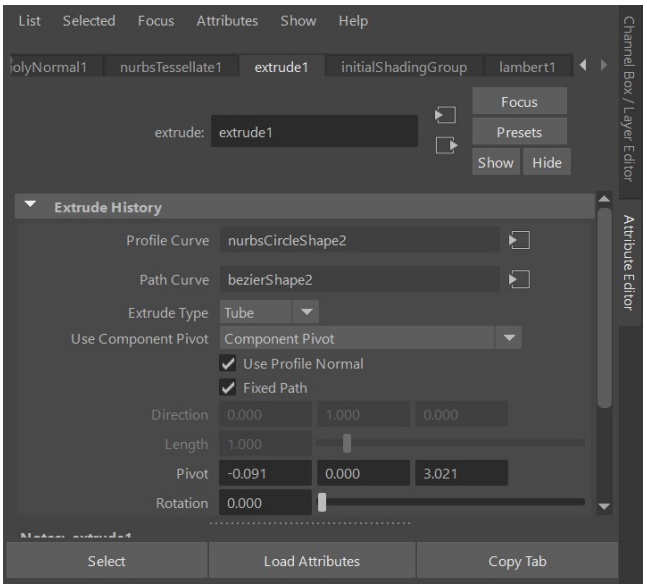


Figure 44: Extrude tab allows tweaking some attributes after creating the Surface

When other challenges come around, searching a little bit on internet forums may help us to find suitable solution (optimal starting points being *MAYA Knowledge base* and eventually also *CGTalk forum*). Chances are, other people were coping with similar situation at some point and solutions they have found useful may help us as well.

2.3 Shaders & Materials

When geometrical objects are created, their overall appearance can be controlled by *assigned material*, which is basically customized instance of a *Shader*, telling renderer, how object's surface responds to incoming light. In addition to those ***procedural Materials***, object's appearance may be also given by external ***Bitmap texture*** via projecting its individual pixels onto selected geometry.

2.3.1 Lambert shader

Since *Shaders* are telling selected *Renderer*, how to display object's surface, they are usually *Renderer-specific*, and may not be compatible with other renderers. MAYA also have so called ***Standard Shaders,*** working correctly in conjunction with all integrated renderers (Maya's Hardware / Software Renderer, Mental Ray, Arnold Renderer), therefore understanding them is essential for establishing materials independently on whatever renderer is utilized during particular production.

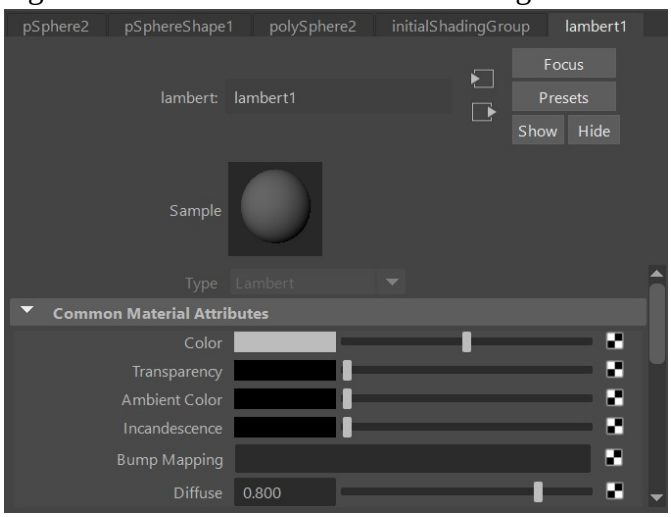


Figure 45: Lambert Shader attributes

Lambert shader was the first one delivered with MAYA staying there up to these days, and its instance called ***Lambert1*** is the default *Material* automatically assigned to any created geometry. Best practice is leaving the instance alone, creating a *new Material*, when object's specific look is desired.

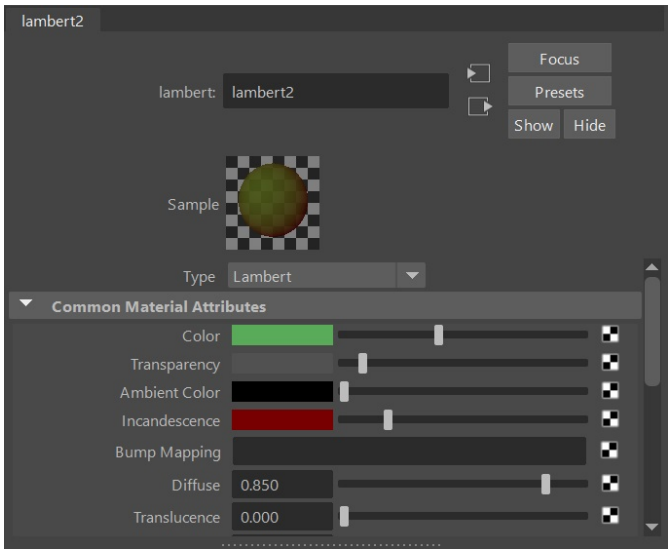


Figure 46: Customized Lambert-type Material

Common Material Attributes are the same among all MAYA's *Standard Shaders*, and although other Renderer-specific shaders may not include them, but usually provide attributes with similar functionality. ***Color*** defines *Material's* overall color shade by combining *Hue, Saturation* and *Value* (Lightness) information.

Transparency allows light to travel through an object by driving *Alpha channel* values represented by its attribute ranging from black (fully opaque) to white (fully transparent).

In *Arnold Renderer*, this functionality is provided by ***Opacity*** attribute, responding inversely to *Transparency* values (white being fully opaque).

If *Ambient lights* are present within scene, ***Ambient Color*** attribute controls how much object's surroundings contribute to *Material's* overall *Color*. ***Incandescence*** simulates light emission from actual geometry, but does not contribute to illumination of other objects. Reflecting light isotopically (in all directions) can be adjusted via ***Diffuse*** attribute, default value being pleasant general-purpose setting.

Translucence allows material to absorb incoming light, diffuse and transmit it, affecting other object within the scene, acting as rough approximation of *Refraction* and *Sub-Surface-Scattering* effects available in more advanced renderer's *Shaders*.

2.3.2 Blinn shader

Although *Lambert Shader* can be utilized to create usable *Materials*, other ones generally provide additional attributes for creating more convenient shading. Main difference, among standard materials, is the way they handle ***Specular Highlights*** (bright shiny surface areas reflecting light sharply).

Concerning customizability and time necessary for setting a specific material properly, ***Blinn Shader*** appears to be single most versatile general-purpose one**,** providing simple, but effective, specular controls for managing bright shiny spots of light reflected from object's surface.

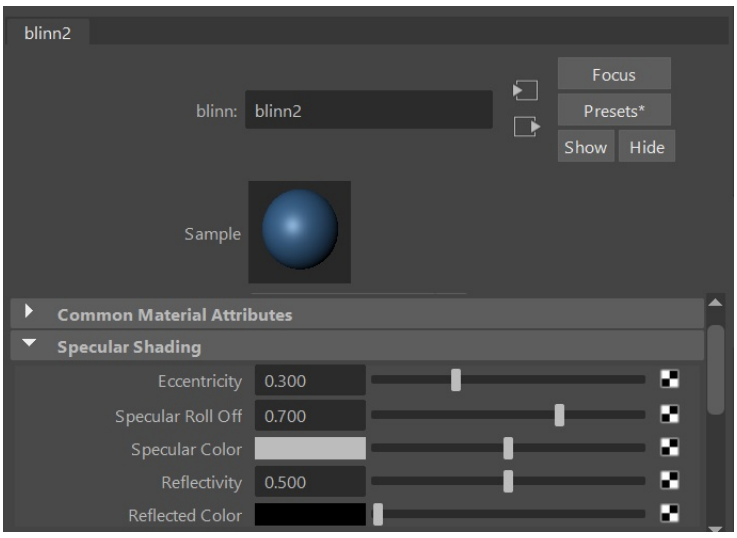


Figure 47: Blinn shader provides decent Specular Shading controls

Eccentricity just determines overall size of the bright area, whereas ***Specular Roll Off*** basically regulates its brightness and sharpness. ***Specular Color*** may be utilized for color-casting bright areas, when desired. ***Reflectivity*** manages how much are nearby objects reflected on geometry, to which Blinn-type *Material* is attached.

2.3.3 Anisotropic shader

An ***Anisotropic Shader*** resembles more customizable version of *Blinn* shader, allowing *Specular Highlights* behaving differently in different directions, which is useful for creating e.g. elongated metallic surfaces, such as electrodes. Although using different attributes, these provide similar functionality (***Fresnel index*** regulating brightness and sharpness analogously to ***Specular Roll Off*** within *Blinn* shader).

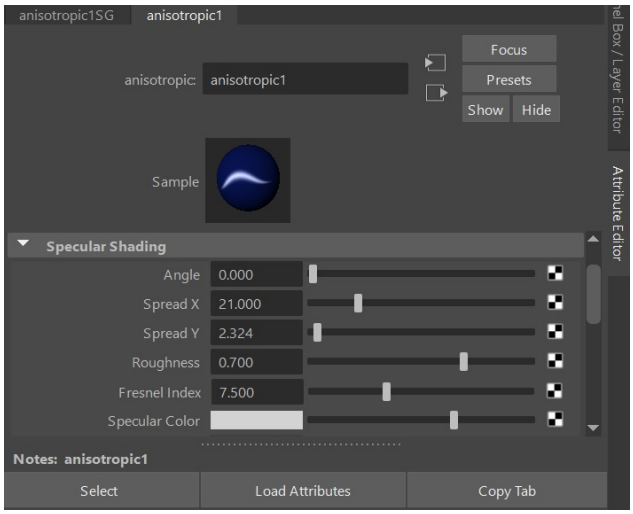


Figure 48: Anisotropic Shader allows directionally customize Specular Highlights

Since *Anisotropic* shader offers more attributes to customize overall look of resulting material, setting it properly for particular situation requires more time, than *Blinn* shader, which is more suitable for most objects without special requirements, as it can resemble metallic and plastic surfaces conveniently, but also behave like standard *Lambert* material, when specular controls are turned down to zero.

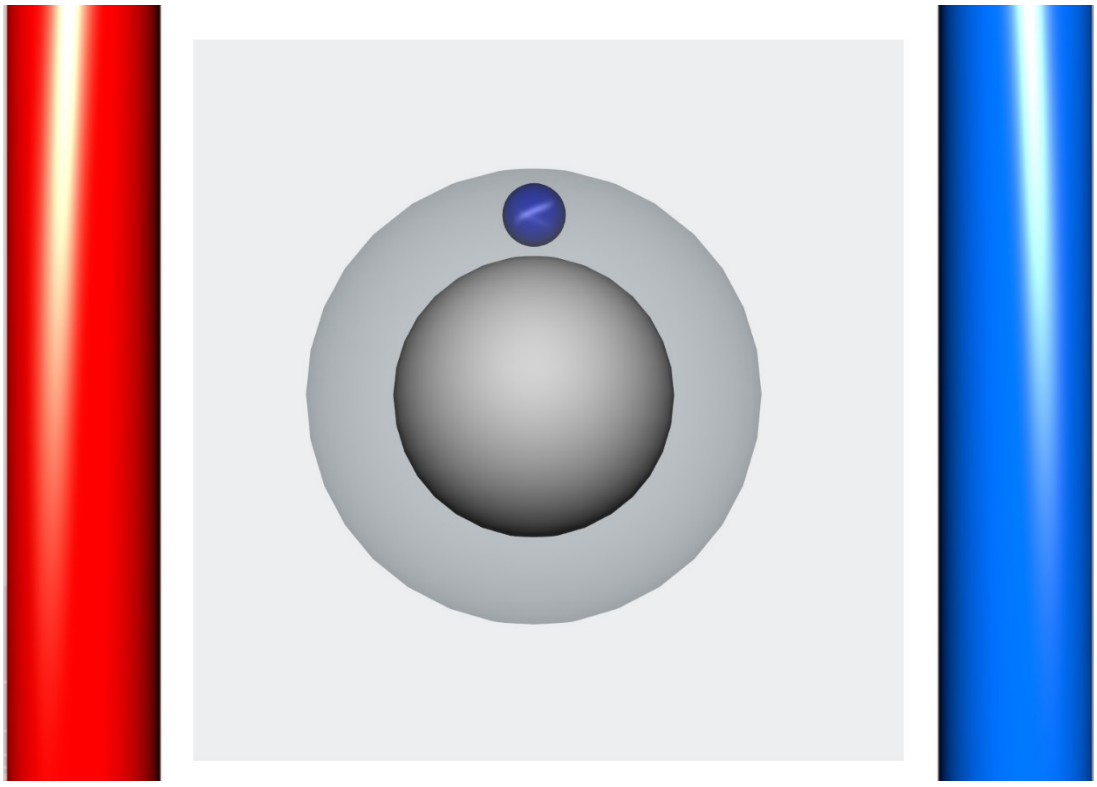


Figure 49: Customized shaders may resemble various materials

2.3.4 Hypershade window

Although materials may be managed via *Attribute editor*, creating and evaluating more sophisticated ones would be extremely cumbersome, therefore ***Hypershade*** window is available as its own interface, where custom *Shading networks* can be vividly build.

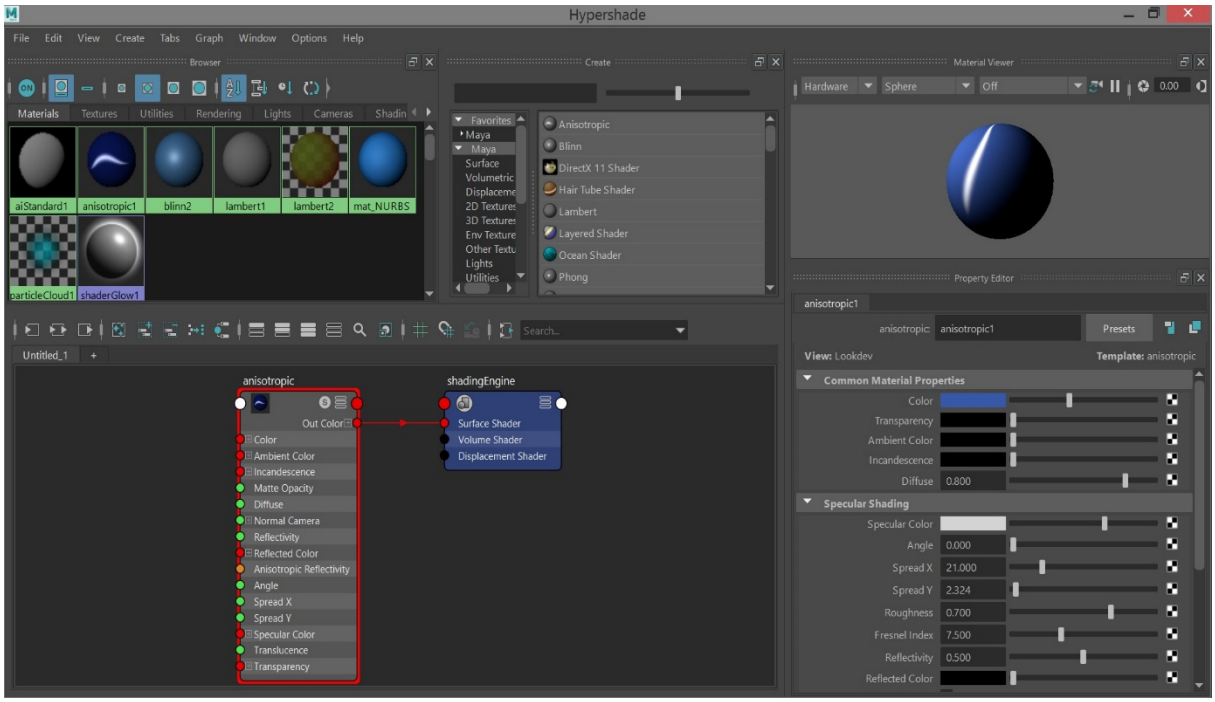


Figure 50: Hypershade window for building Shading networks

Material Browser in the upper-left corner gives an overview of present materials, whereas new ones can be created in many ways. ***Create Render Node*** window shows available *Nodes*, which may be utilized for building *Shading network,* with resulting material being visible on selected pre-defined object within ***Material Viewer***. Individual node's attributes can be adjusted via ***Property Editor***, resembling the *Attribute Editor*. Finally, the largest working area in bottom left corner is a ***Node Editor***, where individual *Shading networks* can be clearly displayed and properly managed.

Currently, an *Anisotropic1* material represents a node, which output color values enter *Shading Engine* as *Surface Shader*, meaning its values will affect attached object's *Surface* appearance. Individual node's attributes are displayed with different color, depending of what numeric type they are, providing thus a visual clue to which attribute can be our selected one attached, feeding it with *outputting* numerical values.

Clicking on a ***Checkerboard box*** next to corresponding attribute slider within *Property Editor* will invoke *Create Render Node* window offering available nodes that can be attached to it as a controller driving attribute's values. This way, we may establish fairly complex *Shading networks* for **texturing object procedurally(!)**, so they will retain same amount of detail when scaled up and can be changing over time in precisely directed manner.

2.3.5 Creating procedural textures

As apparent from chapter dealing with modelling geometrical shapes, creating very fine details manually can be quite time-consuming task. Sometimes, we just need to add some bumpiness to object's surface, so it does appear more realistic, than a flat surface. This can be efficiently performed via ***Bump mapping*** process, during which we assign suitable *Render node* to material's ***Bump / Normal Mapping*** input by clicking to *Checkerboard box* next to the ***Map*** attribute.

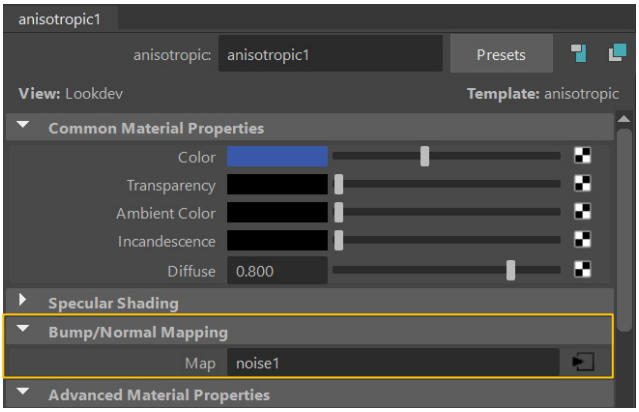


Figure 52: Noise1 driving material's Bump Map

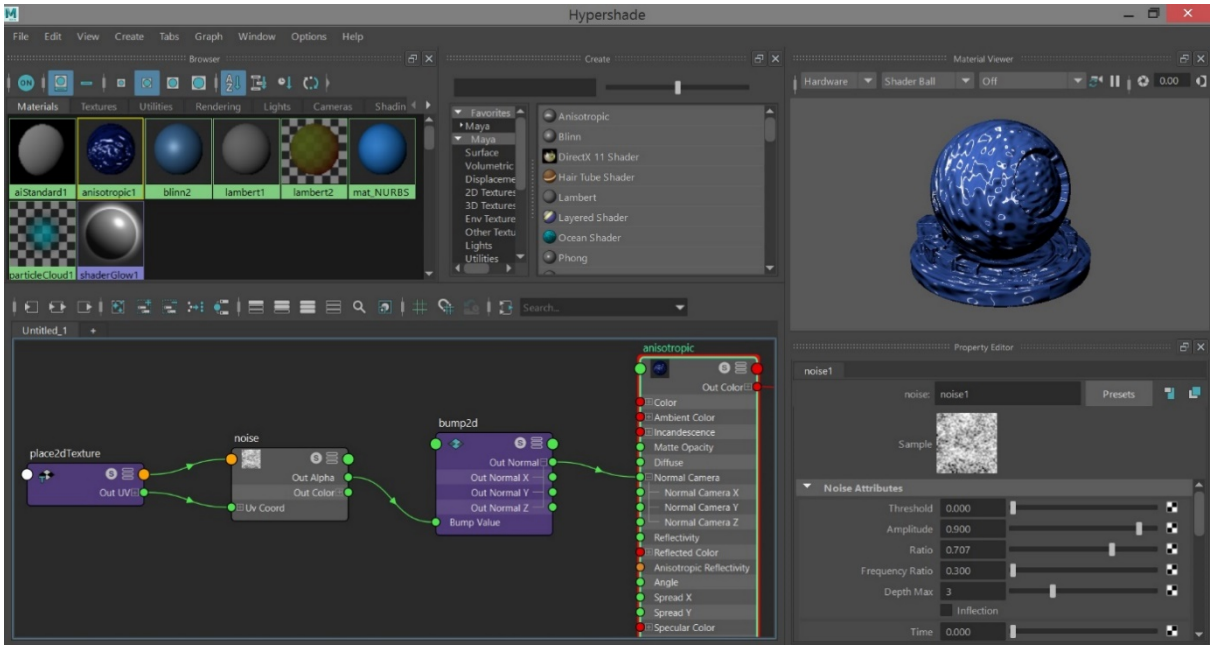


Figure 51: Shading network created automatically for Bump Mapping input node

This automatically creates appropriate *Shader network*, where we can creatively adjust selected *Render node's* attributes to suit our needs. Among multitude of available nodes, ***Noise*** seems to be most versatile one for 2D surfaces, while ***Fractal*** is convenient for semi-transparent volumetric (3D) entities, but may be utilized for 2D mapping as well.

Additional nodes can be created to further adjust overall *Bump Map* appearance, so whole network may be farther more complex and almost unmanageable within *Attribute Editor*. Other material's attributes inputs can be also attached to various *Render nodes*, as long as they are fed by appropriate numerical values. ***Color*** attribute, for example, may be controlled by a *Ramp* node (color gradient) to produce many color shades within a single Material.

2.3.6 Bump mapping vs. Displacement map

Adjusting geometrical shape's surface via ***Bump Mapping*** is convenient either for objects far away from camera, or reasonably small ones, as "bumpy" appearance is achieved by flat grayscale textures establishing physical relief illusion. Closer objects thus may utilize *Bump mapping* conveniently only for very shallow reliefs.

In contrast, ***Displacement Maps*** actually change object's geometry (move vertices), so they look reliably, when close to the camera, and also cast shadows, but require significantly more computing time than *Bump maps*, therefore should be used only where appropriate. Furthermore, they may be used only with software renderers currently, so when intending utilize MAYA *Hardware Renderer* for rendering final images, *Bump Maps* are only plausible replacement.

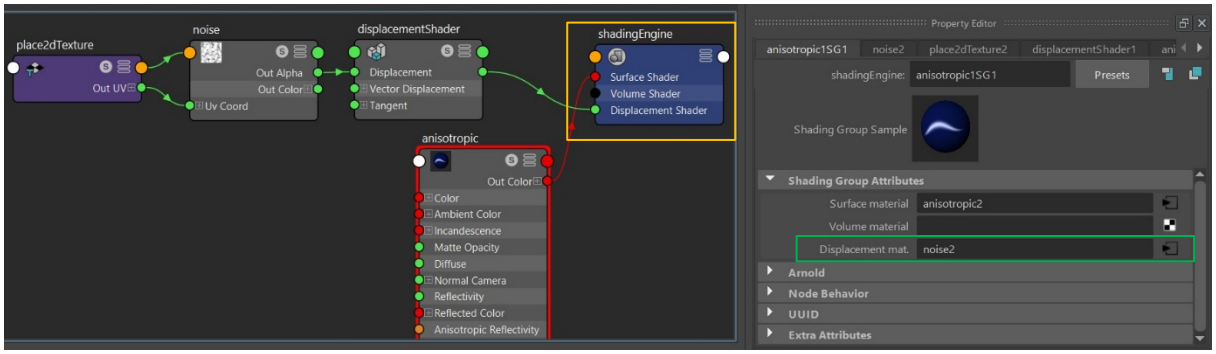


Figure 53: Displacement mapping is accessible from shading-Engine node

Displacement mapping can be accessed via material's *shading-Engine* node, which is created after assigning the material to any geometrical object, under *Shading Group Attributes* tab. As inputting displacement maps are grayscale textures, 2D and 3D maps may be used, as well as bitmap images (formats without an *Alpha Channel* require enabling *Alpha is Luminance* in the *File texture's* node within *Color Balance* section).

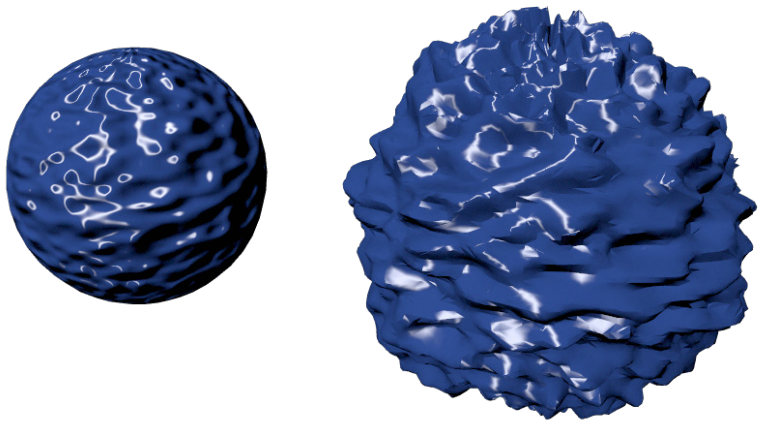


Figure 54: Bump Mapping (left) vs. Displacement Map (right)

2.4 Cameras

Real-world motion-picture cameras are constructed to capture actual events and actions to sequence of still pictures, which, if rapidly played, gives a continuous-movement illusion. **Virtual cameras**, utilized for capturing CGI animations, allow us to overcome real-world cameras physical limitations, making possible practically infeasible camera movements and establishing expensive ones (crane, or aerial shots) quite easily.

To appear reliable, however, they need to be properly set and wisely utilized, especially when **composing** rendered images with real-world footage afterwards.

Establishing appropriate spatial relationships among objects, for capturing individual images properly, is being achieved by *composing* them within a 3D space, but how these relationships appear in final images can be managed by ***Renderable Camera*** settings, discussed within following chapters.

2.4.1 Camera Sequencer

When rendering final images within MAYA interface, only one selected camera can enter rendering process at the time. There may be times, however, when creating multiple cameras with different settings, and switching among them during animation, is more easily manageable, then covering everything with a single camera.

Camera Sequencer, accessible via *Windows → Animation Editors → Camera Sequencer*, allows establishing *Shots* from individual cameras and sequencing them afterwards. Upper numbers correspond to actual *animation time*, within main GUI timeline ("real-time" frames), whereas lower ones represent a *sequence time*, allowing to use same shots repeatedly, which should be avoided, when intending to turn finished *sequence* into an ***Ubercam*** ("master camera" establishing all individual shots by appropriately *keyframing Ubercam's* attributes based on sequenced shots). Since sequence cannot be directly rendered in a production quality, setting-up *Ubercam is* the final step within *Camera Sequencer*.

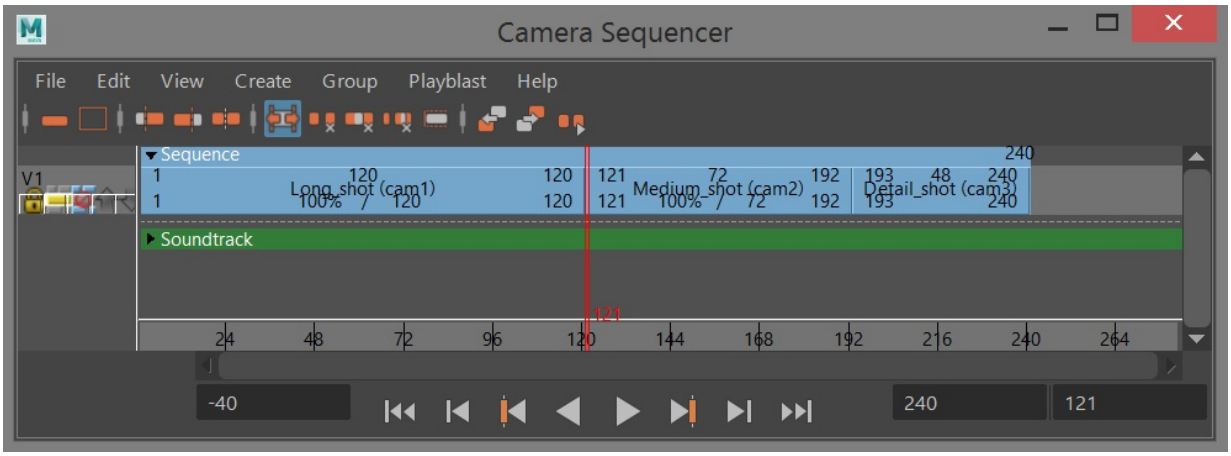


Figure 55: Sequencing different cameras into a single clip

2.4.2 Simulating real-world camera

After creating a *Camera* via *Create → Cameras → Camera*, we obtain an object simulating behavior of Single-reflex camera with 35 mm lens. Changing ***Focal Length*** will affect scene's objects appearance, but also *Field of View* (adjustable by *Angle of View* attribute) affecting how large part of scene will be visible within resulting image's frame.

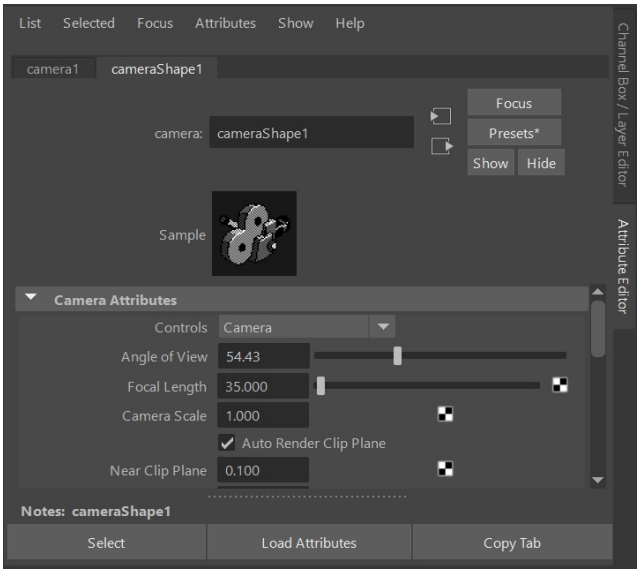


Figure 56: Standard camera attributes

To simulate specific real-world camera type behavior, we need to find out physical dimensions of particular camera's **sensor size** for recording format aspect-ratio, known as **Effective sensor size**. The information may be obtained from product's datasheet, or via internet forums, when not publicly disclosed by a manufacturer. Since usually presented in metric physical units, these values can be subsequently pasted into ***Camera Aperture (mm)*** attribute under *Film Back* section of *CameraShape* settings, automatically adjusting other attributes.

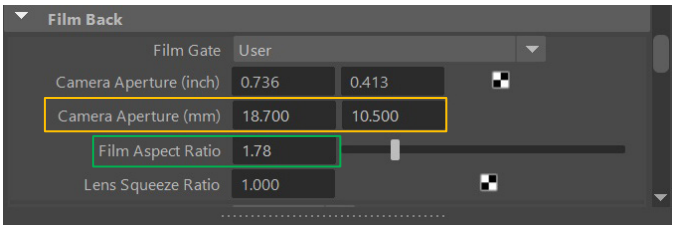


Figure 57 Film back allows simulating real-world camera

Adjusting ***Width*** and ***Height*** attributes, via *Render Settings* window inside *Image Size* section, to match physical camera's image size, used for capturing real-world scenes into digital shots, should automatically provide ***Device aspect ratio*** value identical to ***Film aspect ratio*** within virtual camera's *Film Back* section, re-assuring us that correct numbers have been pasted into *Camera Aperture* attribute.

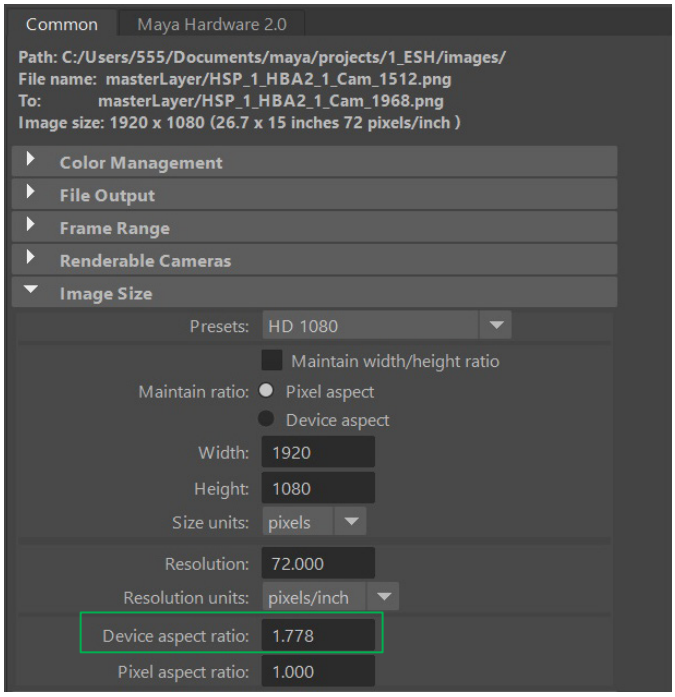


Figure 58: Device aspect ratio should match Film Aspect Ratio

2.4.3 Finalizing camera setup

Last step is typing physical camera's lens *Focal Length*, used when shooting real-world footage, into corresponding virtual camera's attribute. Together with previously discussed adjustments, this setup allows **integrating rendered images into real-world camera shots reliably**, although look of specific lens in *Out of focus* areas (*Bokeh*) may be further customized when utilizing *Arnold Renderer* by providing other aperture-related information (Size, Number of blades, Blade's curvature, Aperture's rotation).

During CGI creation process, camera can be positioned as any other object within MAYA. Considering that scenes may be created on various *Scales*, depending on particular production needs, Camera's *Locator* icon may appear either too big, or small, for working with it comfortably. Since changing its ***Scale*** attributes also affect camera's behavior, it is usually wise to keep them at default values. Adjusting ***Locator Scale*** attribute instead, will change only icon's appearance, while retaining other attributes at their proper values.

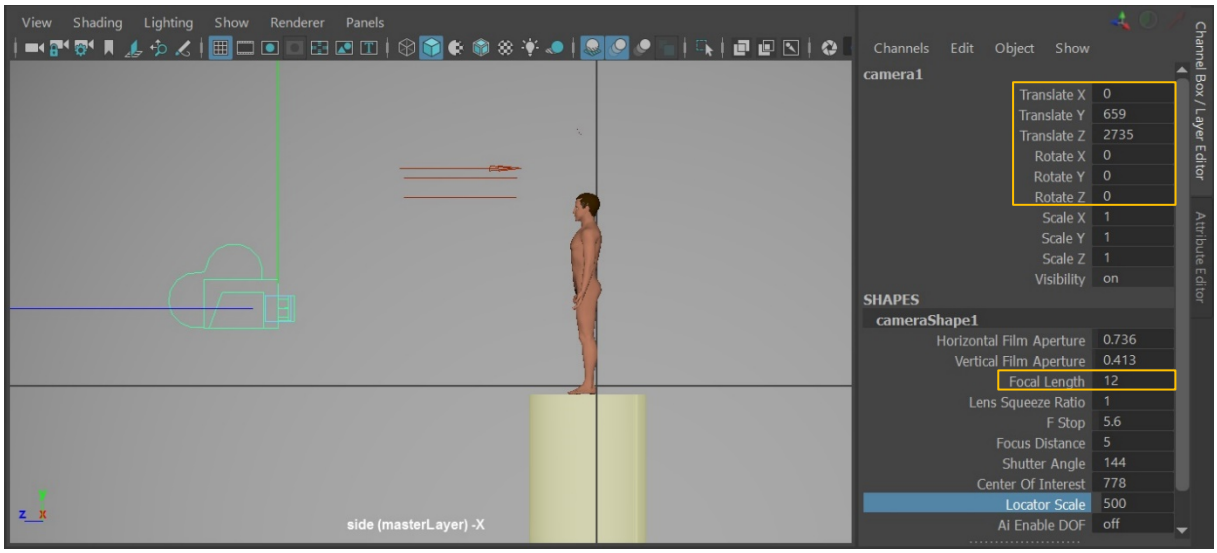


Figure 59: Orthographic View allows position camera precisely

Best practice for composing shot is positioning camera roughly with MAYA's standard tools, along with *Camera Fly Tool*, and fine-tuning corresponding attributes subsequently within *Channel Box*. Since camera may unintentionally tilt while using *Rotate Tool*, setting ***Axis Orientation*** to ***Gimbal*** mode within *Tool Settings* window, along with changing *Rotate Order* inside our Camera's main node via *Attribute Editor*, should prevent weird rotations and camera angles, especially when animating.

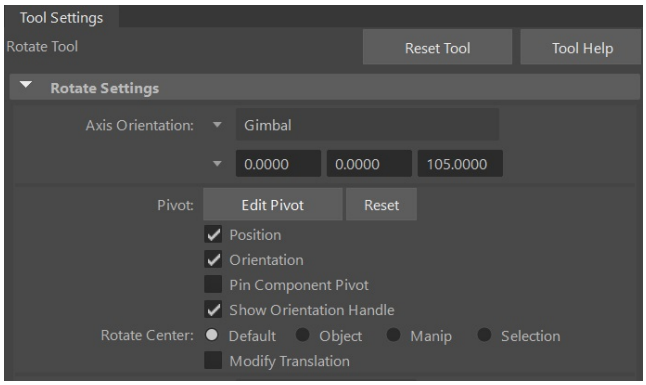


Figure 60: Gimbal mode prevents weird rotations

2.4.4 Resolution Gate

Viewport's aspect ratio is rather fluid, as we can adjust its size and re-arrange multiple *Viewports* as needed. Thus, what is visible within *Viewport's* window does not dimensionally correspond to rendered image's framing. By enabling ***Resolution Gate*** by one of *Viewport's* menu icons, we can accurately see portion, which is going to be rendered with our current *Render Settings.* Left to *Resolution Gate's* icon, there is also a *Film Gate* icon, showing which portion of the scene is seen by virtual camera's sensor. When camera's and rendering setting are properly adjusted, both gates should be identical (having same *Aspect ratio*).

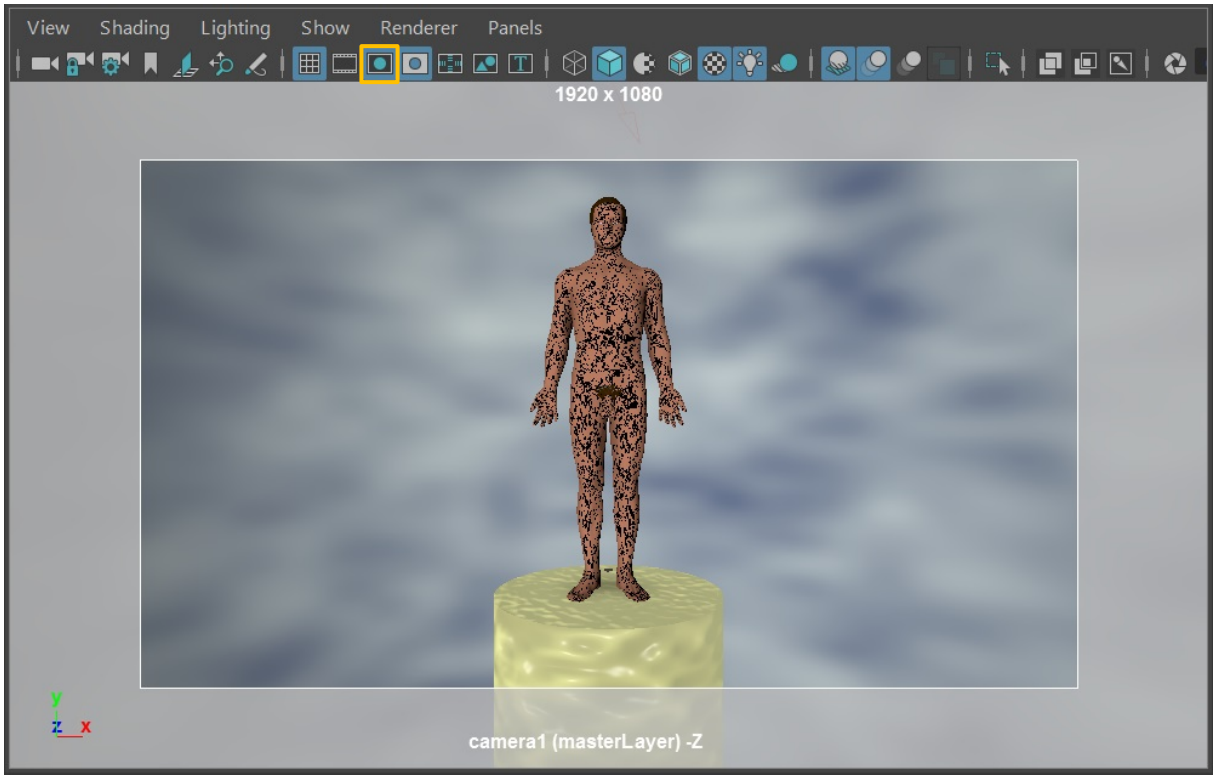


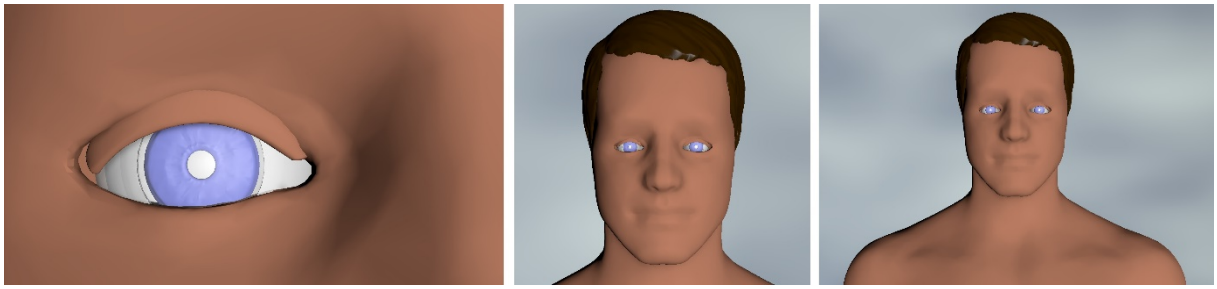
Figure 61: Resolution Gate allows establishing composition accurately

Before positioning any camera within available *Viewports*, displaying *Resolution Gate* is an essential step for establishing final image's composition accurately.

2.4.5 Shot sizes

A single *shot* can be considered as a **sentence** translated into **Visual language**, which advantage is its natural understandability by all human beings. Sequencing multiple *shots*, allows us to create paragraphs and consequently a whole book, telling some interesting story and eventually conveying an important information. But writing any "Visual book" requires learning grammar and semantics of the language we are currently using, which starts by knowing the Alphabet, represented here by well-settled **shot (field) sizes.**

Extreme Long Shot (XLS) | Long Shot (LS) | Medium Shot (MS)



Extreme Close-Up (XCU) | Close-Up (CU) | Medium Close-Up (MCU)

Figure 62: Conventional shot sizes

Framing subject by different sizes allows conveying different types of information, as each one is suitable for different purpose. ***Extreme Long Shot*** shows our subject(s) as a tiny part of big world, being barely visible, or completely hidden, so audience gets only contextual information, **where** following action is going to take place (establishing scene's *diegesis*).

Long shot shows the subject(s) **in context with surrounding environment**, where subject's individual features (e.g. body parts) are barely distinguishable. This makes it suitable for showing *character's* overall actions towards his surroundings, clarifying what is actually happening at the moment. When used in latter phase of events, it may provide answers satisfying audience curiosity built during previous action (releasing tension).

One of these two types is usually selected as a first shot of entire scene, *establishing* **diegetic context** for following events, in which case we call it an ***Establishing shot*** in daily practice.

Medium Shot reveals character's **attitude** towards its neighbor surroundings. Gestures can be seen very clearly, as well as gross face expressions.

Medium Close-Up and ***Close-Up*** are closely related, allowing to see facial expression subtleties, making clear what is happening within character **internally** (looking inside his mind).

Extreme Close-Up heavily focuses on one particular feature, as being most crucial component determining **results** of action taking place, or showing subject's real nature (looking into the soul), revealing its true **essence**.

As can be seen, first three shots show character in **context** to its surrounding environment, or how they **interact towards each other**, while last three shots (Close-anything) reveals what is happening **internally** within character (audience can see character's **Point of View**).

There are also other shot sizes, but they are usually used only for specific situations, whereas discussed ones form basis of any motion picture camera composition decisions, and furthermore can also shape **complex shots** when combined together. However, real power of well-settled basic shot sizes resides in simple truth that by utilizing them thoughtfully, **every story can be conveyed** to the audience effectively.

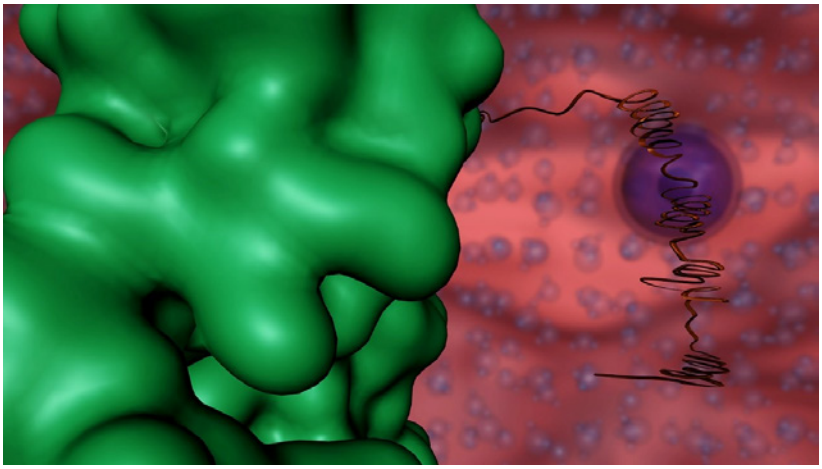


Figure 63: Complex shot featuring unfolded protein (Long Shot) and HSP70 blocking frame partially (Close-Up)

2.4.6 Composing single shot

Within MAYA environment, moving established camera to appropriate **distance** from a subject and setting its **Focal Length** properly is essential for creating desired composition of resulting images, similarly to real-world cinematography.

Although virtually any *Focal Length* may be utilized for any shot size, some are usually more suitable than others, as changing *Focal Length* distorts **Perspective** within an image, affecting audience's perception of its depth.

Standard 50mm lens simulates natural human eye vision, being basic general-purpose ones, most often utilized within Medium-sized shots range.

Telephoto 85mm lens are commonly used for Close-Up and "Portrait" types of shots, as they flatten the image (flatter the subject), effectively smearing skin imperfections until they become unnoticeable at a first glance. This also causes blurring fine details, so if they are essential for the shot, lower *Focal Length* may be used, but camera needs to move much closer to the object of interest.

Telephoto lens narrows **Depth of Field** (perfectly focused area, while blurring everything else with increasing distance in both directions – towards the camera and away from it), but also squeezes image perspective, so all objects appear closer to each other, than are in reality, therefore *Focal Lengths* above 100 mm may be creatively used for overcoming huge distances.

Going opposite way to **35mm and 24mm Wide-angle lenses**, these are effectively stretching the perspective, making objects appear in greater distances from each other, than they actually are. Fine details are much more noticeable, but any imperfections are exaggerated as well, so utilizing such *Focal Lengths* for capturing Close-Up shots should be considered carefully in a particular situation.

Further down to **16mm** and even **8mm Fish-Eye Ultra-wide lenses**, straight lines become more and more curved up to the hemisphere shape, which may be creatively utilized for simulating non-human Point of View.

Table 8: Selecting appropriate type of Lens for common purposes

Lens type	Focal length	Shot Size	Purpose
Telephoto	800 - 85	Close-Up	Facial expressions
Standard	50	Medium Shot	Attitude / Interaction
Wide-angle	35 - 24	Long Shot	Scene context
Fish-eye	16 - 8	Non-Human Shot	Exotic PoV

2.5 Lighting

When we can see anything at all within *Viewport* and *Render View* windows after creating *New scene*, it is due to the ***Default Lighting*** rays bounced off created objects towards default *Virtual camera's* imaginative light sensor.

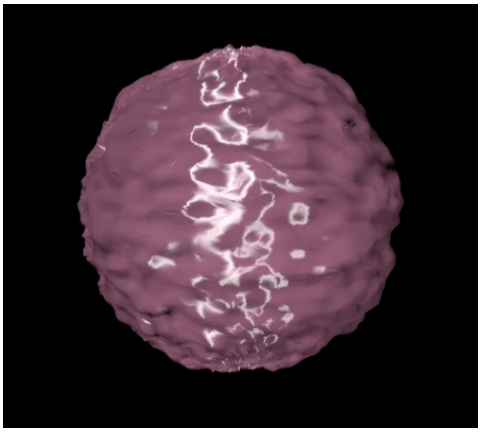


Figure 64: Tissue-analog sphere illuminated by Default Lighting

This works with ***Maya Hardware 2.0*** renderer, which renders everything within *Viewports*, and also ***Maya Software*** renderer. If using other renderers (e.g. *Arnold Renderer*), scene may appear **black**, due to the insufficient number of photons provided by default lighting setup, as different renderers handle lights differently and may even require their own sources of illumination.

2.5.1 Standard Light Attributes

After creating any light source, it is usually wise to adjust its parameters to suit our needs:

Decay rate determines light's fall-off with increasing distance, where *No Decay* means that light intensity remains the same at any distance from light source, whereas ***Quadratic*** option behaves in physically accurate manner.

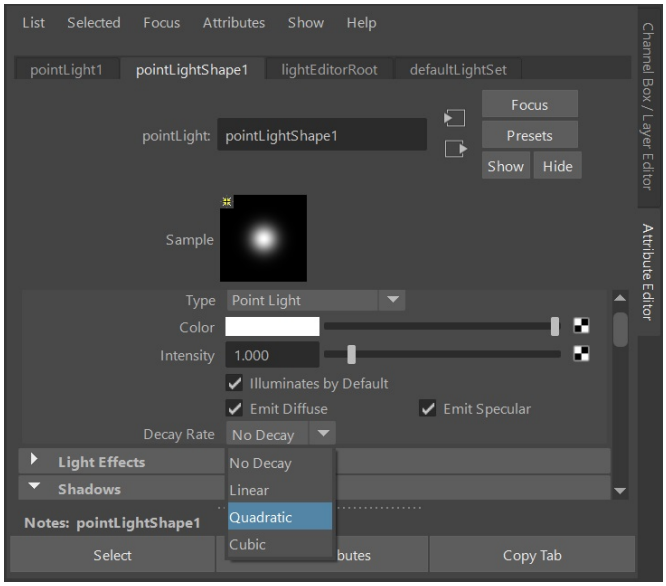


Figure 65: Virtual Lights allow overcoming physical boundaries of real-world lighting

Intensity allows regulating how much light this particular source introduces into the scene. Since different renderers handle light differently, entering sometimes even 1,000x bigger values may be needed for achieving similar results. ***Color*** can provide **color-cast** useful when treating various light sources differently (achieving different effects for illuminated objects). Main source of illumination, however, should be white, as rendered animation can be color graded during post-production.

In comparison to real-world Lights, within CGI realm we may disable various lighting-related features (Shadows, Specular Highlights, or providing only these) by corresponding **check-boxes**. We can also choose, whether *Shadows* will be rendered as a ***Depth Map*** (simple bitmap), or ***Raytraced*** (physically accurate, but render-time heavy).

2.5.2 Available lights

Amb Dir Point Spot Area Vol

Figure 66: Available MAYA standard Lights

Custom lighting may be achieved by thoughtfully utilizing various ***Lights*** available within MAYA. ***Ambient Light*** (Amb) provides even light distribution in all directions, making scene look very flat, although in addition to **direct light** (Light-rays hitting objects directly), **indirect lighting** (objects illuminated by rays bounced off other objects) is calculated as well. ***Directional Light*** (Dir) shines evenly only in one direction, simulating distant light sources, such as the Sun. This is also *Default Lighting* source provided by MAYA, when no Light has been set manually.

Point Light is isotropic source of illumination, shining evenly in all directions like a real-world light bulb does. ***Spot light*** provides even lighting within narrow directional range, defined by a *Cone Angle.* This type of *Light* represents **Fresnel Lights** commonly utilized in movie production. ***Area Light*** is most physically accurate, organic-material friendly light source, behaving as studio **Soft-box Light**, providing very soft shadows and high-quality illumination at cost of long rendering times.

Volume Light (Vol) distributes light only within specific area, therefore can be utilized for achieving various effects. Contrary to physically-based Lights, this one may provide **negative illumination**, acting as a “darkness generator”.

Table 9: Assorting MAYA standard Lights

Light source	Directions	Real-world Equivalent	Typical use
Ambient	All	-	Cartoonization
Directional	One	Sun	Default Lighting
Point	All	Light Bulb	Flattening image Visible light source (candle)
Spot	Narrow range	Fresnel Lantern	Strongly lighting defined area
Area	Wide range	Studio Soft-box	“Showroom” lighting Soft natural illumination
Volume	Customizable	-	Visual effects

2.5.3 Basic illumination & Aesthetic utilization

Default lighting setup actually provides a **Basic illumination** to the scene, so there are enough photons for *illuminating* objects properly within **full dynamic range** (darkest pixel within rendered picture being pure black, brightest one pure white).

And in many situations, this can be sufficient for resulting animation serving its purpose, therefore rendering scene with *Maya Hardware / Software* renderer is possible without introducing additional *Lights* into the scene.

Going beyond *Basic illumination*, custom lighting setup may be utilized to aesthetically create **contrast** among different objects (dark object next to light one), increase / decrease **depth perception** ("layering" well-illuminated and shadowy areas along Z-axis), or **(mis)direct** audience's **attention** (bright area within dark image).

Creative Lighting is also primary method for **establishing visual subtext** within the scene, as it can establish an **atmosphere** (e.g. dark scene enhancing dramatic situations, bright scene enhancing cheerful moments), reveal character's **true nature** (white / light = good, black/ dark = bad, half-light-half-dark = mixed / complex character), **state of mind** (red cast = loving / angry, green cast = natural / sick, blue cast = calm / cold), or **relationship** to another one (character stepping into hard light = dominant, another receding into shadow simultaneously = recessive).

Some of these results may be also achieved, to some extent, in a quick & dirty way within dedicated composing application by color grading and visual effect (VFX) approaches, after *Basically illuminated* images have been rendered. Preferred method depends on production needs in particular situation and available resources, while the proper way provides significantly better fidelity at cost of little bit more qualified work.

Table 10: Achieving various purposes by different lighting

Purpose	Example	Light
Basic illumination	Balanced clip for post-production	Default / Directional
Adjusting contrast	Dark object next to light object	Spot / Area
Depth perception	Interlacing bright and dark areas	Directional / Spot
Directing attention	Bright area within dark image	Point
Establishing atmosphere	Dark scene for dramatic situation	Directional
Exposing true nature	Light object = good, Dark object = bad	Area + Spot
Revealing state of mind	Red cast = angry, Blue cast = calm	Spot
Displaying relationships	Light = dominant, Shadow = recessive	Volume

2.5.4 Three-point lighting

In Photography and conventional Cinematography, we usually use three lights, at least, for illuminating subject properly. This setup, known as **Three-point Lighting**, consists of three different light sources complementing each other.

Key Light (Sun / Fresnel Light) provides usually Basic illumination to the scene, but since being usually quite a strong source, it also creates undesirable shadows. Therefore, we introduce **Fill Light** from lower position, shining upwards reasonably, lightening shadowy areas. It may also provide slightly warm (yellow) color cast, making living-being related materials more visually appealing. In any case, this Light should be very soft, so lighting indirectly (reflective surface, soft-box) is a common practice. Since **Back Light**, being located behind subject, illuminates only subject's edges, can be referred to as a **rim** light, or a **kicker** (when partially illuminating object's surface). Its main purpose is separating subject from a background.

By adding fourth Light, illuminating background separately, we obtain **Four-point Lighting** setup suitable for creating **Green-screen** shots, as background needs to be as flat as possible in such cases.

Since within CGI realm, *Shadows*, casted by individual Lights, may be deliberately disabled. Therefore, we may use only Key and Rim Lights to achieve Three-point Lighting results in some cases, or create more sophisticated Lighting setups quite easily.

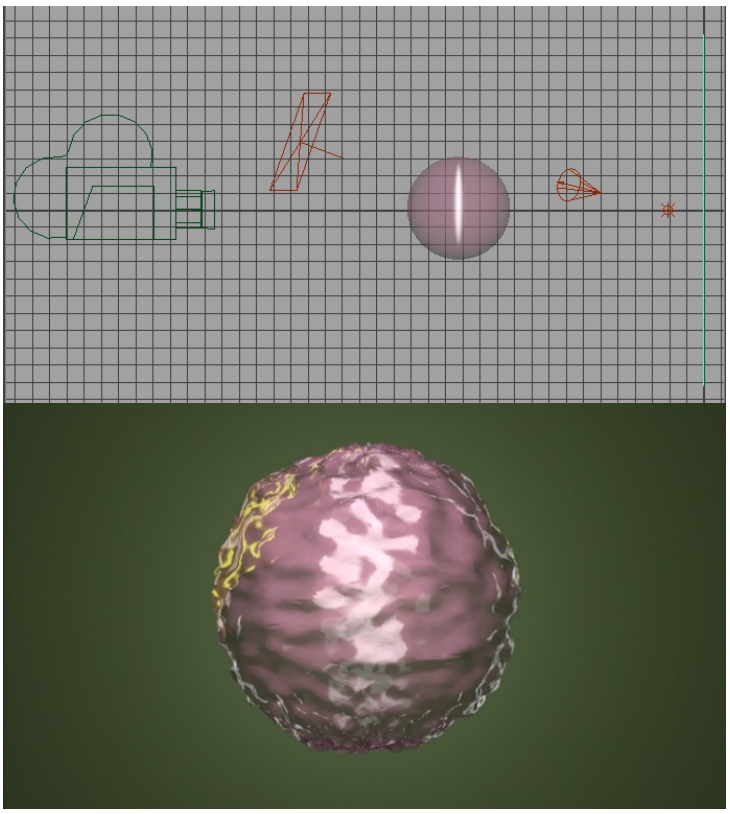


Figure 67: Four-point Lighting setup established by only three Lights

2.6 Animation approaches

Introducing **Any motion** over time is being achieved by object's *Attributes* having different values at different frames. We can see results of such changes as object's motion / transformation within *Viewport*, while interface allowing us to move throughout individual frames over time is located at the bottom of MAYA's main interface as a ***Timeline***. We can jump directly to desired frame, or ***Play(back)*** whole animation with ***Playback control*** buttons. Displaying it properly, however, requires appropriately set ***Playback Preferences***.

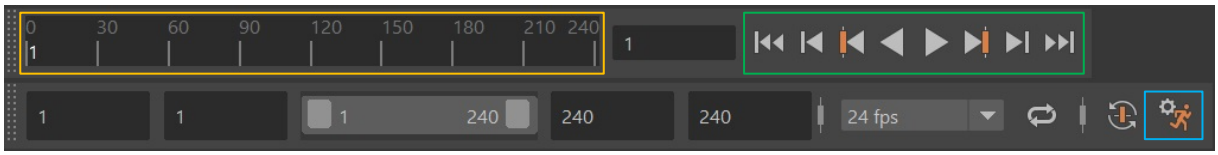


Figure 68: Viewport's Playback interface

2.6.1 Playback preferences

Within *Preferences* → ***Time Slider*** → *Playback* section, ***Playback speed*** determines how an actual animation will be presented. ***24 fps x 1*** corresponds to ***24 fps [real-time]*** option from previous versions of MAYA, meaning that 24 frames are being played over a single second, and if animation being too heavy to playback smoothly, MAYA drops some frames for maintaining overall animation speed over time, which is crucial for making timing-related decisions.

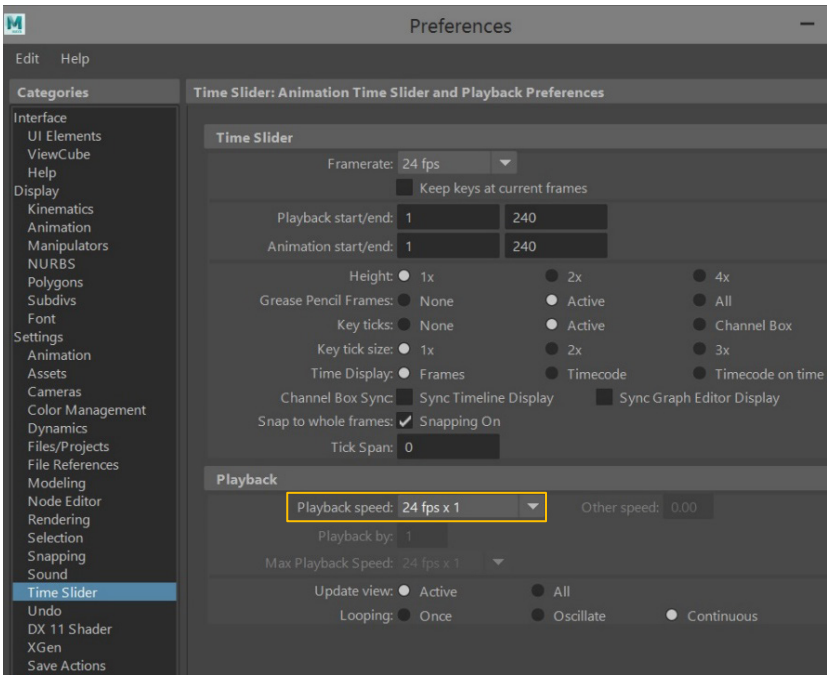


Figure 69: Playback Preferences are crucial for making timing-related decisions correctly

We may also change different *Playback speed*, or choose to ***Play every frame*** no matter how computationally heavy the animation is, for assessing its appearance on frame-by-frame basis (e.g. when evaluating *nParticle* physical *Simulation* results).

2.6.2 Keyframes

If we explicitly set specific value to some attribute and want it change to different one after some time, it is possible to set ***Keys*** at appropriate frames, between which continuous attribute change will be automatically interpolated. Adjusting this interpolation graphically may be performed within ***Graph Editor*** window.

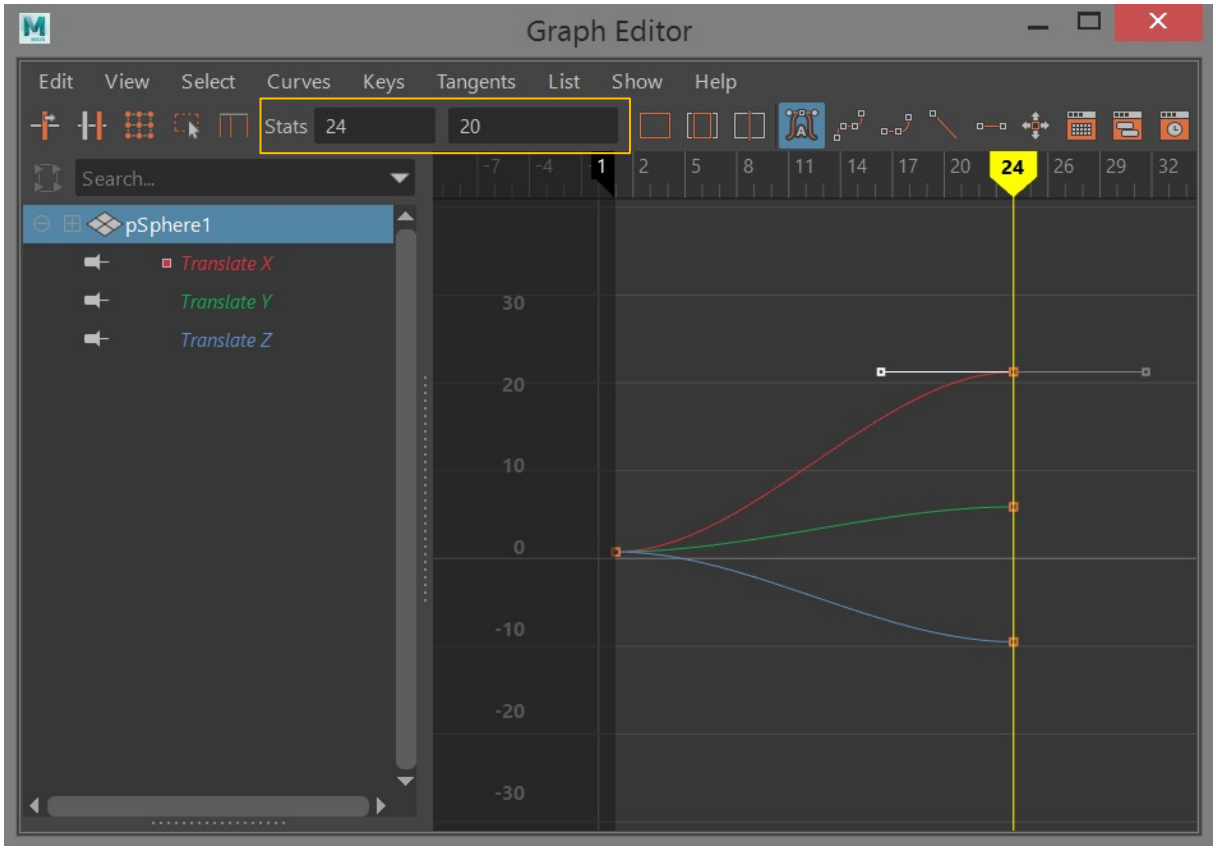


Figure 70: Graph Editor allows graphically customize interpolation between Keys

Stats boxes allow precise setting of individual ***Keyframes*** numerically, where first one corresponds to *Frame* within timeline (temporal information) and second matches object's attribute numerical value at given time.

Default ***curve's*** type is a *Bezier* organically interpolating movement in-between two frames to appear natural, accelerating it when movement begins and decelerating when reaching second *Keyframe* (by slowing down attribute's value change within *Keyframe's* proximity). Further customization is possible via handling ***Bezier Handles*** after selecting a single *Key*. Since it automatically sets *curve's* tangent when adjusting individual Keys, this interpolation mode is being referred to as ***Auto tangents***.

In some cases, however, different behavior may be more desirable (*Linear tangents* when cycling mechanical movements), or necessary (*Step tangent* for changing object's visibility on / off). Cycling established animation can be achieved by *Curves* → *Post Infinity* → ***Cycle*** or ***Cycle with offset***, depending on particular animation's behavior.

2.6.3 Motion Path

Although we may develop animation with object traveling through particular route by manually *Key-framing* important points of its path, for precisely controlled movement along more complex *Paths*, utilizing ***Motion Path*** constrain is significantly more time-effective.

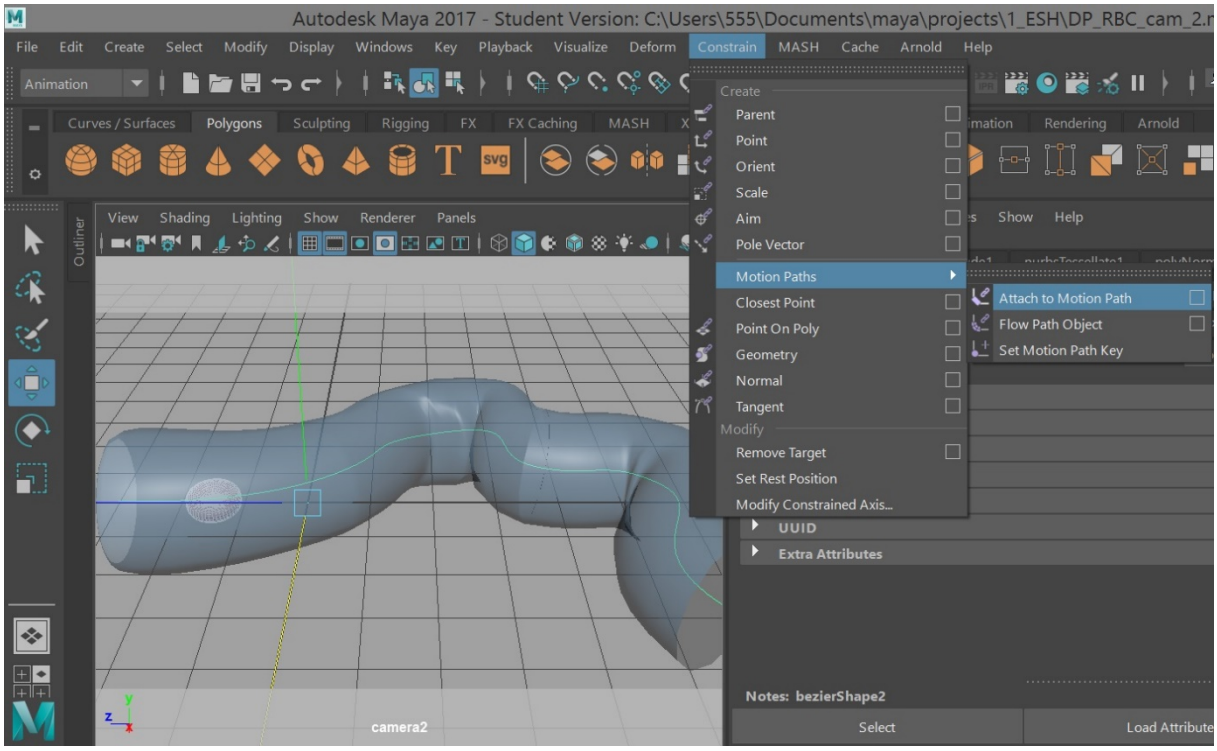


Figure 71: Attaching Red blood cell to Motion Path

Motion Path is usually represented by a NURBS curve, while visible path being actually an *Extruded Surface*. Constraining object's movement to a particular path can be achieved, after selecting the ***Object*** and the ***Curve***, by *Constrain → Motion Path → Attach to Motion Path [box]*.

Within its *Options* window, we can set ***Time range*** of an animation (how much time object spends by traveling from beginning to the end) and also object's orientation towards the actual *Path* by setting ***Front axis*** and ***Up axis*** appropriately.

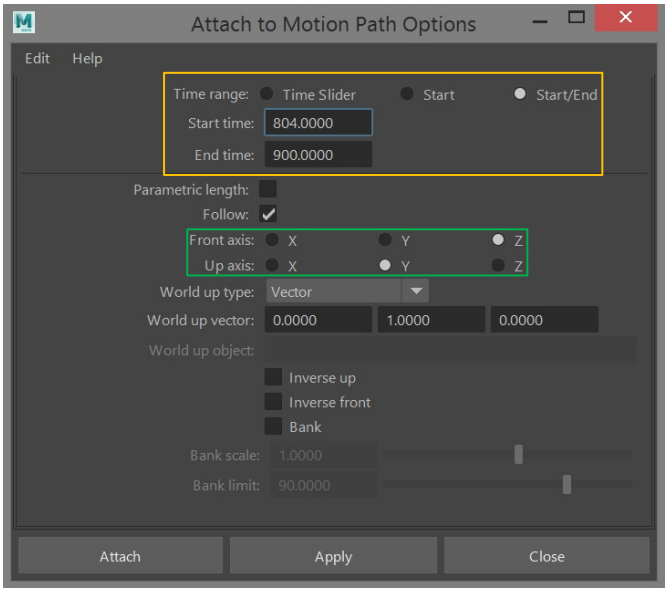


Figure 72: Setting Motion Path attributes to obtain expected animation results

2.6.4 Expressions

Binding different attributes together, so that adjusting one attribute's value causes defined change of the other attribute, may be achieved by establishing an ***Expression*** between these two attributes. We can then control animation by Key-framing "master" attribute, as driven one simply follows the rules established by the *Expression.* This way, we may rig various parts of complex machinery, ranging from car engine to ATP-Synthase, and subsequently animate whole system easily with very few *Keyframes*.

Establishing *Expression* can be performed within *Windows* → *Animation Editors* → ***Expression Editor***.

In the ***Expression*** section, driven *attribute* (*translateY* of *Piston* object) on left side is being controlled by driving attribute (*rotateZ* of *Crankshaft* object) on the right side.

Since *translate* attributes are in length units (mm) and *rotate* in degrees (°), dividing *rotateZ* by 360 causes *Piston* to move up by 1 unit for each full rotation of *Crankshaft*.

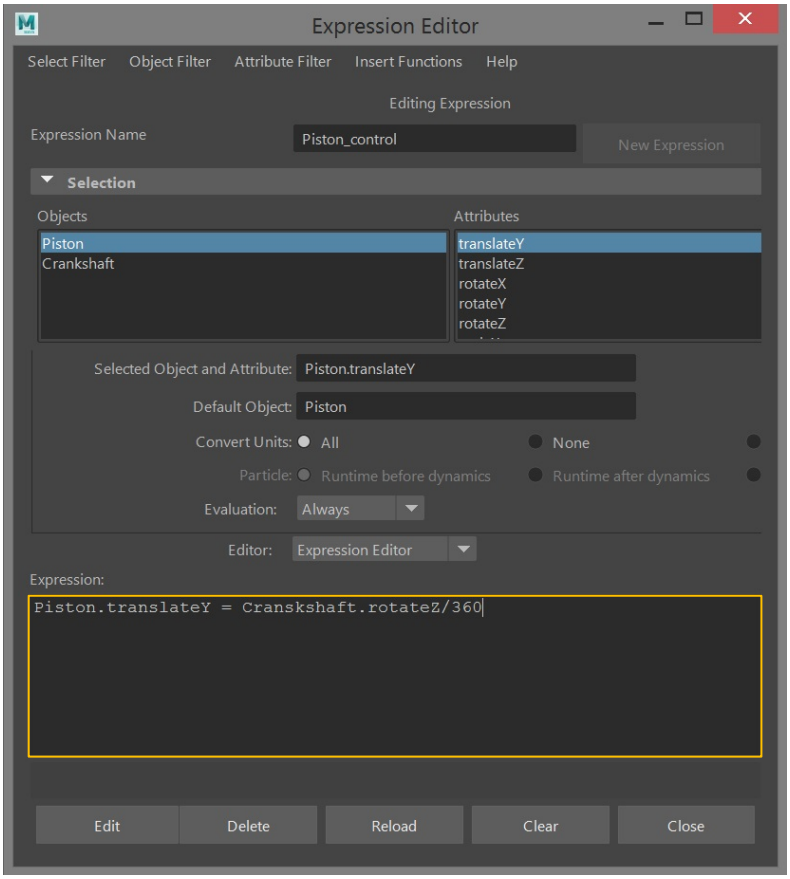


Figure 73: Expression Editor allows to establish Expression between selected attributes

For moving *Piston* up and suddenly falling down to its initial position, after full rotation of *Crankshaft* is completed, it one possible solution may be **subtracting** integer portion of *translateY* attribute by *trunc* function (returning selected parameter's integral value), so the resulting *Expression* would be:

Piston.translateY = ((Crankshaft.rotateZ/360) – trunc(Crankshaft.rotateZ/360))

After completing our *Expression*, and naming it conveniently, we can establish it by clicking on ***Create*** button (or ***Edit*** when adjusting already established one) and ***Close*** the *Expression Editor*.

2.6.5 Physically based Simulations

Traditional *Keyframing* is appropriate for handling individual objects. But when we need to prepare a *group* containing hundreds, or thousands similar objects, interacting with each other and their environment in physically accurate manner, ***nParticle Simulation*** may create such complex animation by rules we set to approximate object's real-world behavior.

This tool bas been developed to generate and manage *numerous Particles* based effects and regulate particle's behavior by physical *Fields* and *Forces* acting upon them. Furthermore, we can assign geometrical objects to generated particles via an ***Instancer***, so instead of "dots", actual geometry is visible, acting according to previously established rules.

Therefore, to create, for example, Blood cells flowing through a Blood vessel, first we need to prepare an actual geometry, in this case **Blood vessel** (*Extruded NURBS curve*)**, Red blood cell** (*Re-shaped polygonal sphere*) and **White blood cell** (*Polygonal sphere with a Displacement map*) resembling shapes. After our geometry models are finished, and eventually shaded, we may proceed to an actual simulation.

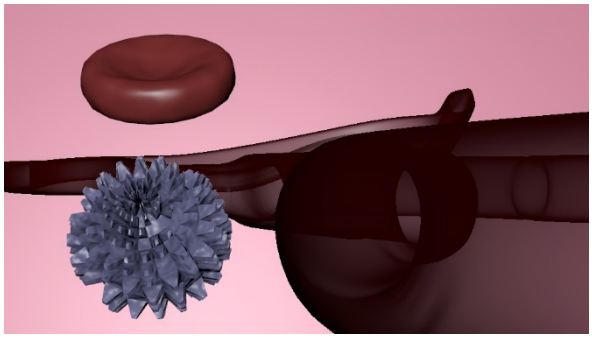


Figure 74: Assets should be ready before developing nParticle Simulation

2.6.6 Approximating real-world behavior

Since blood cells should be traveling throughout the Blood vessel, physical *Field* pushing them through needs to be created. We can switch menu-set to *FX* (or *Dynamics*) and after selecting vessel's original NURBS *curve*, create a physical ***Field*** via *Fields / Solvers* → ***Volume curve***.

Source *emitting* particles can be created via *nParticles* → ***Create Emitter*** *[box]*, where *Emitter type* allows to choose, whether particles should travel evenly, or in specific direction. After placing it to somewhere within previously created *Field,* we may link those two objects within *Windows* → *Relationship Editors* → ***Dynamic Relationships Editor*** just by clicking on both of them.

Particle's behavior can be customized through *nParticle* object via *Attribute Editor*. Under *Dynamic Properties* tab, enabling ***Ignore Solver Wind*** and ***Ignore Solver Gravity*** checkboxes allows particles "float" within the Blood vessel, being driven by applied *Field.*

Now we can proceed to setting actual rules, approximating particle's real-world behavior, by adjusting *Field's* attributes within *Attribute Editor*. ***Magnitude*** determines how strongly will be particles pushed across the *Field*, and ***Attenuation*** how much slower will be ones floating at the periphery in comparison to those traveling along central line. Under ***Volume Speed Attributes*** tab, we may adjust particle's speed in various directions relative to *Field's* axis.

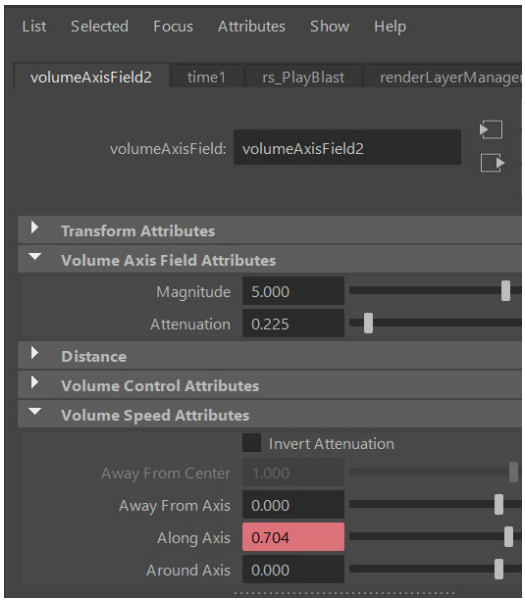


Figure 76: Essential Field's attributes

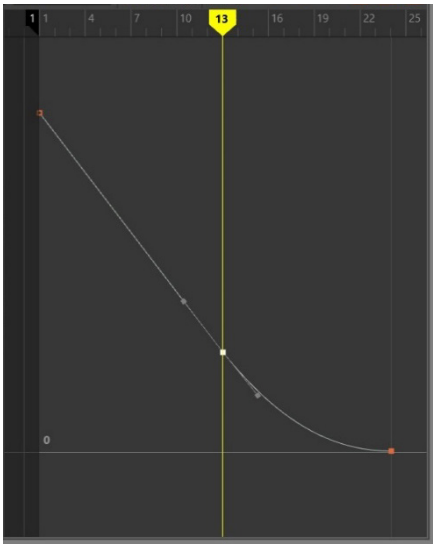


Figure 75: Pulsing blood requires only three Keyframes

By Keyframing ***Along Axis*** speed between 1 and 0 gradually slowing down, **Blood Pulse** may be approximated after ***Cycling*** it ***with Offset*** via *Graph Editor* menu.

Before previewing animation within *Viewport*, *Playback speed* needs to be set to ***Play every frame*** (max. real-time), as simulation is being evaluated on frame-by-frame basis and skipping any frame would result in unexpected behavior, so when moving to specific frame within animation, *going back to start* and proceeding frame-by-frame is crucial for displaying animation correctly.

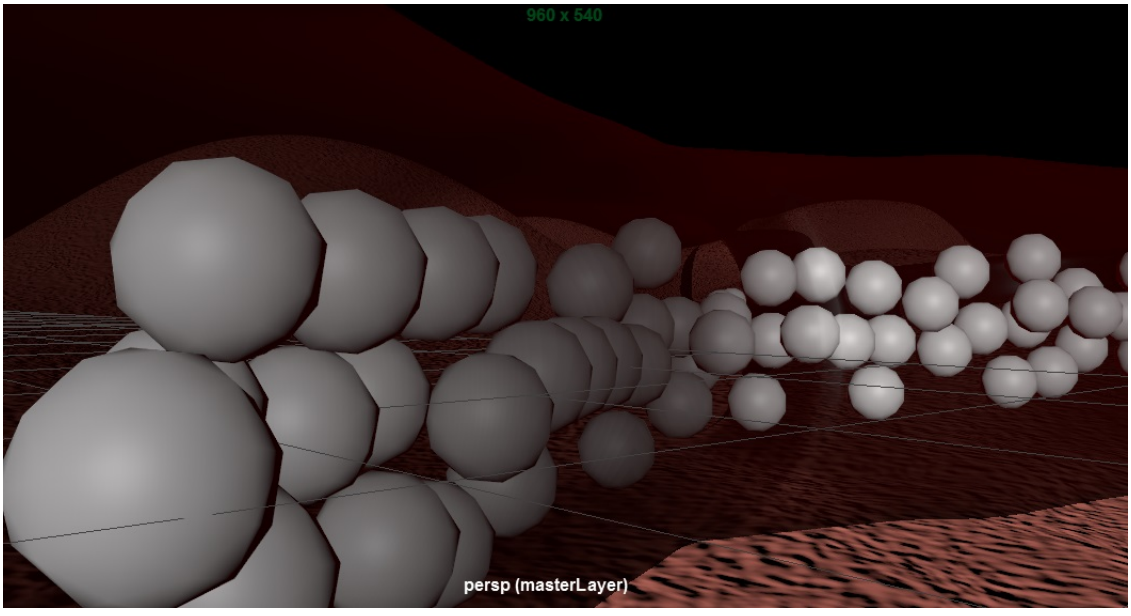


Figure 77: nParticles travelling with pulsations inside a Blood vessel

2.6.7 Instancing geometry

For utilizing our modelled Blood cells instead of default particle shapes, ***Instancer*** can conveniently assign external geometry to individual particles. After selecting desired geometry within *Outliner*, we can proceed via *nParticles* → *Instancer [box]* opened window, where geometry should be listed under *Instanced Objects*. If so, it is possible to just select *nParticle* object within *Outliner* and subsequently click on *Apply* button for assigning listed geometry to particles. If both Blood cell types are assigned properly, we can still see only 0:[Geometry] being instanced. We can use *Expression* to set ratio, in which both shapes will replace individual particles, after improving their movement fidelity.

To hide default particle shapes, it is necessary to dive into ***nParticleShape*** node via *Attribute Editor*, and under *Object Display* section, there is an unchecked *Intermediate Object* checkbox. By checking it, default shapes get hidden, but our geometry remains visible.

Although Blood cells are flowing along the vessel, we may also introduce *rotation* on per-particle basis, so each one keeps rotating slightly differently. Under *Rotation*, checking ***Compute Rotation*** checkbox enables particles to start rotating, but we also need to tell *Instancer*, how they should rotate, so under *Instancer* → *Rotation Options* → ***Rotation*** attribute offers ***rotationPP*** variant, which allows rotation being computed individually for each particle.

Now, we can replace nParticles by instanced geometry in physiologically accurate manner. For doing so, first step is creating a custom attribute under *Add Dynamic Attributes* by clicking on ***General*** button. Since this new attribute will assign indexing number to each particle, we may set *Long name* as ***IndexPP***, or any convenient name. *Data Type* needs to be a ***Vector***, as we will set *Attribute Type* to ***Per Particle.***

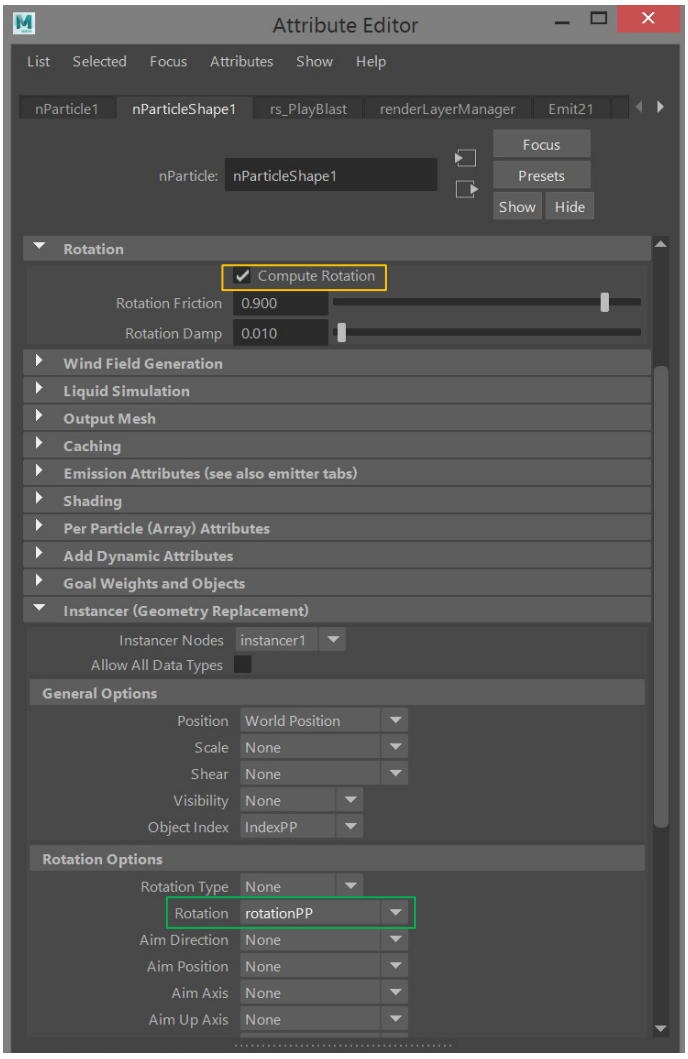


Figure 78: Rotation Per-Particle enables computing rotation for each particle individually

After creating an Instancer with ***OK*** button, we may find it under *Per Particle (Array) Attributes.* By right-mouse-button clicking within its value box, ***Creation Expression*** window can be invoked. After selecting ***nParticleShape.IndexPP*** attribute, it can be copied into clipboard for establishing an *Expression*. But before assigning indexing numbers to individual particles, we generate random numbers and put them into a new variable. For generating random integer number starting from "0" up to "999", this may be written as:

Int $i = rand(0, 1000)

Then we can set condition telling that if this generated number is greater than "0", geometry assigned to currently computed particle will be Red blood cell. And as this one corresponds to 0:[Red blood cell] within Instancer section of *nParticleShape* node, we may write whole condition as:

if($i > 0)
nParticleShape.IndexPP = 0;

But if this number will be exactly "0", then we want to use a White blood cell geometry. Since this one corresponds to 1:[White blood cell], the condition will look like:

if($i = 0)
nParticleShape.IndexPP = 1;

Then we can confirm *Expression* creation by clicking on ***Create*** button.

This setup would assign White blood cell geometry with 0.1% probability corresponding to the natural occurrence of this Blood cell type within human blood plasma. Depending on other settings (*Particle emission rate, max. number of particles*), this ratio could be too low to see even single White blood cell over course of entire simulation, therefore we may *Edit* our expression, provide number suiting our needs (hereby using an **Artistic License**), and confirm it by clicking on ***Edit*** button.

The last, but most important, step after fine-tuning a *Simulation* is ***Baking an Instancer.*** Without it, sending whole scene to *Batch Render* images would be extremely slow, as each frame being currently rendered requires computing simulation in all previous frames over and over again, therefore we may convert instanced objects into densely *Keyframed* geometry via *MASH* → *Utilities* → ***Bake Instancer to objects*** → ***Bake Animation*** button.

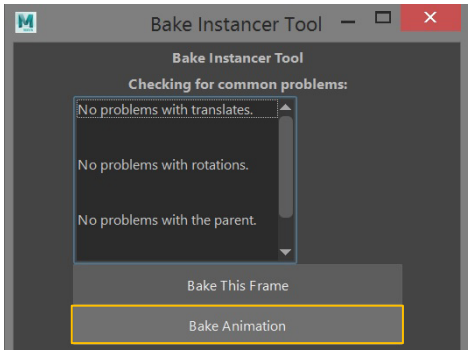


Figure 79: Baking an Instancer

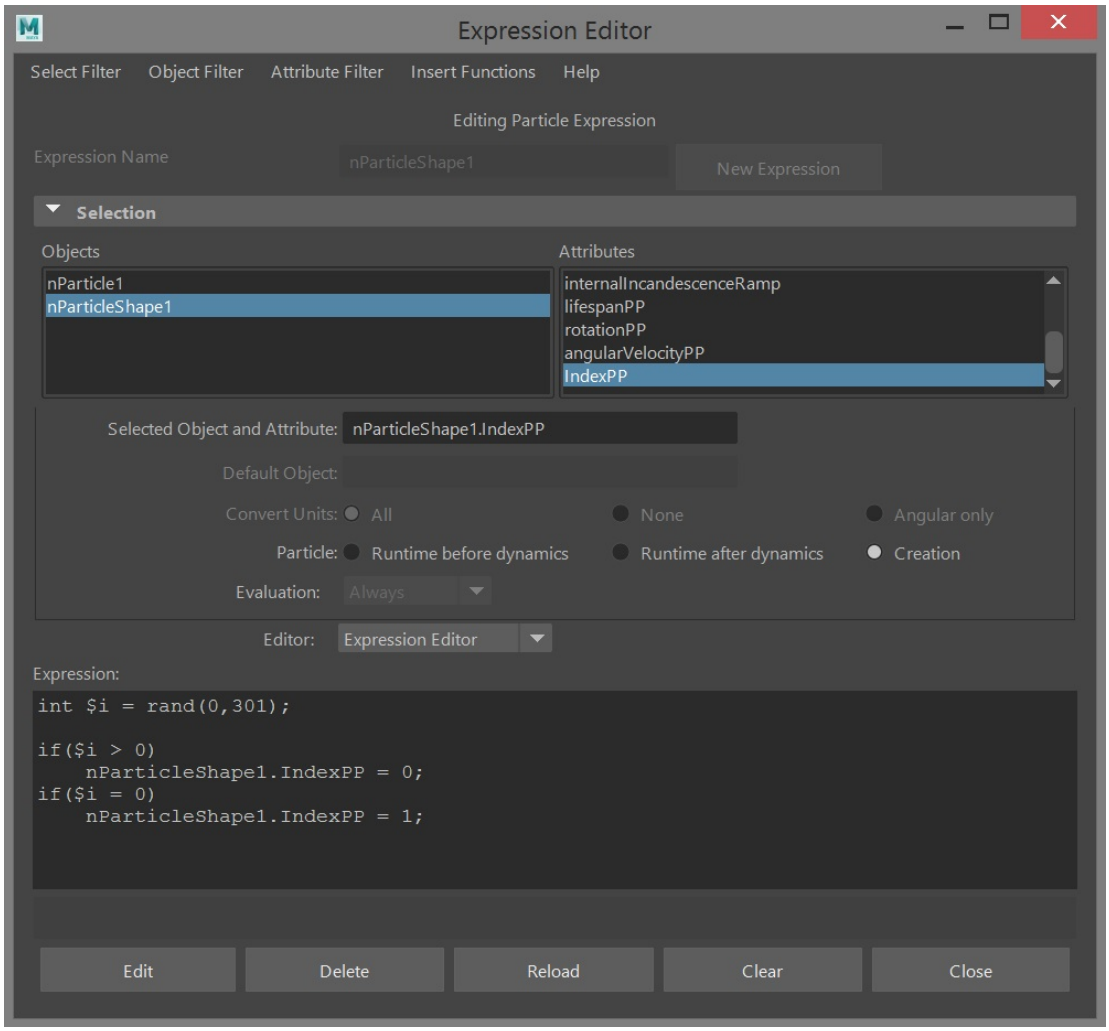


Figure 80: Editing Expression for assigning different shapes to nParticles proportionally

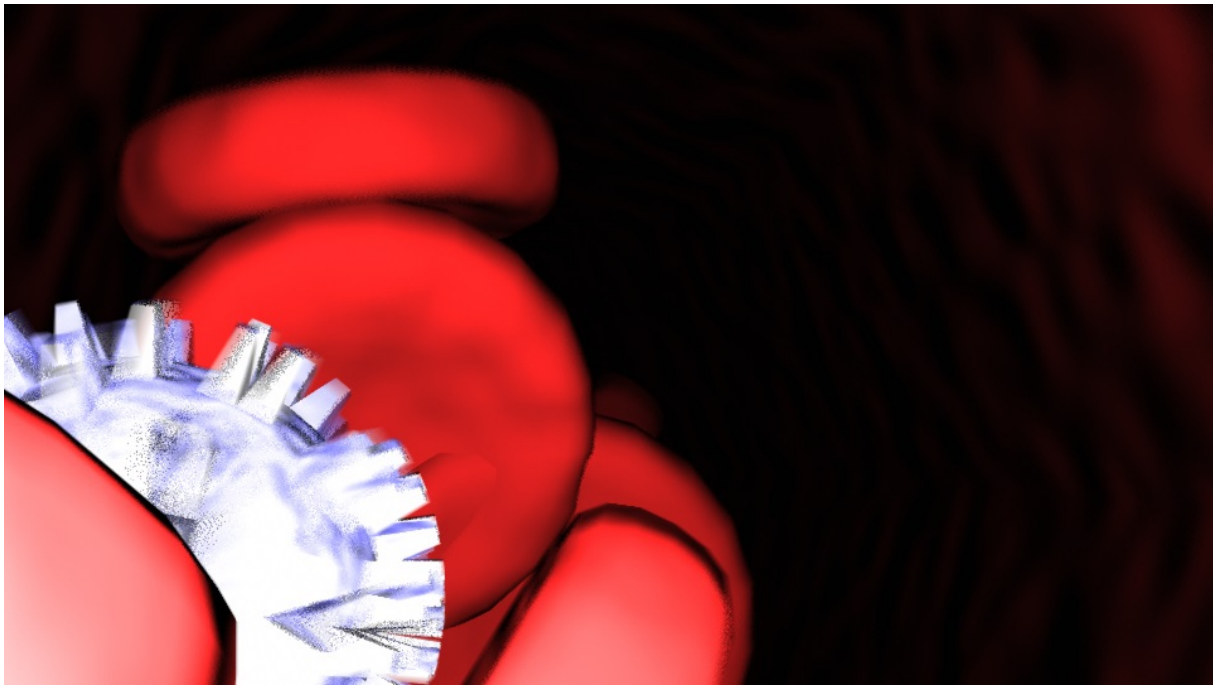


Figure 81: White blood cell flowing within Pulsating blood stream inside an artery

2.6.8 Procedural animation

Building a **System,** instead of manually keyframing every attribute change for each individual object, allows changing multiple object's attributes with subtle differences as seen in previous chapter. We have covered *nParticle Simulations*, which usually need to be created from ground up each time we start a new project. Their setting and fine-tuning may be also time-consuming, and sometimes quite tedious task, but allows things behaving in physically accurate manner, namely *colliding* with each other, as well as with any other geometrical object (after attaching it as a *Passive collider*), where each collision affects particle's motion (particles behaving as *Active colliders*).

For developing fairly complex behavior in rather time-effective and comfortable way, ***MASH*** is most convenient tool within MAYA environment, allowing us to build procedural systems, which are reusable and simply adaptable to meet various scenes requirements.

After selecting our geometry within an *Outliner*, we may create a *MASH Network* under *Animation*, or *FX* menu-set, via *MASH* → ***Create MASH Network***. Consequently, bunch of available nodes appear inside main *MASH* ***Waiter*** node.

First thing to set is *Distribution* of newly created geometrical objects (*Primitives*), accessible via ***Distribute*** node. We can choose which shape would their overall distribution take (*Line*, *Sphere*, *Grid*) and subsequently adjust number of *replicated* objects, as well as their spatial relationships (how far they should be from each other).

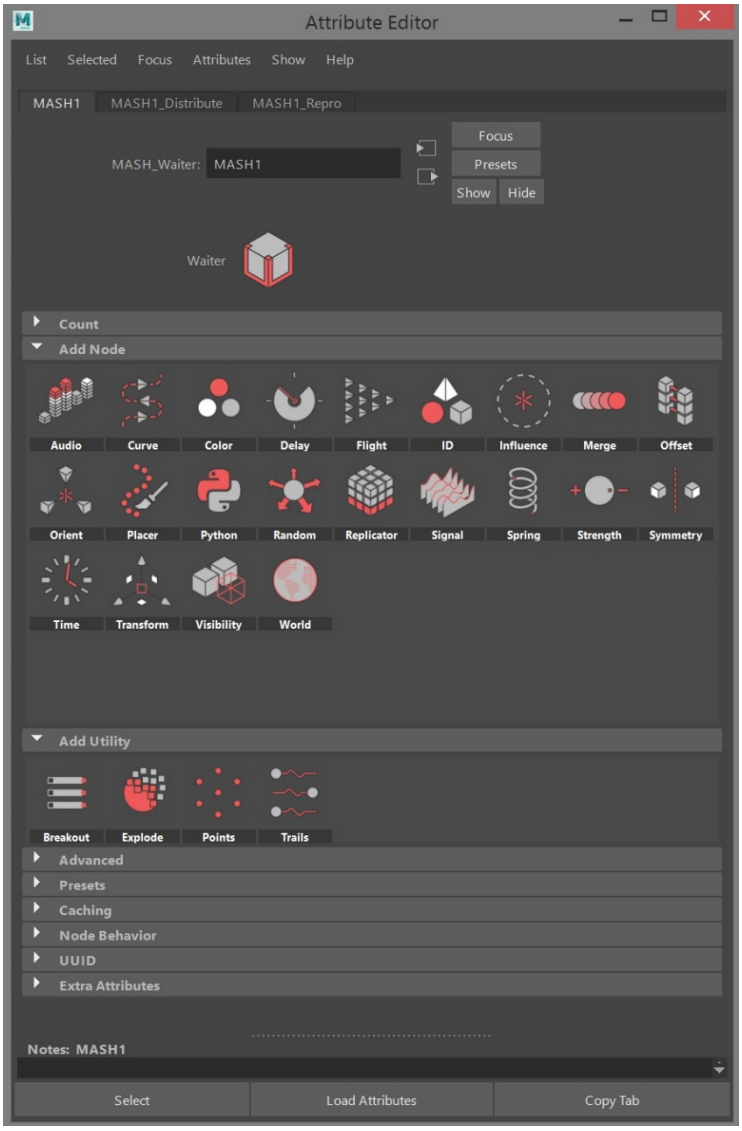


Figure 82: MASH is most convenient tool for creating procedurally based animations

2.6.9 Developing complex behavior via MASH

We can even attach *Primitives* to another geometrical object (***Mesh***), behaving as advanced version of *Displacement Map*, or disperse them within its **volume** (object serving as container) by just a few clicks:

1. Selecting *Distribution Type* as a ***Mesh***.
2. Changing distribution *Method* to ***Voxel***.
3. Under *Voxel Settings*, typing reasonably small number into ***Maximum Voxel Count*** box to avoid overloading computer after attaching an external geometrical shape.
4. Changing *Voxel Mode* to ***Fill Only***, so the external geometry will properly encapsulate MASH primitives.
5. Attaching external geometrical shape by Middle-mouse-button dragging the object from *Outliner* to ***Input Mesh*** box (initially saying *Not Connected*).

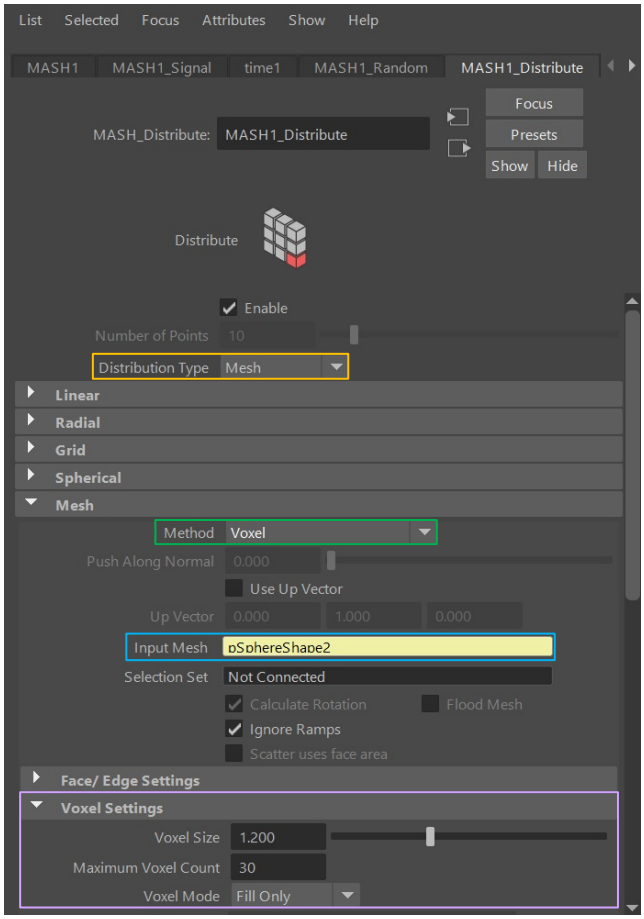


Figure 83: Setting volumetric Distribution for attached geometrical object (Mesh)

In case of *Mesh* distribution, MASH primitives are automatically re-positioned, according to *Mesh* distribution settings. Other types (*Linear*, *Radial*, *Grid*, *etc.*), however, cause objects to emerge in scene's center and cannot be moved via *Move Tool* or *Channel Box* attributes, but may be repositioned via an ***Offset*** node, providing *Channel Box* attributes with a few more controls, allowing advanced re-positioning in relation to the scene, or eventually attached geometry in case of *Mesh* distribution type.

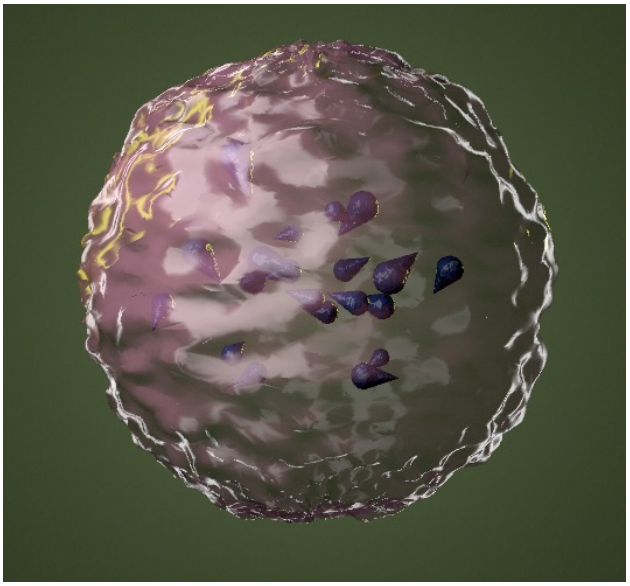


Figure 84: Blue cones dispersed within Tissue-analog sphere

To randomly move and rotate replicated objects, we may add ***Random node***, which is rather static. Moving, rotating and scaling primitives over time in random, but precisely controllable, manner can be achieved by ***Signal node.*** Most useful *Signal Type* is pleasantly behaving ***4D Noise***, although other types may be convenient for particular production needs as well. ***Noise Scale*** allows managing amount of overall random motion over time, retaining proportional randomness in different directions. ***Strength*** provides similar effect, but can also introduce additional randomness.

Since *Signal* motion appears rather jerky, smoothing it with ***Spring node*** provides more natural appearance almost effortlessly.

Controlling *Strength* of individual node's effect is possible also by utilizing ***Falloff Objects***, in which the *Inner sphere* corresponds to area with full *Strength*, whereas *Outer sphere* borders area, where involved effect diminishes, allowing it to rise gradually.

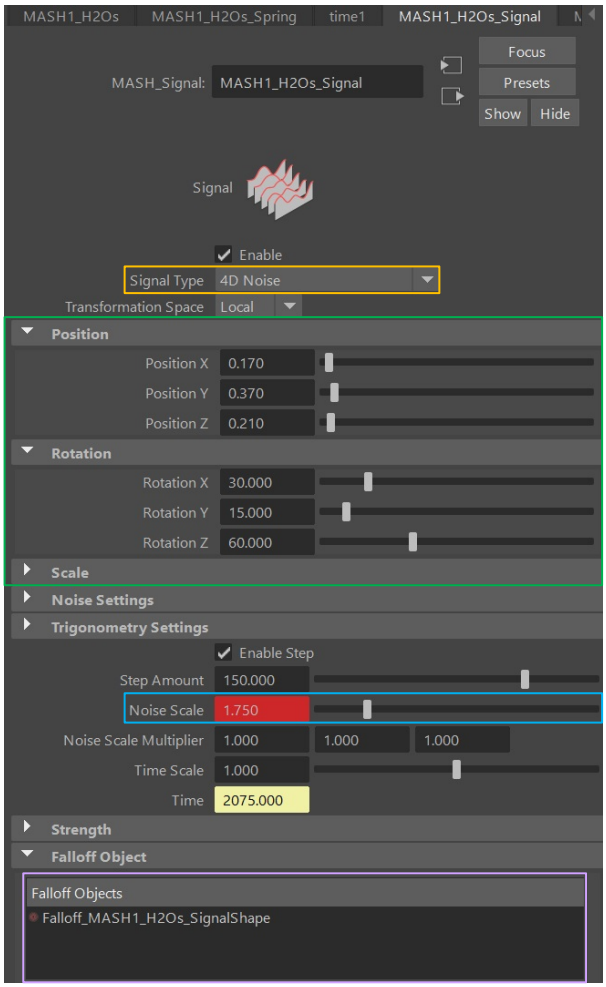


Figure 85: Signal node allows controlling random motion over time comprehensively

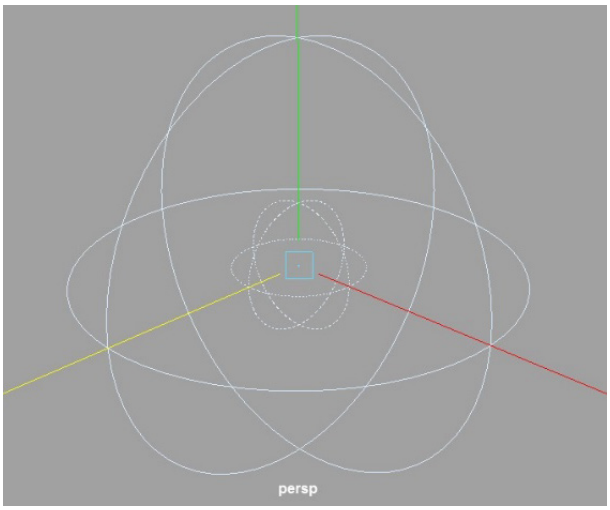


Figure 86: Falloff Object for art-directing procedural animation manually

Since *Falloff Object* can be handled via standard toolset (*Move Tool*, *Channel Box*), Keyframed and move through various areas differently over time, it provides unique opportunity for **art-directing procedural animation manually**, which, in case of physically-based simulation, would otherwise require rather advanced, time-consuming scripting.

The third initially created MASH *Core node* is ***Repro node*** containing an *Object List* to which we can import different geometrical objects from the *Outliner*, by MMB click&drag method, for reproducing them within given MASH Network.

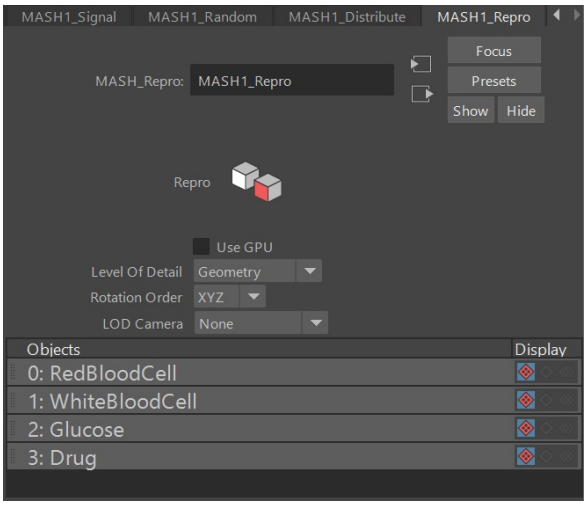


Figure 87: Importing geometry by MMB

By default, only ***ID "0"*** object (RedBloodCell) is being replicated. Introducing other ones can be achieved by creating an ***ID node*** and subsequently setting *ID Type* as ***Random***. Number of other Shapes taking place can be adjusted via *Strength* settings, *Falloff Object*, or custom *Expression*.

Last step within *nParticle* simulation was *Baking Instancer to Objects* to avoid overloading computer / confusing rendering farm. Similarly, MASH Network can be "*Baked*" into an ***Alembic Cache*** file (~.ABC), after selecting ***MASH_ReproMesh,*** via *Cache* → *Alembic Cache* → ***Export Selection to Alembic*** → *Export Selection.* In *Advanced Options,* we can choose, which parameters will be written into the *Alembic* file (*UV Write* and all below being usually useful without significantly increasing file size).

After *exporting Alembic Cache*, we can ***import*** it via *Cache* → *Alembic Cache* → ***Import Alembic…*** and subsequently delete original *MASH Network*, along with corresponding *MASH_ReproMesh*. ***Save Scene As…*** is recommended before deleting an actual *Network*, as we may want to adjust its attributes in the future. Unchecking *Outliner* → *Display* → ***DAG Objects Only*** checkbox reveals remaining MASH nodes, so we can delete them manually as well.

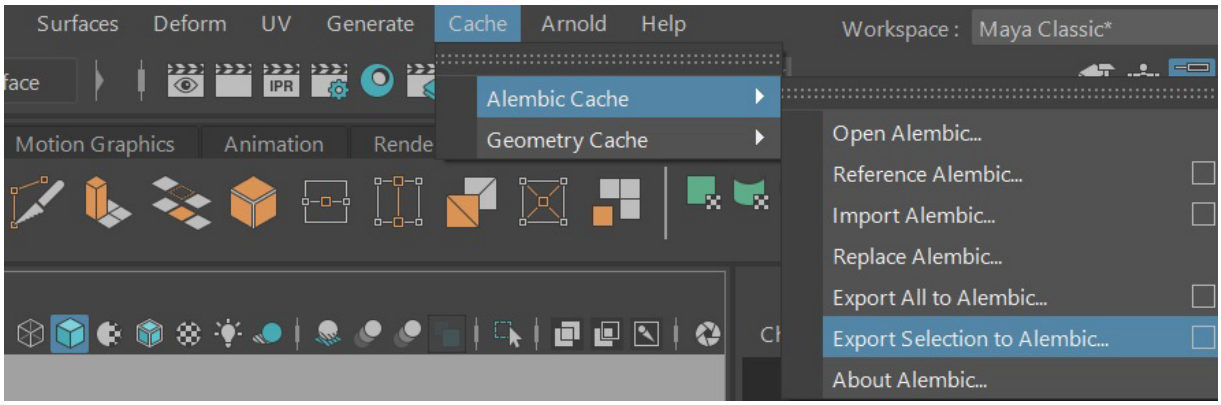


Figure 88: Exporting MASH Network as Alembic Cache for ensuring proper rendering

If colors within *MASH Network* have not been set explicitly (e.g. by *Colors Per Vertex*, requiring rather complicated preparation before rendering properly), or exported from original geometry even after check-boxing *Write Color Sets* and *Write Face Sets* within *Export Alembic* window, *Assigning Material* represents last step. In such situation, however, only one *Material* can be assigned to all objects from a single cache file.

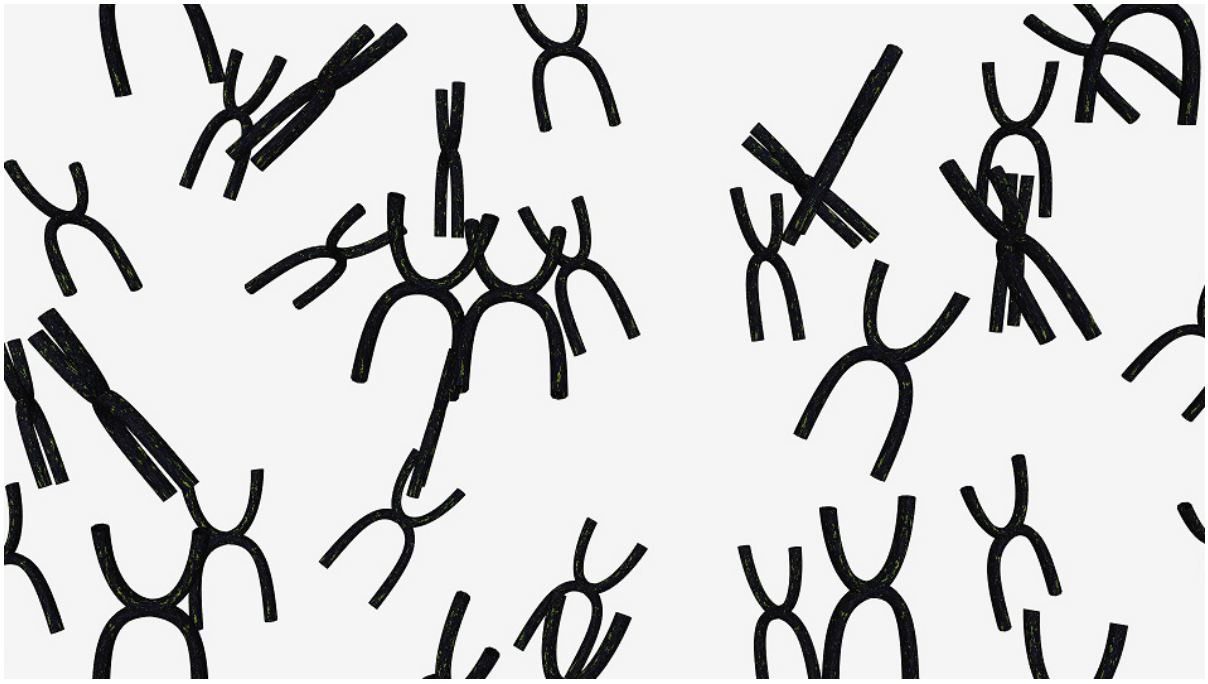


Figure 89: Randomly floating Chromosomes animated procedurally via MASH

2.6.10 Comparing animating approaches

As we have seen, each approach for creating an **Animation** entails different challenges determined by tools it utilizes for achieving final results. Before starting an animation process, it is important to realize, what should be the final **Outcome** (individual object undergoing change / several geometrical shapes exhibiting complex movements / thousands of particles colliding with each other, while dragged by some physical field) and what **Resources** we have available at the time (time for development, computational power of computer / render farm).

Based on these factors, we can competently select most suitable approach for given production needs in a particular situation.

Table 11: Comparing different animating approaches

Animating approach	Number of objects	Suitable utilization	Developing time
Traditional Keyframing	Individual	Changes over time	Short
nParticle Simulation	Many	Colliding particles	Long
MASH Network	Several	Complex behavior	Medium

2.7 Rendering images

After our scene is completed, we want to ***render*** sequence of resulting images, that will provide continual movement illusion, if played at appropriate speed (e.g. 24 frames per second). But *rendering* process can follow different rules, leading to completely different results.

Furthermore, we may actually choose from various ***Renderers***, each of them handling these rules in its own way, similarly to a **Painter** putting a live three-dimensional scene on two-dimensional canvas. He would have his own *Point of View*, pre-determined by his personality and experiences, but also would prefer to use particular *Painting techniques* he favors most for his own reasons. Therefore, different *Renderers* will produce completely different **Look** of resulting image, even when we set them to work as similarly as possible.

Selecting most suitable *Renderer* for particular production should happen very early in project's development stage, since different renderers demand different setting of various rendering-related objects (*Cameras*, *Lights*, *Shaders*) and may even require using their own.

Each *Renderer* provides specific benefits at some cost. Deciding which one is most appropriate, in a particular situation, depends on production **needs** and available **resources**, and can be assessed by ***Project Management Triangle (PMT)***. The task here is to pick two aspects, we consider most important, at the expense of the third, so there are basically three options:

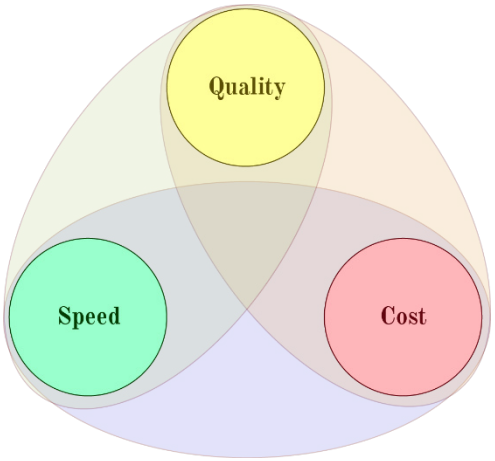


Figure 90: Project Management Triangle in "Pick any two" form

1. We can have a renderer that provides high-quality images produced relatively quickly, but its license will cost a lot of money.
2. There are also some fast renderers free of charge, but their quality is usually inferior to commercial solutions.
3. Producing superb quality images at low cost will be considerably slow, either due to extremely long rendering times, or time spent to manually tweak various parameters and perform additional tasks manually (also requiring profound under-the-hood rendering skillset).

2.7.1 Built-in renderers

MAYA currently has two intrinsic renderers: ***MAYA Hardware Renderer [2.0]*** which is fast and efficient, as it utilizes GPU acceleration for computations and provides reasonable quality of resulting images, although inferior to software rendering approaches. ***MAYA Software Renderer*** can fully utilize all MAYA's intrinsic functions and capabilities. By using CPU, it may process data with more sophisticated algorithms, providing more physically accurate results at the expense of much longer rendering times. Both MAYA renderers are free of charge, or, more precisely, their costs are included within MAYA license.

Since MAYA 2017 features industry-leading ***Arnold Renderer***, fully integrated within MAYA's environment, this would be the way to go, as it provides superb quality photorealistic results (when appropriately managed) at competitively short rendering times. For performing **Background rendering** tasks, such as ***Batch Rendering*** (automatized creation of multiple images), or executing a ***Backburner job*** (rendering via interconnected computers available within particular network), it is necessary to purchase a full-blown license from ***Solid Angle*** (company developing Arnold). However, images may be rendered *interactively* within MAYA's interface as **single images** (and subsequently saved to hard drive manually), or even **sequence of images (!)** via ***Render Sequence*** command. This makes ***Arnold*** a prime and ultimate solution for rendering photorealistic images.

In earlier versions of MAYA (up to 2016), *NVIDIA* ***Mental Ray*** was licensed and fully integrated production-quality renderer, providing less photorealistic results than *Arnold Renderer* for longer rendering times, but still superior to *MAYA Software Renderer*, by managing more advanced physically-based effects (e.g. *Material Light Refractions*, or *Sub-Surface-Scattering*). ***Mental Ray*** can be still used, but needs to be installed and licensed separately, when convenient for particular production needs.

Table 12: Comparing MAYA integrated renderers by PMT

Renderer	Quality	Speed*	Cost	Production
MAYA Hardware [2.0]	Poor	Extremely Fast	Free	Modest
MAYA Software	Good	Relatively Slow	Free	Demanding
Arnold Renderer	Superb	Reasonably Fast	Free / Paid	Photorealistic
Mental Ray	Very good	Moderate	Paid	Motion Graphic

Speed* here is interconnected with *Quality* and concerns only *Rendering speed* for obtaining a given *Quality* result, assessing thus **overall renderer's performance (e.g. *MAYA Software* provides good quality images, but produced **Relatively Slow** in comparison to demanding same *Quality* output from other renderers).

2.7.2 Selecting suitable renderer

Since all discussed renderers offer different **Qualities**, we cannot say whether one is better, or worse, than another, because **each renderer is optimized for particular production needs**, thus more productive question would be: "*Which renderer is most suitable for this particular production?*", in other words – *which features are essential for achieving given task effectively* and consequently *which renderer offers these features*?

Only then we can qualifiedly evaluate various renderer's performance and select most suitable one in a particular situation. For example, when we need to produce a lot of images in very short time, but can accept "poor" image quality (images looking obviously artificial), most appropriate renderer here is ***MAYA Hardware [2.0]***, as its rendering times are second to none, in comparison to the others, making it convenient when producing e.g. Educational movies for schools.

MAYA Software allows utilizing advanced features (such as *Displacement Maps*) and provides better-looking images, although still obviously artificial, but takes considerably more time to render. If we just need to access *Maya Hardware* inaccessible features and get more pleasant-to-the-eye images, this one may be quite convenient.

When requiring **Photorealistic results**, not only render times get significantly longer, but adjusting all relevant *Shader* nodes takes a lot more time, as well as developing extended shading skillset (while *setting* convenient, although artificially looking, Materials with standard *Shading* nodes takes about 5 - 15 minutes, *developing* photorealistic ones may be rather matter of 5 - 15 hours). If intending to *compose* resulting images with a real-world footage, ***Arnold Renderer*** is most convenient, but long render times should be expected. However, this may change in near future, since ***Solid Angle*** is tweaking *Arnold* to utilize GPU acceleration, which is anticipated to speed up rendering at least 2 – 5 times generally, but, in some cases, can provide **50x faster** rendering (!), when compared to utilizing only CPU computational workflow.

Obtaining ***Mental Ray*** license may be appropriate, if our production pipeline is heavily based upon *Mental Ray* nodes, especially when collaborating with companies satisfied with this renderer for many years, and cannot be expected to change it in nearest future.

There are other powerful renderers as well, many also free of charge (*RenderMan*, *V-Ray*), however, they are not deep-integrated with MAYA, so managing them properly would be more challenging and time-consuming than utilizing abovementioned ones, so they might be more convenient for work performed within other CGI packages, such as open-sourced *Blender*.

2.7.3 Setting Common rendering attributes

In order to render animation into sequence of images properly, selected *renderer* needs to be set appropriately via ***Render Settings*** window, containing a ***Common*** tab, shared across all renderers, and bunch of *renderer-specific* tabs.

Uppermost ***Color management*** section allows applying Look-up-Table (LUT) during final rendering steps, basically just changing **Luminance** (brightness) and **Chrominance** (color) of each image. ***Use View Transform*** variant assures that rendered images appear the same as within a ***Render View*** window, but we can also choose another pre-set, or even import our own LUT, which suits particular production pipeline.

If intending to perform **Color correction** within dedicated composing application (Nuke / Adobe After Effects), we can just leave corresponding checkbox unchecked.

Under ***File Output***, allows selecting an ***Image format*** suitable for our particular productional needs. **PNG** is a great general-purpose choice, as *~.png* images are directly viewable via standard operating system's image-viewing applications, retain **transparency** (*Alpha Channel*), and are considerably space-saving (~1 MB per full HD image at 72 dpi resolution). Naming convention can be adjusted via ***Frame / Animation ext.***, where *"#"* stands for *frame's number*. Usually, extension should be written at the end, as various applications may not be able to handle output files properly, if using exotic naming convention.

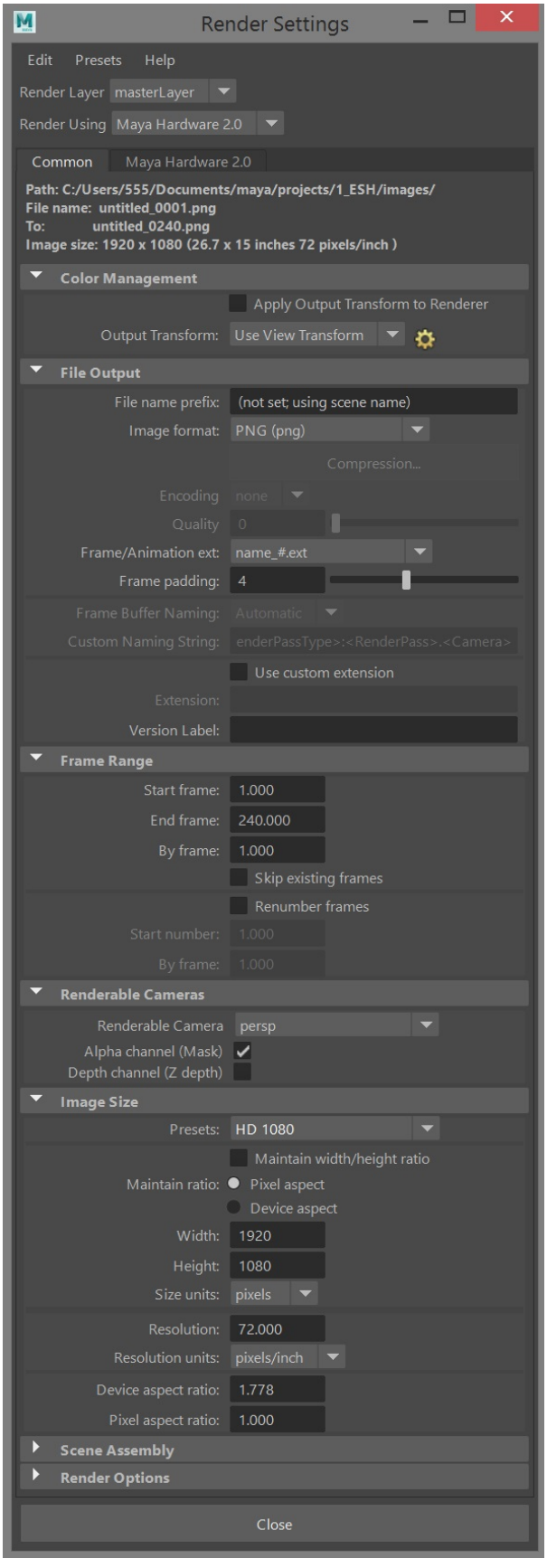


Figure 91: Common Render settings

Frame Range determines, which frames from *Timeline* should be rendered, when producing multiple images via *Batch Render* or *Render Sequence* command.

Within ***Renderable Cameras***, we can select which camera's Point of View will be used for *framing* whole scene. When utilizing multiple cameras, converting them into *Ubercam* object allows rendering their individual *Views* at corresponding frames, established inside *Camera Sequencer*, in a single rendering pass. ***Alpha Channel*** checkbox should be checked for preserving *Transparency* if rendering multiple layers (e.g. **Foreground** object, which will be subsequently composed on solid **Background**).

Image Size should be established before setting individual *Cameras*, as changing it afterwards would change image composition, so we would have to rework it, which could be quite tedious task, if camera attributes are animated. ***HD 1080*** is currently considered an industry standard, but different one may be utilized as well for particular production needs (Cinema formats, Real-world camera framing, Special-purpose exotic Aspect Ratios). Also, ***Device aspect ratio*** should match *Film aspect ratio* within individual camera's *Film Back* section.

2.7.4 Adjusting render-specific attributes

Various renderers provide different tabs and diverse attributes, but most important quality-determining ones are based upon ***Sampling*** controls, allowing us to set number of *Samples* for computing various scene's features. ***Anti-Aliasing*** determines how much are individual object's edges smoothed, having significant impact on overall image quality, but also being computationally heavy. Number "3" here stands for an exponent, actual number of samples being 2^3 = **8** and same goes for other sub-attributes.

Enabling ***Raytracing*** will vastly improve rendering images in physically accurate manner, regarding vital light-propagation properties, thus substantially contributing to achieving conveniently photorealistic look (along with actual *Shaders* tweaked during *Material* development stage), but is extremely **computationally heavy**.

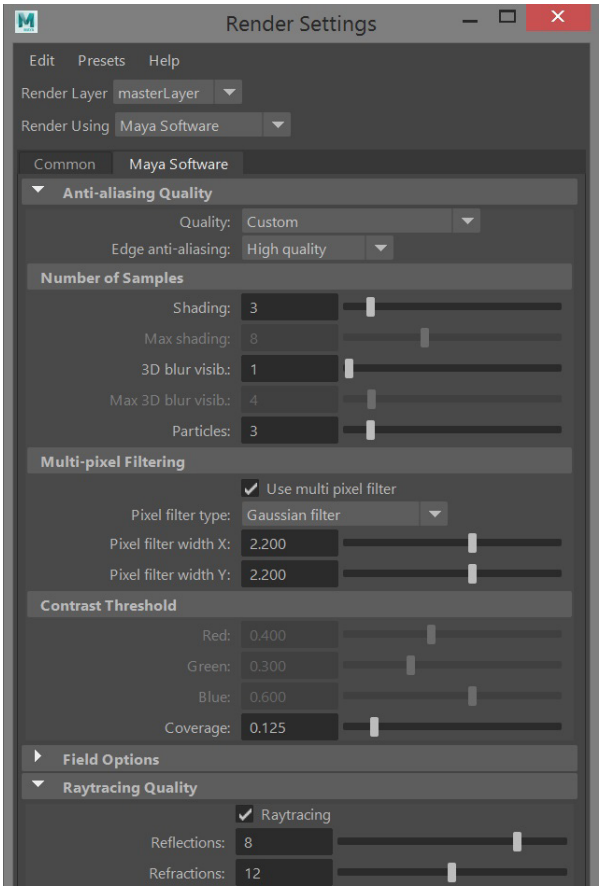


Figure 92: Sampling determines image quality and rendering time

Sliders here resemble number of passes, taken by each *Light Ray*. Reflections = 8 means light will bounce 8 times among objects, creating reflection on them, until it disappears. When using semi-transparent objects, having enough ***Refractions*** is important for actual Rays passing throughout whole scene properly (each physical *Material* in Ray's way, having different **Index of Refraction** from its ambient environment, will require adding +1 to this attribute).

Arnold Renderer has multiple tabs, but the native one determines overall image quality. We can precisely control number of samples for Arnold-specific *Shading* features, where ***Camera(AA)*** serves as a general sampling **multiplier**, so each sub-attribute sampling number is multiplied by this value.

Setting individual values will depend on feature's importance within particular scene, therefore e.g. shading organic materials, to appear photorealistic, will require sufficient number of *Sub-Surface-Scattering* (SSS) samples.

Arnold also allows ***Environmental Lighting***, so by clicking on *Environment [checkerboard]*, we can create a physical sky resembling *Sphere*, and establish, for example, *Image Based Lighting* by selecting *Create Sky Shader* and inputting external *Bitmap file* into its *Color* attribute, which will project the image onto Sphere's surface and introduces additional light to the scene, colorized by *Bitmap's* color channels.

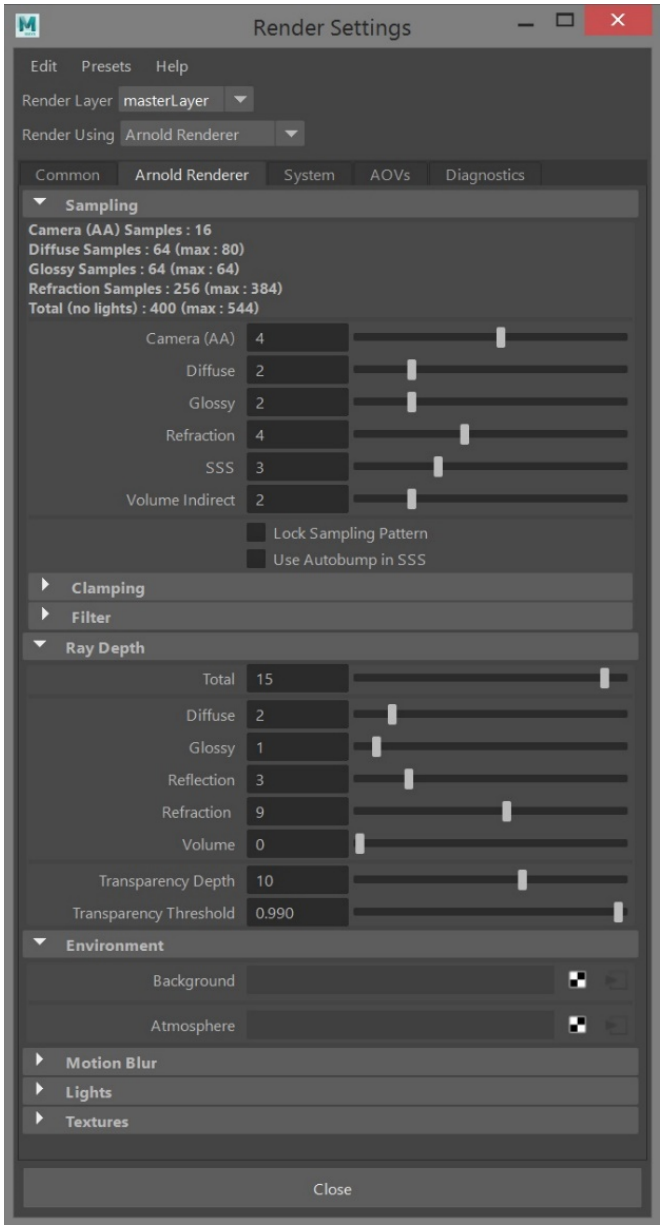


Figure 93: Arnold allows precisely manage individual feature's quality by sampling values

2.7.5 Tweaking MAYA Hardware [2.0] renderer

Since *Viewports* are using ***MAYA Hardware [2.0] Renderer*** for displaying our scene, if we are intending to perform final render via this *Renderer*, we can precisely evaluate various scene aspects over whole production process, as they are displaying in very similar way to final images. Their appearance, however, may vary depending on enabled features within *Renderer Settings*, so we may choose which ones we currently need to see in a particular CGI production stage.

For final render, we just enable all those we desire and reasonably boost their attributes, based on required image quality and speed of our computer. Since *Hardware Renderer* takes advantage of GPU acceleration, switching *Vertex Animation Cache* to ***Hardware*** will speed up things considerably. Most comprehensive ***Transparency Algorithm*** is ***Depth Peeling***, but different ones are available as well, handling *Transparency* attribute differently, so they provide slightly different results.

Screen-space Ambien Occlusion here is a post-rendering effect, approximating how much are individual points in 3D scene exposed to ambient light, introducing visually pleasing shadows to final images.

Hardware Fog simulates real-world atmospheric haze and can be adjusted as needed. But for being actually computed (and thus visible), we also need to check ***Hardware Fog*** checkbox hidden within ***Render Options*** section.

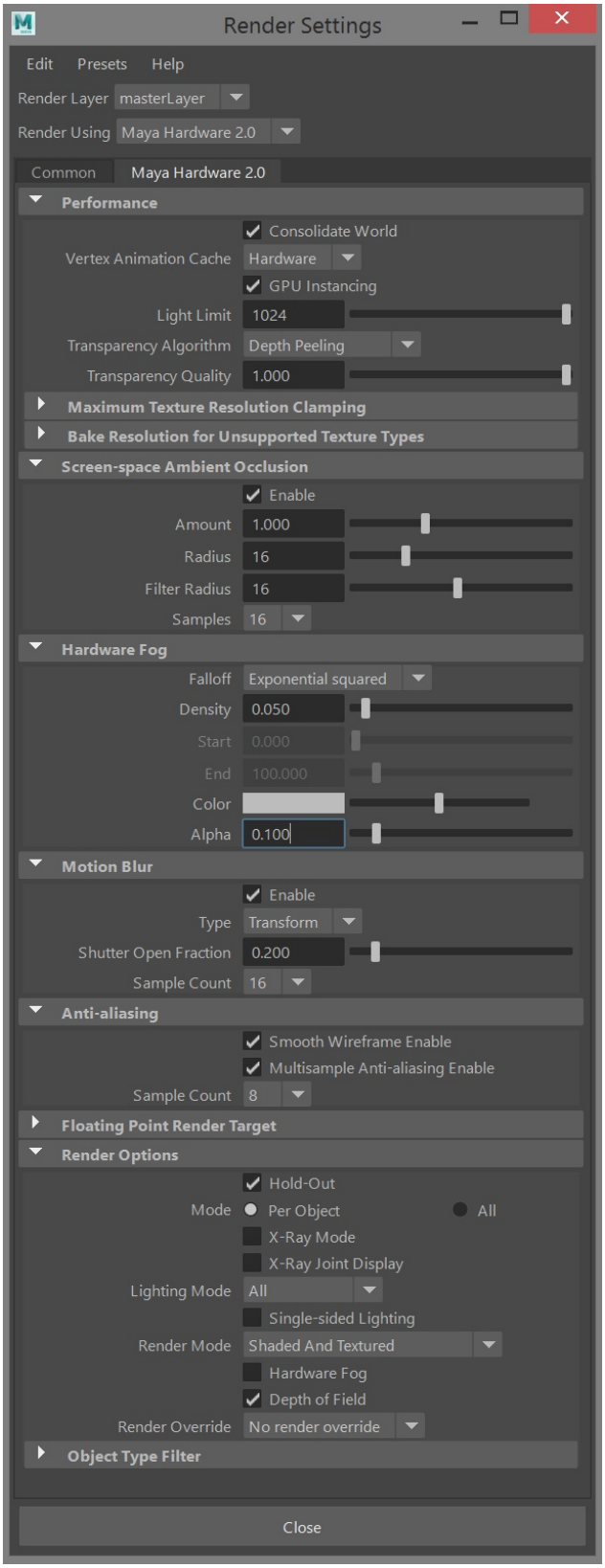


Figure 94: Reasonable settings for rendering images in production quality

Motion blur prevents jerky movements during animations by introducing physically based Motion blur to the scene, where amount of blurriness taking place can be managed by ***Shutter Open Fraction*** attribute. ***Anti-Aliasing*** here allows choosing an actual number of samples, in comparison to other Renderers. Selecting which objects will be visible in finally rendered images can be performed by checking corresponding checkboxes within ***Object Type Filter*** section, or via establishing ***Render Layers.***

2.7.6 Layers vs. Passes

When we finish tweaking our *Renderer*, ***Setting Project*** directory via *File* → ***Set Project…*** is a next step, if we have not done it before. Subsequently starting ***Batch rendering*** process, accessible via *Render* → ***Batch Render*** command, will create an *image folder* within previously specified directory and starts rendering all images defined by *Frame Range* under common tab.

Proceeding this way will render everything we have set into series of images, saved within *[Project_directory] / images /* ***masterLayer*** directory, by ***passing*** all data through renderer one time. Since it renders all features (variables) contributing to resulting image looking as ***beauty*** as possible, it is being referred to as a ***Beauty Pass***.

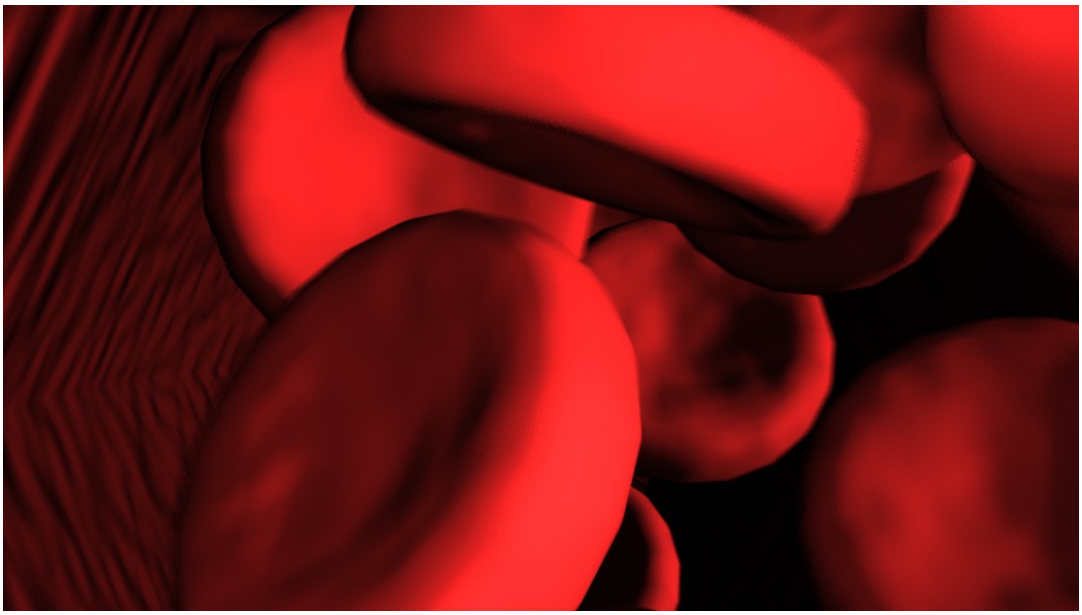


Figure 95: A Beauty Pass image of Blood artery immersive shot

By playing these images as a *sequence* at appropriate speed in **Non-Linear Editing (NLE)** application, we can preview whole animation and adjust individual shots as a whole, similarly to real-world footage. But since we are working within CGI realm, this provides us a unique opportunity to render animation divided into multiple ***Layers*** (such as *Foreground* and *Background*), **adjust them individually** and *compose* together within a **Compositing application** (or NLE software). Setting up these *Layers* conveniently within MAYA environment can be performed inside ***Render Setup Window***.

2.7.7 Render Setup Window

This window has a ***Render Setup*** tab, where we can manage individual layers, and a ***Property Editor*** for adjusting what and how will be rendered within selected layer.

Scene section corresponds to our *Master Layer*, while section beneath it allow us to create individual layers by clicking on ***Create a New Layer*** button. Renaming layers can be easily performed by clicking on them twice with Left-mouse-button.

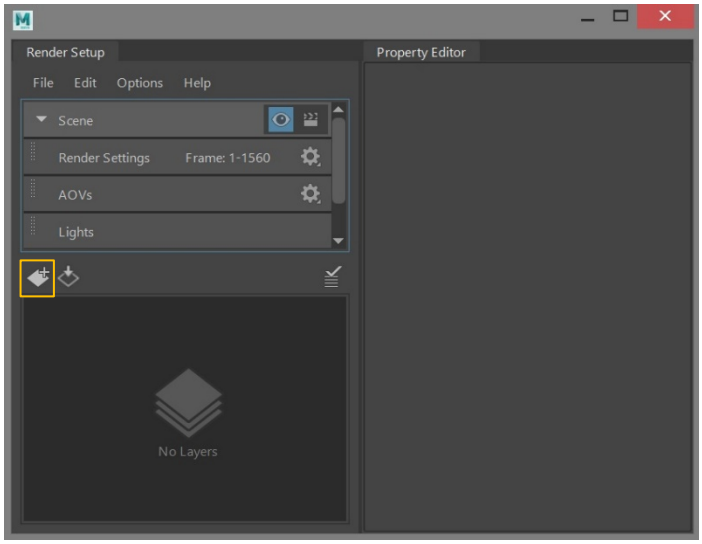


Figure 97: Render Setup Window

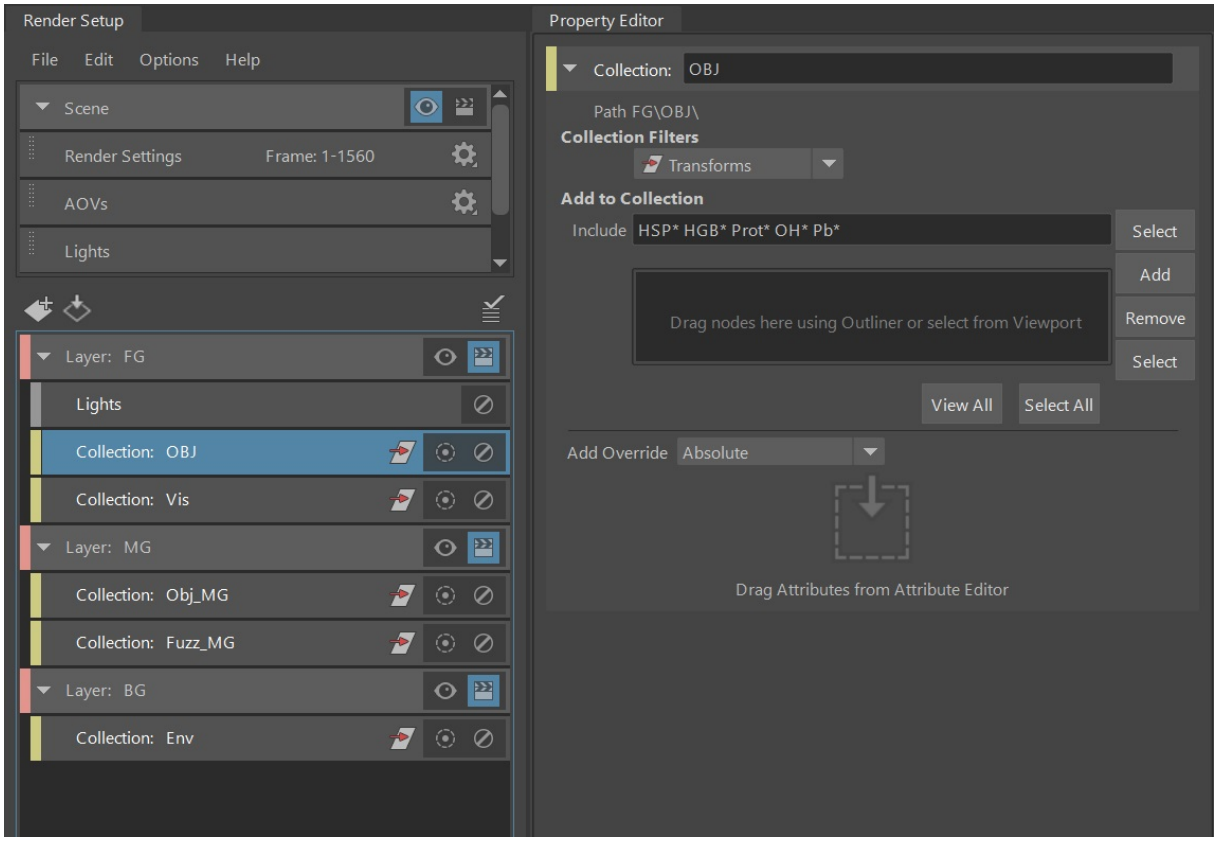


Figure 96: Three Layers are optimal for even complex scenes

For simple scenes, two layers, containing *Foreground* objects and *Background* elements may be enough, however, for more complex ones, **three layers** seems to be optimal number, since they provide sufficient amount of control concerning animation's overall appearance, can be managed within NLE application (without processing them via a compositing software) and still are reasonably quick to render.

2.7.8 Collections and Overrides

Since ***Layers*** are mainly intended for organizing *Render Passes*, we can select individual objects, which are intended to be rendered, and gather them into ***Collections*** after RMB clicking on a single layer and choosing ***Create Collection*** command. Adding objects into such *Collections* can be achieved either by MMB ***Dragging*** an object from *Outliner* window to corresponding ***Add to Collection*** area, or typing their name into ***Include*** (*Create Expression*) box.

The latter one is more convenient, if all objects are appropriately named, as we can type only a few characters within an *Asterisks*, and MAYA automatically selects all objects containing these characters, whatever is at the place of an Asterisk.

This has a big advantage when working with a more complex scene containing tens, hundreds, or thousands individual objects, as selecting them manually would take considerably more time.

Furthermore, if adjusting scene in a future and creating additional objects, these will be **automatically** included within corresponding collection, provided they are named appropriately.

Selecting all object of the same type is also possible via ***Collection Filters*** for managing them at once.

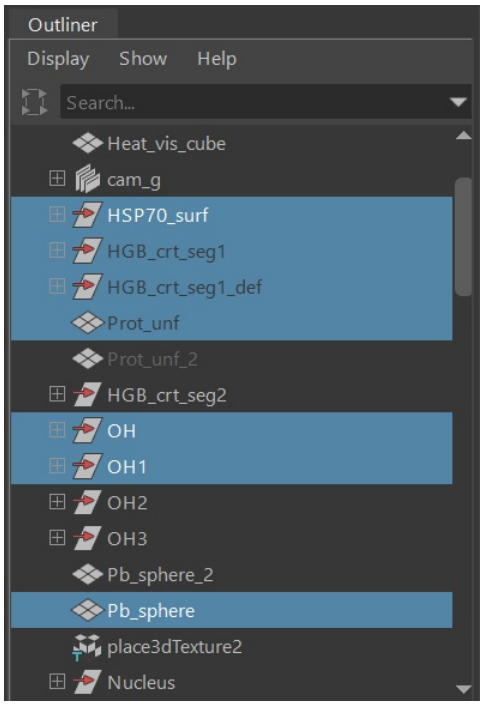


Figure 98: Expression adds new objects automatically

After *collecting* all desired objects, we may also add an ***Override*** to various object's attributes affecting every object within a collection, without changing actual values of selected attributes.

These changes can be either ***Absolute*** (all objects will have exactly this new value within corresponding attribute), or ***Relative*** (each object will retain its original value, which will be adjusted by typed number).

We can also affect *Materials* by ***Create Material Override*** command and subsequently input whatever *Material* we desire to replace original ones of all objects within given collection (e.g. whilst *Foreground* layer contains a collection with object's original materials, *Masking* layer may contain similar collection overriding these materials by *black Lambert-type Material*, acting as a perfect object's *Matte* within a particular layer).

Each *Layer* may contain multiple *Collections*, which helps us to keep scene *Render Setup* window organized and create different attribute's overrides for objects within the same layer.

2.7.9 Rendering image Sequence

Each *Layer* within *Render Setup Window* has two icons next to its name. The ***Eye*** shows corresponding layers in our *Viewport* (helpful for visually checking they are properly set), whereas the ***Clapperboard*** icon determines, which Layers will be rendered (so we may e.g. disable *Master Layer*, or render only one layer in the future, if being re-worked).

Rendering only ***Master Layer*** is convenient for performing pre-view renders at minimal quality setting, to make sure that everything is working as expected. Before starting a *Batch Render* process, rendering whole animation range, rendering only a few frames of all three Layers provides last visual verification, re-assuring us that they are set-up properly.

Depending on utilized *Renderer,* we can render sequence of images either by ***Batch Render*** command (MAYA Hardware / Software renderer), or ***Render Sequence*** (Arnold). If having access to rendering farm, or computer network, a ***Backburner Job*** may be also created for background rendering on multiple machines (workstation's firewall rules must be properly configured, as well as *Backburner Server* on a rendering farm).

When *mayaBatch.exe* process is running (can be checked by *Microsoft Windows* **Task Manager**, or similar application on particular operating system), we may close MAYA to preserve more computational power for rendering process, when needed.

After rendering is completed, we can find all rendered images saved inside *[Project_directory] / images /* [**Layer_name**] directories, corresponding to individual layers, and import them into **Compositing** application, or **Non-Linear Editing** software, for further processing (Compositing, Color correction & Color grading, Audio mixing, Encoding).

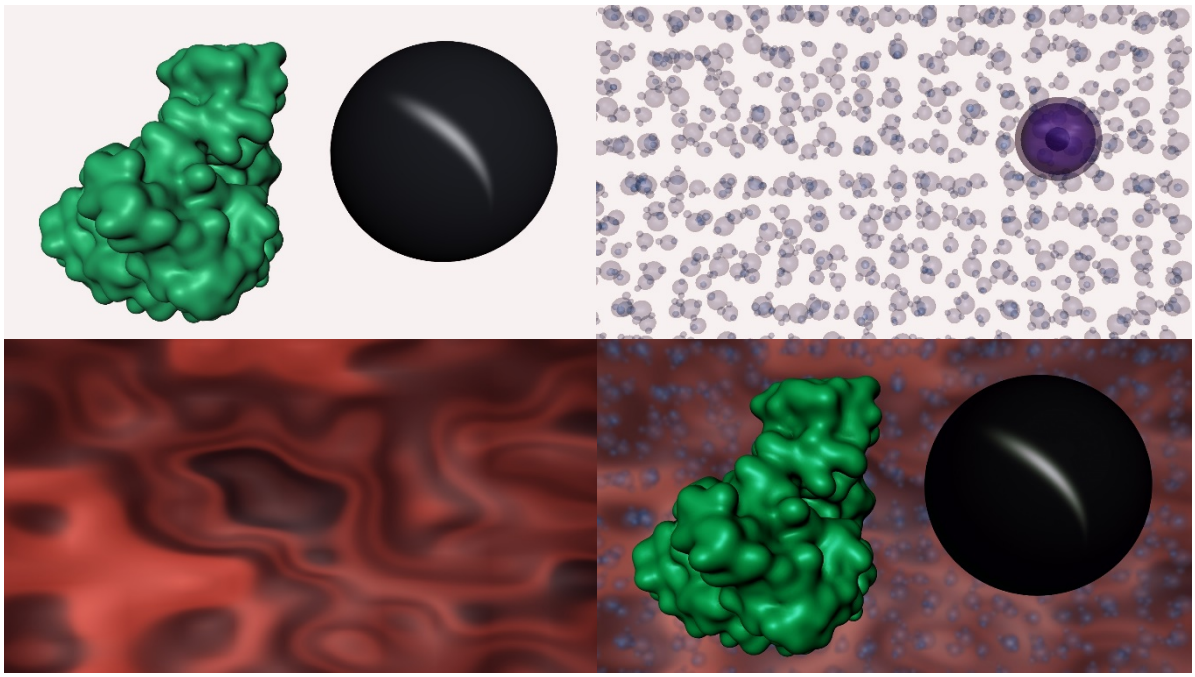


Figure 99: Rendered images from FG (top left), MG (top right), BG (bottom left) Layers and Final composite image from resulting Movie after NLE operations (bottom right)

CONCLUSION

The aim of this thesis was to present current state-of-the-art approaches for visualizing Biological structures, and animating them, in visually appealing manner.

Within **FIRST PART**, we have briefly covered three fundamental areas for creating visually appealing content, effectively conveying core scientific knowledge ideas:

Communicating them visually requires wide spectrum of competencies involving scientific, pedagogical and artistic skills, whose profound understanding can be accomplished by utilizing theoretical concepts in daily practice.

Data acquisition is essential for understanding physical phenomena taking place in our world and their physically accurate digital representations. Suitable methods in different areas of biological study provide information in various output formats, which needs to be consequently represented in standardized form to assure their usability in different scientific applications all over the world.

CGI workflow allows creating visually appealing content effectively. Choosing appropriate software solution is important first step, as various CGI tools are optimized for different user needs. Although various digital models may be obtained and customized without building them from scratch, fundamental modelling and animating skills are essential for working with them properly, as well as creating supplemental content, which needs to be constructed individually to fit particular scene.

Practical example of creating animated movie, based on current scientific knowledge, from scratch is provided, along with brief description of important steps and putting crucial decisions into context with previously discussed topics, illustrating their factual contribution to everyday practice.

After covering theoretical background, necessary for understanding how scientific concepts can be vividly and comprehensibly presented as visually appealing content, as well as discussing tools convenient for doing it effectively in as physically accurate manner as possible, the **SECOND PART** was aimed to provide best practices guiding scientific workers in Bio-medical field through process of creating biological structure's animations within Autodesk MAYA®, being most comprehensive software environment for creating computer generated imagery (CGI), although discussed concepts can be utilized also within other CGI software packages, such as an open-sourced *Blender*.

Since MAYA enormous complexity, we are first discussing software's **Graphic user interface**, navigation and different approaches for accessing same functionality, which are being commonly used in a daily practice.

Subsequently, we introduce **fundamental modelling and animating approaches**, along with other tasks (establishing *Cameras*, *Lighting* and *Rendering* setup), necessary for turning animated geometry into visually appealing images. Different approaches are discussed to suit various scientific and production needs.

As a practical example, how these techniques can be utilized for producing astonishing, formerly infeasible, results, portraying physically accurate biological structures, **short animated Movie**, along with Pulsing blood flow animation, has been produced and subsequently placed on DVD meeting current European standards.

Since main scope of the second part was visualizing physically accurate biological structures in visually appealing manner within Autodesk MAYA environment, next steps concerning *Composing* individual image layers together, creating a *Voice-over commentary*, *Editing* image & sound to create a movie, add *Visual Effects* and perform *Color Correction*, as well as *Color Grading* to enhance overall cinematographical experience, finishing by *Encoding* the completed piece into film-industry standardized formats suitable for *Digital cinema projection* and *Television broadcasting*, will be discussed within the **THIRD PART** in a near future, so they could be incorporated in a daily scientific practice.

Literature

[1] ANDRIKANIS, Ekaterina and Sergej KONDAKOV. *Homevideo, aneb Sám sobě kameramanem*. 1st ed. B.m.: Grada Publishing, 2008. ISBN 978-80-247-2192-7.

[2] BOKOVÁ, Petra. *Základy obecné pedagogiky a didaktiky*. B.m.: Brno University of Technology, 2005.

[3] BONYADLOU, Shahram. *Hybrid imaging technology - The so-called 'One Stop Shop'* [online]. 2006. Available at: http://www.ihe-online.com/fileadmin/artimg/hybrid-imaging-technology---the-so-called-one-stop-shop.pdf

[4] BOSHKOVIKJ, Veselin, Christopher J FLUKE, Russell J CRAWFORD and Elena P IVANOVA. Three-dimensional visualization of nanostructured surfaces and bacterial attachment using AutodeskH MayaH. *Scientific reports* [online]. 2014, 6. Available at: doi:10.1038/srep04228

[5] CALLAWAY, J, M CUMMINGS and B DEROSKI. Protein Data Bank contents guide: Atomic coordinate entry format description. *Brookhaven National Laboratory* [online]. 2008, 194. Available at: ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_A4.pdf

[6] DE FREITAS, Sara, Genaro REBOLLEDO-MENDEZ, Fotis LIAROKAPIS, George MAGOULAS and Alexandra POULOVASSILIS. *Learning as immersive experiences: Using the four-dimensional framework for designing and evaluating immersive learning experiences in a virtual world* [online]. 1st ed. 2006. ISSN 00071013. Available at: doi:10.1111/j.1467-8535.2009.01024.x

[7] DIAS, Jan. *Struktura filmového vyprávění* [online]. B.m., 2012. Univerzita Tomáše Bati ve Zlíně. Available at: http://digilib.k.utb.cz/bitstream/handle/10563/23119/dias_2012_bp.pdf?sequence=1

[8] DICKSON, William Kennedy and William HEISE. *Annabelle Serpentine DANCE* [online]. USA: Edison Manufacturing Company. 1895. Available at: https://upload.wikimedia.org/wikipedia/commons/d/d0/Serpentine_Dance_

[9] DRASTICH, Aleš. Tomografické zobrazovací systémy [online]. 2014, 432. Available at: https://moodle.vutbr.cz/mod/resource/view.php?id=108051

[10] DRZEZGA, a., M. SOUVATZOGLOU, M. EIBER, a. J. BEER, S. FURST, A. MARTINEZ-MOLLER, S. G. NEKOLLA, S. ZIEGLER, C. GANTER, E. J. RUMMENY and M. SCHWAIGER. First Clinical Experience with Integrated Whole-Body PET/MR: Comparison to PET/CT in Patients with Oncologic Diagnoses. *Journal of Nuclear Medicine* [online]. 2012, **53**(6), 845–855. ISSN 0161-5505. Available at: doi:10.2967/jnumed.111.098608

[11] ERAT, Okan, Olivier PAULY, Simon WEIDERT, Peter THALLER, Ekkehard EULER, Wolf MUTSCHLER, Nassir NAVAB and Pascal FALLAVOLLITA. How a surgeon becomes superman by visualization of intelligently fused multi-modalities. In: David R. HOLMES and Ziv R. YANIV, eds. [online]. 2013, p. 71. Available at: doi:10.1117/12.2006766

[12] ESTÉVEZ-GARCÍA, Román, Jorge MARTÍN-GUTIÉRREZ, Saúl MENÉNDEZ, Rodríguez MARANTE, Pablo CHINEA-MARTÍN, Ovidia SOTO-MARTÍN and Moisés LODEIRO-. Open Data Motion Capture : MOCAP-ULL Database. *Procedia - Procedia Computer Science* [online]. 2015, **75**, 316–326. ISSN 1877-0509. Available at: doi:10.1016/j.procs.2015.12.253

[13] FERNANDO, Randima. *GPU Gems* [online]. B.m.: Addison-Wesley Professional, 2004. ISBN 978-0321228321. Available at: https://developer.nvidia.com/sites/all/modules/custom/gpugems/books/GPUGems/gpugems_ch23.html

[14] FICHTINGER, Gabor, Anton DEGUET, Ken MASAMUNE, Emese BALOGH, G.S. FISCHER, Herve MATHIEU, R.H. TAYLOR, S.J. ZINREICH and L.M. FAYAD. Image Overlay Guidance for Needle Insertion in CT Scanner. *IEEE Transactions on Biomedical Engineering* [online]. 2005, **52**(8), 1415–1424. ISSN 0018-9294. Available at: doi:10.1109/TBME.2005.851493

[15] FIĽOVÁ, PhDr. Petra and PH.D. *Kapitoly z pedagogické psychologie* [online]. B.m.: Brno University of Technology, 2015. Available at: https://www.vutbr.cz/www_base/priloha.php?dpid=88359

[16] FRYDMAN, Lucio, Adonis LUPULESCU and Tali SCHERF. Principles and Features of Single-Scan Two Dimensional NMR Spectroscopy. *Journal American Chemical Society* [online]. 2003, **125**(12), 9204–9217. Available at: https://www.researchgate.net/profile/Lucio_Frydman/publication/8346109_Principles_and_Features_of_Single-Scan_Two-Dimensional_NMR_Spectroscopy/links/54a4f7c80cf256bf8bb32aa8.pdf

[17] FURHT, Borko and Julie CARMIGNIANI. *Handbook of Augmented Reality* [online]. New York, NY: Springer New York, 2011. ISBN 978-1-4614-0063-9. Available at: doi:10.1007/978-1-4614-0064-6

[18] GELDERBLOM, Erik C., Hendrik J. VOS, Frits MASTIK, Telli FAEZ, Ying LUAN, Tom J A KOKHUIS, Antonius F W VAN DER STEEN, Detlef LOHSE, Nico DE JONG and Michel VERSLUIS. Brandaris 128 ultra-high-speed imaging facility: 10 years of operation, updates, and enhanced features. *Review of Scientific Instruments* [online]. 2012, **83**(10), 11. ISSN 00346748. Available at: doi:10.1063/1.4758783

[19] GOODSELL, David S. Visual Methods from Atoms to Cells. *Structure* [online]. 2005, **13**, 347–354. Available at: doi:10.1016/j.str.2005.01.012

[20] GOODSELL, David S. Putting Proteins in Context: Scientific illustrations bring together information from diverse sources to provide an integrative view of the molecular biology of cells. *Bioessays* [online]. 2013, **34**(9), 718–720. Available at: doi:10.1002/bies.201200072.Putting

[21] GOODSELL, David S. Bridging Boundaries in Molecular and Cellular Visualization. *BioZoom*. 2012, 6–8.

[22] GOODSELL, David S and Graham T JOHNSON. Filling in the Gaps : Artistic License in Education and Outreach. *PLoS Biology* [online]. 2007, **5**(12), 2759–2762. Available at: doi:10.1371/journal.pbio.0050308

[23] ILGEN, Jonathan S, Jonathan SHERBINO and David A COOK. Technology-enhanced Simulation in Emergency Medicine: A Systematic Review and Meta-Analysis. *Academic Emergency Medicine* [online]. 2013, **20**(2), 117–127. ISSN 10696563. Available at: doi:10.1111/acem.12076

[24] JADVAR, Hossein and Patrick M. COLLETTI. Competitive advantage of PET/MRI. *European Journal of Radiology* [online]. 2014, **83**(1), 84–94. ISSN 0720048X. Available at: doi:10.1016/j.ejrad.2013.05.028

[25] JANG, Susan, Jonathan M VITALE, Robert W JYUNG and John B BLACK. Direct manipulation is better than passive viewing for learning anatomy in a three-dimensional virtual reality environment. *Computers & Education* [online]. 2017, **106**, 150–165. ISSN 03601315. Available at: doi:10.1016/j.compedu.2016.12.009

[26] JOHNSON, Graham T and Samuel HERTIG. A guide to the visual analysis and communication of biomolecular structural data. *Nature Reviews Molecular Cell Biology* [online]. 2014, **15**(10), 690–698. ISSN 1471-0072. Available at: doi:10.1038/nrm3874

[27] KHAN, Asad Ullah and Raffaele ZARRILLI. *Multi Drug Resistance: A Global Concern* [online]. B.m.: Bentham e-books, 2012. ISBN 978-1-60805-255-4. Available at: doi:10.2174/97816080529291120101

[28] KHERLOPIAN, Armen R, Ting SONG, Qi DUAN, Mathew A NEIMARK, Ming J PO, John K GOHAGAN and Andrew F LAINE. A review of imaging techniques for systems biology. *BMC Systems Biology* [online]. 2008, **2**(1), 74. ISSN 1752-0509. Available at: doi:10.1186/1752-0509-2-74

[29] KING, Geoff. *Spectacle of the Real: From Hollywood to 'Reality' TV and Beyond*. 2005. ISBN 1841501204.

[30] KOSTER, Abraham J and Judith KLUMPERMAN. Electron microscopy in cell biology: integrating structure and function. *Nature reviews. Molecular cell biology* [online]. 2003, 6–10. ISSN 1471-0072. Available at: doi:10.1038/nrm1194

[31] KRISHNAMURTHY, Ravi, John W. WOODS and Pierre MOULIN. Frame interpolation and bidirectional prediction of video using compactly encoded optical-flow fields and label fields. *IEEE Transactions on Circuits and Systems for Video Technology* [online]. 1999, **9**(5), 713–726. ISSN 10518215. Available at: doi:10.1109/76.780361

[32] KUEHNEL, Wolfgang. *Color Atlas of Cytology, Histology,and Microscopic Anatomy*. 4th ed. B.m.: Thieme, 2003. ISBN 9781588901750.

[33] LEE, Yeonkyung and Hoon YOO. Low-cost 3D motion capture system using passive optical markers and monocular vision. *Optik - International Journal for Light and Electron Optics* [online]. 2017, **130**, 1397–1407. ISSN 00304026. Available at: doi:10.1016/j.ijleo.2016.11.174

[34] LOK, Corie. From monsters to molecules. *Nature*. 2011, **477**, 359–361.

[35] MCGILL, Gaël. Molecular Movies… Coming to a Lecture near You. *Cell* [online]. 2008, **133**, 1127–1132. Available at: doi:10.1016/j.cell.2008.06.013

[36] MCGILL, Gaël. *Villus Capillary - Your First Maya Scene from A-to-Z* [online]. Available at: http://www.molecularmovies.com/learning/

[37] MOORE, Simon C. and Mike OAKSFORD. *Emotional Cognition: From brain to behaviour* [online]. B.m.: John Benjamins Publishing Company, 2002. ISBN 9789027251688. Available at: doi:10.1075/aicr.44

[38] MPAA. *Theatrical market statistics* [online]. 2015. Available at: http://www.mpaa.org/wp-content/uploads/2016/04/MPAA-Theatrical-Market-Statistics-2015_Final.pdf

[39] PERES, Michael R. *The Focal Encyclopedia of Photography*. 4th ed. 2007. ISBN 978-0240807409.

[40] PERES, Michael R., Mark OSTERMAN, Grant B. ROMER, Nancy M. STUART and Tomas LOPEZ. *The Concise Focal Encyclopedia of Photography*. B.m.: Focal Press, 2008. ISBN 978-0-240-80998-4.

[41] PITATI, Bonifacio de. *Dives and Lazarus* [online]. Venice: Gallerie dell'Accademia. 1540. Available at: https://commons.wikimedia.org/wiki/File:Bonifacio_de_Pitati_-_Dives_and_Lazarus_-_WGA02417.jpg

[42] POPOVIĆ, Sinisa, Marko HORVAT, Davor KUKOLJA, Branimir DROPULJIĆ and Kresimir COSIĆ. Stress inoculation training supported by physiology-driven adaptive virtual reality stimulation. *Studies in health technology and informatics* [online]. 2009, **144**(1), 50–4. ISSN 0926-9630. Available at: doi:10.3233/978-1-60750-017-9-50

[43] RABIGER, Michael. *Directing Filme Techniques and aesthetics* [online]. 4th ed. B.m.: Focal Press, 2008. ISBN 9788578110796. Available at: doi:10.1017/CBO9781107415324.004

[44] READ, Paul and Mark-Paul MEYER. *Restoration of Motion Picture Film*. 1st ed. B.m.: Butterworth-Heinemann, 2000. ISBN 978-0750627931.

[45] REGER, Greg M. and Kevin M. HOLLOWAY. Effectiveness of Virtual Reality Exposure Therapy for Active Duty Soldiers in a Military Mental Health Clinic. *Journal of Traumatic Stress* [online]. 2010, **23**(2), 4. ISSN 08949867. Available at: doi:10.1002/jts.20510

[46] RIZZO, Albert, Jarrell PAIR, Ken GRAAP, Brian MANSON, Peter J MCNERNEY, Brenda WIEDERHOLD, Mark WIEDERHOLD and James SPIRA. A Virtual Reality Exposure Therapy Application for Iraq War Military Personnel with Post Traumatic Stress Disorder: From Training to Toy to Treatment. *NATO Advanced Research Workshop* [online]. 2006, **6**, 235–250. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.2415&rep=rep1&type=pdf

[47] ROSEN, Joseph and Gary BROOKER. Non-scanning motionless fluorescence three-dimensional holographic microscopy. *Nature Photonics* [online]. 2008, **2**(3), 190–195. ISSN 1749-4885. Available at: doi:10.1038/nphoton.2007.300

[48] SEYMOUR, Neal E, Anthony G GALLAGHER, Sanziana A ROMAN, Michael K O'BRIEN, Vipin K BANSAL, Dana K ANDERSEN and Richard M SATAVA. Virtual reality training improves operating room performance: results of a randomized, double-blinded study. *Annals of surgery* [online]. 2002, **236**(4), 458-63–4. ISSN 0003-4932. Available at: doi:10.1097/01.SLA.0000028969.51489.B4

[49] SHIGEMOTO, K. Weber-Fechner's Law and Demand Function. *Microeconomics* [online]. 2003, **1**(0), 6. Available at: http://arxiv.org/abs/physics/0303118

[50] SHIROMANI, P, T KEANE and J E LEDOUX. *Post-Traumatic Stress Disorder* [online]. Totowa, NJ: Humana Press, 2009. ISBN 978-1-60327-328-2. Available at: doi:10.1007/978-1-60327-329-9

[51] SIKOS, Leslie F. *Rich Semantics for Interactive 3D Models of Cultural Artifacts* [online]. 2016. Available at: doi:10.1007/978-3-319-49157-8_14

[52] SPENCER, Nik and Senior ILLUSTRATOR. Style and Substance: Visualising Science and the Development of Art at Nature. *BioZoom*. no date, **3**, 9–11.

[53] STERNBERG, Robert J. *Sternberg-kognitivni_psychologie.pdf*. B.m.: Portal, 2009. ISBN 978-80-7367-638-4.

[54] ULBRICH, Ed. *How Benjamin Button got his face* [online]. B.m.: TED Conferences. 2009. Available at: https://www.ted.com/talks/ed_ulbrich_shows_how_benjamin_button_got_his_face

[55] VAUGHAN, Neil, Bodgan GABRYS and Venketesh N. DUBEY. An overview of self-adaptive technologies within virtual reality training. *Computer Science Review* [online]. 2016, **22**, 65–87. ISSN 15740137. Available at: doi:10.1016/j.cosrev.2016.09.001

[56] VELTEN, Andreas, Ramesh RASKAR, Di WU, Adrian JARABO, Belen MASIA, Christopher BARSI, Chinmaya JOSHI, Everett LAWSON, Moungi BAWENDI and Diego GUTIERREZ. Femto-Photography: Capturing and Visualizing the Propagation of Light. *ACM Transactions on Graphics* [online]. 2013, **32**(4), 9. ISSN 07300301. Available at: doi:10.1145/2461912.2461928

[57] WU, Xiongwu and Bernard R BROOKS. Structure and Dynamics of Macromolecular Assemblies from Electron Microscopy Maps. In: *Modern Electron Microscopy in Physical and Life Sciences* [online]. B.m.: InTech, 2016, p. 21. Available at: doi:10.5772/62085

[58] YU, Hua-yin, Nathanael D. HEVELONE, Stuart R. LIPSITZ, Keith J. KOWALCZYK and Jim C. HU. Use, Costs and Comparative Effectiveness of Robotic Assisted, Laparoscopic and Open Urological Surgery. *The Journal of Urology* [online]. 2012, **187**(4), 1392–1399. ISSN 00225347. Available at: doi:10.1016/j.juro.2011.11.089

[59] ZOPPÈ, Monica, Yuri POROZOV, Raluca ANDREI, Stefano CIANCHETTA, Maria Francesca ZINI, Tiziana LONI and Marco CALLIERI. Using Blender for molecular animation and scientific representation. In: *Blender Conference* [online]. 2008, p. 1–6. Available at: http://www.bioblender.eu/Database/blender_zoppe_et_al3.pdf

[60] Methods for Determining Atomic Structures. *PDB-101* [online]. 2016. Available at: http://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/methods-for-determining-structure

[61] *3D Modelling* [online]. B.m.: Wikimedia Foundation, 2016. Available at: https://en.wikipedia.org/wiki/3D_modeling

[62] Worldwide Protein Data Bank. *wwPDB Foundation* [online]. 1971. Available at: http://www.wwpdb.org/

[63] *How to Use Color in Film*. B.m.: StudioBinder, 2016.