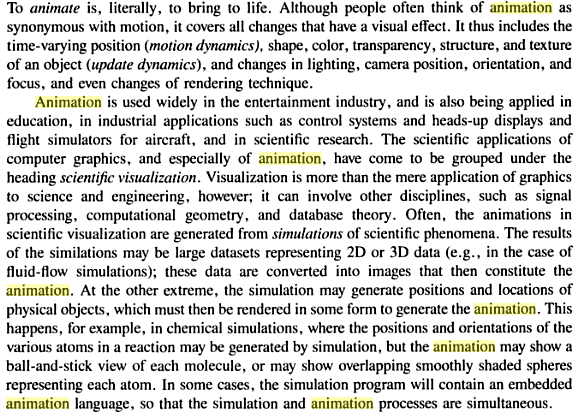
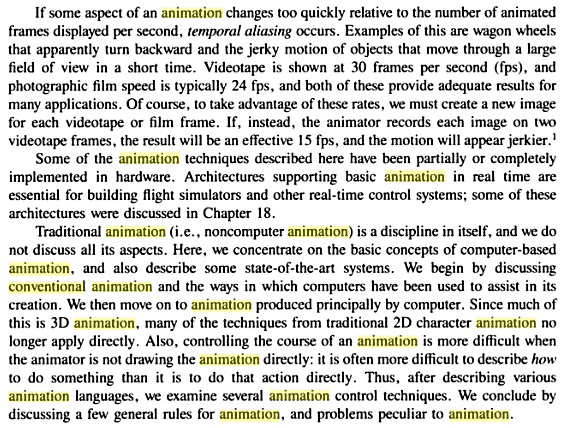
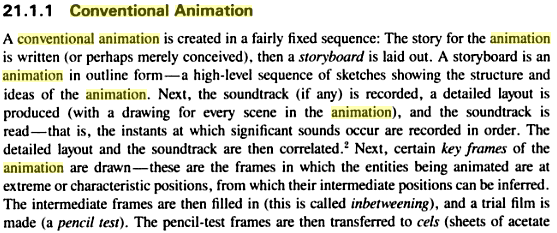
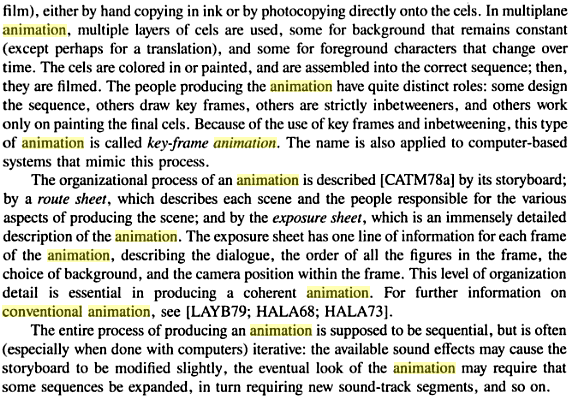
**ANIMATION**

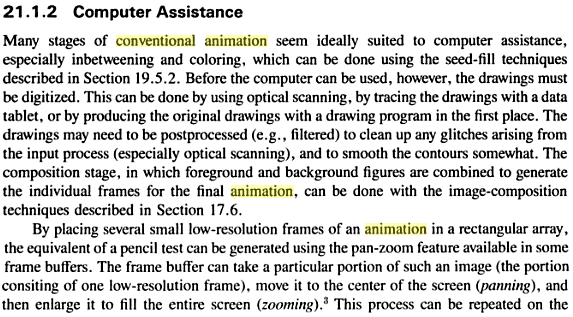


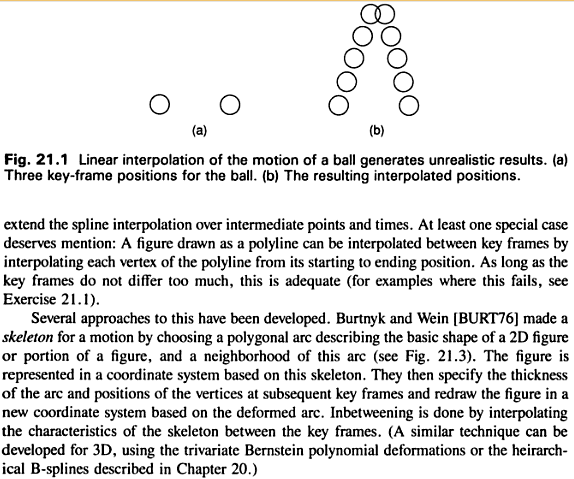


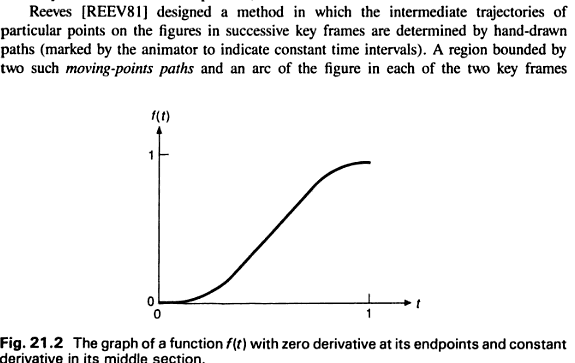
**Conventional and computer based animation**

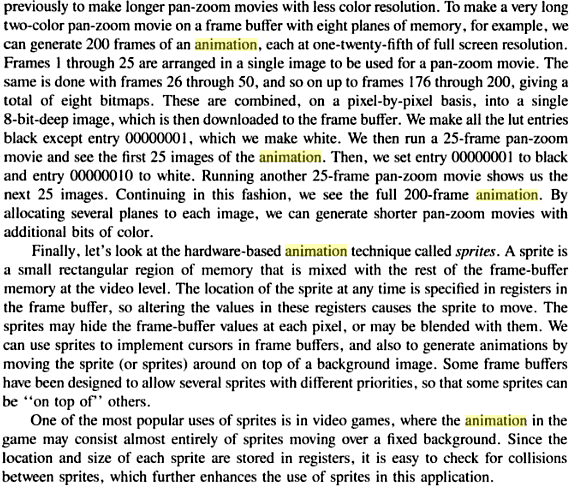












**Methods of Controlling Animations**

* Explicitly Declared
* Procedural
* Constraint-Based
* Analyzing Live Action-Based
* Kinematic and Dynamic

Explicitly Declared:

All events that could occur in an animation are declared. This can be done at the

* + object level by specifying simple transformations (translations, rotations, scaling) to objects
  + frame level by specifying key frames and methods for interpolating between them.

Procedural:

Based on communication among different objects whereby each object obtains knowledge about the static/dynamic properties of other objects.

* + Can be used to ensure consistency

Constraint-Based:

* Many objects movements in the real world are determined by other objects which they come in contact with
  + E.g. presence of strong wind or fast moving large objects
* Instead of explicit declaration, constraints based on the environment can be used to control objects’ motion.
* Example Systems: Sketchpad and ThingLab.

Analyzing Live Action-Based:

* Control is achieved by examining the motions of objects in the real world.
  + Rotoscoping: is a technique where animators trace live action movement, frame by frame, for use in animated films.
    - Originally, pre-recorded live-film images were projected onto a frosted glass panel and redrawn by an animator.
      * This projection equipment is called a Rotoscope.
* Another way is to attach indicators to key points on the body of a human actor.
  + For example the data glove [gesture language for hearing-impaired people]

Kinematic and Dynamic:

* Kinematics refer to the position and velocity of points
  + “The cube is at the origin at time t = 0. Thereafter, it moves with constant acceleration in the direction (1 meter, 1 meter, 5 meters)”
* Dynamics takes into account the physical laws that govern kinematics
  + Newton laws for the movement of large objects
  + Euler-Lagrange equations for fluids
  + A particle moves with an acceleration proportional to the forces acting on it.

For example: “At time t = 0, the cube is at position (0 meter, 100 meter, 0 meter). The cube has a mass of 100 grams. The force of gravity acts on the cube.

**Basic guidelines of animation**

The 12 principles:

### Squash and stretch

the most important principle is "[squash and stretch](http://en.wikipedia.org/wiki/Squash_and_stretch)",[[4]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-Johnston-4) the purpose of which is to give a sense of weight and flexibility to drawn objects. It can be applied to simple objects, like a bouncing ball, or more complex constructions, like the musculature of a human face.[[5]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-5)[[6]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-6) Taken to an extreme point, a figure stretched or squashed to an exaggerated degree can have a comical effect.[[7]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-7) In realistic animation, however, the most important aspect of this principle is the fact that an object's volume *does not* change when squashed or stretched. If the length of a ball is stretched vertically, its width (in three dimensions, also its depth) needs to contract correspondingly horizontally.[[8]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-8)

### Anticipation

[Anticipation](http://en.wikipedia.org/wiki/Anticipation_(animation)) is used to prepare the audience for an action, and to make the action appear more realistic.[[9]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-9) A dancer jumping off the floor has to bend his knees first; a golfer making a swing has to swing the club back first. The technique can also be used for less physical actions, such as a character looking off-screen to anticipate someone's arrival, or attention focusing on an object that a character is about to pick up.[[10]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-10)

For special effect, anticipation can also be omitted in cases where it is expected. The resulting sense of anticlimax will produce a feeling of surprise in the viewer, and can often add comedy to a scene.[[11]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-11)This is often referred to as a 'surprise gag'.[[12]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-12)

### Staging

This principle is akin to [staging](http://en.wikipedia.org/wiki/Staging_(theatre)) as it is known in theatre and film.[[13]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-J.26T53-13) Its purpose is to direct the audience's attention, and make it clear what is of greatest importance in a scene;[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) Johnston and Thomas defined it as "the presentation of any idea so that it is completely and unmistakably clear", whether that idea is an action, a personality, an expression or a mood.[[13]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-J.26T53-13) This can be done by various means, such as the placement of a character in the frame, the use of light and shadow, and the angle and position of the camera.[[15]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-15)The essence of this principle is keeping focus on what is relevant, and avoiding unnecessary detail.[[16]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-16)[[17]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-IPA_S-17)

### Straight ahead action and pose to pose

These are two different approaches to the actual drawing process. "Straight ahead action" means drawing out a scene frame by frame from beginning to end, while "pose to pose" involves starting with drawing a few key frames, and then filling in the intervals later.[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) "Straight ahead action" creates a more fluid, dynamic illusion of movement, and is better for producing realistic action sequences. On the other hand, it is hard to maintain proportions, and to create exact, convincing poses along the way. "Pose to pose" works better for dramatic or emotional scenes, where composition and relation to the surroundings are of greater importance.[[18]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-18) A combination of the two techniques is often used.[[19]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-IPA_SAA.26PtP-19)

Computer animation removes the problems of proportion related to "straight ahead action" drawing; however, "pose to pose" is still used for computer animation, because of the advantages it brings in composition.[[20]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-20) The use of computers facilitates this method, as computers can fill in the missing sequences in between poses automatically. It is, however, still important to oversee this process and apply the other principles discussed.[[19]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-IPA_SAA.26PtP-19)

### Follow through and overlapping action

[Follow through and overlapping action](http://en.wikipedia.org/wiki/Follow_through_and_overlapping_action) is a general heading for two closely related techniques which help to render movement more realistically, and help to give the impression that characters follow the [laws of physics](http://en.wikipedia.org/wiki/Physical_law). "Follow through" means that separate parts of a body will continue moving after the character has stopped. "Overlapping action" is the tendency for parts of the body to move at different rates (an arm will move on different timing of the head and so on). A third related technique is "drag", where a character starts to move and parts of him take a few frames to catch up.[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) These parts can be inanimate objects like clothing or the antenna on a car, or parts of the body, such as arms or hair. On the human body, the torso is the core, with arms, legs, head and hair appendices that normally follow the torso's movement. Body parts with much tissue, such as large stomachs and breasts, or the loose skin on a dog, are more prone to independent movement than bonier body parts.[[21]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-21) Again, exaggerated use of the technique can produce a comical effect, while more realistic animation must time the actions exactly, to produce a convincing result.[[22]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-22)

Thomas and Johnston also developed the principle of the "moving hold". A character not in movement can be rendered absolutely still; this is often done, particularly to draw attention to the main action. According to Thomas and Johnston, however, this gave a dull and lifeless result, and should be avoided. Even characters sitting still can display some sort of movement, such as the torso moving in and out with breathing.[[23]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-23)

### Slow in and slow out[[edit](http://en.wikipedia.org/w/index.php?title=12_basic_principles_of_animation&action=edit&section=7" \o "Edit section: Slow in and slow out)]

The movement of the human body, and most other objects, needs time to accelerate and slow down. For this reason, animation looks more realistic if it has more drawings near the beginning and end of an action, emphasizing the extreme poses, and fewer in the middle.[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) This principle goes for characters moving between two extreme poses, such as sitting down and standing up, but also for inanimate, moving objects, like the bouncing ball in the above illustration.[[24]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-IPA_SI.26SO-24)

### Arcs[[edit](http://en.wikipedia.org/w/index.php?title=12_basic_principles_of_animation&action=edit&section=8" \o "Edit section: Arcs)]

Most natural action tends to follow an arched [trajectory](http://en.wikipedia.org/wiki/Trajectory), and animation should adhere to this principle by following implied "arcs" for greater realism. This can apply to a limb moving by rotating a joint, or a thrown object moving along a [parabolic](http://en.wikipedia.org/wiki/Parabola) trajectory. The exception is mechanical movement, which typically moves in straight lines.[[25]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-25)

As an object's speed or momentum increases, arcs tend to flatten out in moving ahead and broaden in turns. In baseball, a fastball would tend to move in a straighter line than other pitches; while a figure skater moving at top speed would be unable to turn as sharply as a slower skater, and would need to cover more ground to complete the turn.

An object in motion that moves out of its natural arc for no apparent reason will appear erratic rather than fluid. Therefore when animating (for example) a pointing finger, the animator should be certain that in all drawings in between the two extreme poses, the fingertip follows a logical arc from one extreme to the next. Traditional animators tend to draw the arc in lightly on the paper for reference, to be erased later.

### Secondary action[[edit](http://en.wikipedia.org/w/index.php?title=12_basic_principles_of_animation&action=edit&section=9" \o "Edit section: Secondary action)]

Adding secondary actions to the main action gives a scene more life, and can help to support the main action. A person walking can simultaneously swing his arms or keep them in his pockets, he can speak or whistle, or he can express emotions through facial expressions.[[26]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-26) The important thing about secondary actions is that they emphasize, rather than take attention away from, the main action. If the latter is the case, those actions are better left out.[[27]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-PoA_SA-27) In the case of facial expressions, during a dramatic movement these will often go unnoticed. In these cases it is better to include them at the beginning and the end of the movement, rather than during.[[28]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-28)

### Timing[[edit](http://en.wikipedia.org/w/index.php?title=12_basic_principles_of_animation&action=edit&section=10" \o "Edit section: Timing)]

Timing refers to the number of drawings or frames for a given action, which translates to the speed of the action on film.[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) On a purely physical level, correct timing makes objects appear to obey the laws of physics; for instance, an object's weight determines how it reacts to an impetus, like a push.[[29]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-PoA_T-29) Timing is critical for establishing a character's mood, emotion, and reaction.[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) It can also be a device to communicate aspects of a character's personality.[[30]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-30)

### Exaggeration[[edit](http://en.wikipedia.org/w/index.php?title=12_basic_principles_of_animation&action=edit&section=11" \o "Edit section: Exaggeration)]

Exaggeration is an effect especially useful for animation, as perfect imitation of reality can look static and dull in cartoons.[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) The level of exaggeration depends on whether one seeks realism or a particular style, like a caricature or the style of an artist. The classical definition of exaggeration, employed by Disney, was to remain true to reality, just presenting it in a wilder, more extreme form.[[31]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-31) Other forms of exaggeration can involve the supernatural or surreal, alterations in the physical features of a character, or elements in the storyline itself.[[32]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-32) It is important to employ a certain level of restraint when using exaggeration; if a scene contains several elements, there should be a balance in how those elements are exaggerated in relation to each other, to avoid confusing or overawing the viewer.[[33]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-PoA_E-33)

### Solid drawing[[edit](http://en.wikipedia.org/w/index.php?title=12_basic_principles_of_animation&action=edit&section=12" \o "Edit section: Solid drawing)]

The principle of [solid](http://en.wikipedia.org/wiki/Solid_geometry) drawing means taking into account forms in three-dimensional space, giving them volume and weight.[[14]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-AT-14) The animator needs to be a skilled [draughtsman](http://en.wikipedia.org/wiki/Draughtsman) and has to understand the basics of three-dimensional shapes, anatomy, weight, balance, light and shadow, etc.[[34]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-34) For the classical animator, this involved taking art classes and doing sketches from life.[[35]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-IPA_SD-35) One thing in particular that Johnston and Thomas warned against was creating "twins": characters whose left and right sides mirrored each other, and looked lifeless.[[36]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-36) Modern-day computer animators draw less because of the facilities computers give them, yet their work benefits greatly from a basic understanding of animation principles, and their additions to basic computer animation.[[35]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-IPA_SD-35)

### Appeal[[edit](http://en.wikipedia.org/w/index.php?title=12_basic_principles_of_animation&action=edit&section=13" \o "Edit section: Appeal)]

Appeal in a cartoon character corresponds to what would be called [charisma](http://en.wikipedia.org/wiki/Charisma) in an actor.[[37]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-J.26T68-37) A character who is appealing is not necessarily sympathetic – villains or monsters can also be appealing – the important thing is that the viewer feels the character is real and interesting.[[37]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-J.26T68-37) There are several tricks for making a character connect better with the audience; for likable characters a symmetrical or particularly baby-like face tends to be effective.[[38]](http://en.wikipedia.org/wiki/12_basic_principles_of_animation#cite_note-38) A complicated or hard to read face will lack appeal, it may more accurately be described as 'captivation' in the composition of the pose, or the character design.

**Animation languages: Introduction to OpenGL ES**

### What is OpenGL?

OpenGL stands for Open Graphics Library. It is a specification of an API for rendering graphics, usually in 3D. OpenGL implementations are libraries that implement the API defined by the specification.

Graphics cards usually have an OpenGL implementation. Because the OpenGL specification is not platform-specific, it is possible to write an application that will be possible to use against many different types of graphics cards. It also increases the chance that the application will continue to work when new hardware will become available.

### What is NOT OpenGL?

The OpenGL API *only* deals with rendering graphics. OpenGL does not provide functions for animations, timing, file IO, image file format processing, GUI, and so forth. OpenGL is concerned only about rendering.

GLUT is not OpenGL. It is not a part of OpenGL; it is simply a library that is used by some users to create an OpenGL window.

### Who maintains the OpenGL specification?

The specification is maintained by [OpenGL Architectural Review Board](http://www.opengl.org/wiki/OpenGL_Architectural_Review_Board) or ARB.

### Is OpenGL Open Source?

No, OpenGL doesn't have any source code. GL is a specification which can be found on this website. It describes the interface the programmer uses and expected behavior. OpenGL is an open *specification*. Anyone can download the spec for free. This is as opposed to ISO standards and specifications, which cost money to access.

There is an implementation of GL that is Open Source and it is called [Mesa3D](http://www.mesa3d.org/) It [announces itself](http://www.mesa3d.org/intro.html) as implementing [OpenGL 3.0](http://www.opengl.org/wiki/History_of_OpenGL#OpenGL_3.0_.282008.29) and[GLSL](http://www.opengl.org/wiki/GLSL) 1.30.

### Where can I download OpenGL?

Just like the "Open Source?" section explains, OpenGL is not a software product. it is a specification.

On Mac OS X, Apple's OpenGL implementation is included.

On Windows, companies like nVidia and AMD/ATI use the spec to write their own implementation, so OpenGL is included in the drivers that they supply. For laptop owners, however, you'll need to visit the manufacturer of your laptop and download the drivers from them.

### Where can I download OpenGL?

Updating your graphics drivers is usually enough to get the latest OpenGL implementation for your graphics hardware. This is sufficient for those who want to use applications that require OpenGL.

For programmers, installing drivers is generally insufficient. You will need to load the OpenGL function pointers, either [manually](http://www.opengl.org/wiki/Load_OpenGL_Functions) or[automatically with a library](http://www.opengl.org/wiki/Extension_Loading_Library). More information on this can be found in the [Getting started](http://www.opengl.org/wiki/Getting_started) page.

### Is there an OpenGL SDK?

There is no actual OpenGL SDK. There is a collection of websites, some (outdated) documentation, and links to tutorials, all found[here](http://www.opengl.org/wiki/Getting_started#SDK). But it is not an SDK of the kind you are thinking about.

NVIDIA and ATI have their own SDKs, both of which have various example code for OpenGL.

### What platforms have GL?

* Windows: 95 and above
* Mac OSX: all versions
* Linux: OpenGL is provided by open source drivers and MESA library, or by proprietary drivers.
* FreeBSD: OpenGL is provided by open source drivers and MESA library or proprietary Nvidia drivers.

OpenGL ES is often supported on embedded systems, but OpenGL ES is a different API from regular OpenGL.

**OpenGL ES:**

OpenGL® ES is a royalty-free, cross-platform API for full-function 2D and 3D graphics on embedded systems - including consoles, phones, appliances and vehicles. It consists of well-defined subsets of desktop OpenGL, creating a flexible and powerful low-level interface between software and graphics acceleration. OpenGL ES includes profiles for floating-point and fixed-point systems and the EGL™ specification for portably binding to native windowing systems. OpenGL ES 1.X is for fixed function hardware and offers acceleration, image quality and performance. OpenGL ES 2.X enables full programmable 3D graphics. OpenGL SC is tuned for the safety critical market.

OpenGL® ES is a low-level, lightweight API for *advanced embedded graphics* using well-defined subset profiles of OpenGL. It provides a low-level applications programming interface (API) between software applications and hardware or software graphics engines.

This standard 3D graphics API for embedded systems makes it easy and affordable to offer a variety of advanced 3D graphics and games across all major mobile and embedded platforms. Since OpenGL ES (OpenGL for Embedded Systems) is based on OpenGL, no new technologies are needed. This ensures synergy with, and a migration path to and from desktop OpenGL -- the most widely adopted cross-platform graphics API.

**Developer Advantages**

* **Industry Standard and Royalty Free**  
  Anyone can download the OpenGL ES specification and implement and ship products based on OpenGL ES. With broad industry support, OpenGL ES is the only truly open, vendor-neutral, multi-platform embedded graphics standard. The standardized higher level of abstraction that it offers means developers can concentrate more on content and less on the minor code and platform details.
* **Small footprint & low power consumption**  
  The embedded space varies widely, ranging from 400Mhz PDAs with 64MB RAM to 50MHz mobile phones with 1 MB RAM. OpenGL ES is designed to accommodate these differences by requiring a minimum footprint with minimum data storage requirements, minimized instruction/data traffic, and is both integer and floating point friendly. For users this means smaller binaries to download that take up less storage on the device.
* **Seamless transition from software to hardware rendering**  
  Although the OpenGL ES specification defines a particular graphics processing pipeline, individual calls can be executed on dedicated hardware, run as software routines on the system CPU, or implemented as a combination of both dedicated hardware and software routines. This means that software developers can ship a conformant software 3D engine today, that lets applications and tools seamlessly transition over to using OpenGL ES hardware-acceleration in higher powered devices.
* **Extensible & Evolving**  
  OpenGL ES allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism and for the API to be easily updated. As extensions become widely accepted, they are considered for inclusion into the core OpenGL ES standard. This process allows OpenGL ES to evolve in a controlled yet innovative manner.
* **Easy to use**  
  Based on OpenGL, OpenGL ES is well structured with an intuitive design and logical commands.
* **Well-documented**  
  Because OpenGL ES is based on OpenGL, there are numerous relevant books, and a great deal of relevant sample code, making information about OpenGL ES inexpensive and easy to find. With the introduction of OpenGL ES, a developer can now write basically the same code for cell phones to supercomputers.

**Conformance Testing**

Specifications are reviewed and revised on a yearly basis. To label an implementation as an OpenGL ES compliant implementation, the implementation must pass a set of [conformance tests](http://www.khronos.org/opengles/adopters/). The conformance test definitions are maintained as a separate part of the OpenGL ES documentation.

**The OpenGL ES Two Track Roadmap**

The OpenGL ES roadmap has been tailored to the diverse needs of the embedded industry and contains two tracks with "1.X" and "2.X" specification roadmaps that will evolve in parallel. The 1.X roadmap will continue to be developed for new-generation fixed function 3D accelerators while the 2.X roadmap will enable emerging programmable 3D pipelines. This dual-track roadmap enables OpenGL ES to meet the graphics requirements of a huge range of 3D enabled device and platforms in embedded markets - from low-end cell-phones to high-end gaming consoles. Khronos is committed to providing backwards compatibility between successive versions of the APIs in each of the 1.X and 2.X tracks to ensure that applications can be trivially ported from one version to the next.

* **Profiles:**   
  The OpenGL ES specification includes the definition of several profiles. Each profile is a subset of a version of the desktop OpenGL specification plus some additional OpenGL ES-specific extensions. The OpenGL ES profiles are part of a wider family of OpenGL-derived application programming interfaces. As such, the profiles share a similar processing pipeline, command structure, and the same OpenGL name space. Where necessary, extensions are created to augment the existing desktop OpenGL functionality. OpenGL ES-specific extensions play a role in OpenGL ES profiles similar to that played by OpenGL ARB extensions relative to the OpenGL specification. OpenGL ES-specific extensions are either precursors of functionality destined for inclusion in future core profile revisions, or formalization of important but non-mainstream functionality. Each profile definition implies a distinct header file and link/runtime library defining the commands and tokens in the profile. To simplify maintenance a single superset header can be defined with appropriate conditional preprocessing directives to control the visibility of tokens and command prototypes. At run-time an application can determine which profile is running using the OpenGL version string query.
* ***The Common Profile is intended for consumer entertainment***and related devices such as telephone handsets, PDAs, set-top boxes, game consoles, etc. It addresses the broadest range of the market including support for platforms with varying capability.
  + Minimum footprint full function 3D with texture-mapping
  + Good gaming platform
  + Implementable on cell phones
* ***The Safety Critical Profile is intended for consumer and industrial applications*** where reliability and certifiability are the primary constraints.
  + Absolute minimum 3D to ease safety certifications
  + Used in avionics and automotive displays
* **Extensions**:  
  OpenGL ES implementations may include extensions that add new features to the implementation. An OpenGL ES profile consists of two parts: a subset of the full OpenGL pipeline, and some extended functionality that is drawn from a set of OpenGL ES-specific extensions to the full OpenGL specification. Each extension is pruned to match the profile's command subset and added to the profile as either a core addition or a profile extension. Core additions differ from profile extensions in that the commands and tokens do not include extension suffixes in their names. Profile extensions are further divided into required (mandatory) and optional extensions. Required extensions must be implemented as part of a conforming implementation, whereas the implementation of optional extensions is left to the discretion of the implementor.
* **Native Platform Graphics Interface Layer - EGL:**  
  OpenGL ES also includes a specification of a common platform interface layer, called EGL. This layer is platform independent and may optionally be included as part of a vendor's OpenGL ES distribution. The platform binding also has an associated conformance test. Alternatively, a vendor may choose to define their own platform-specific embedding layer.

**Implementers of OpenGL ES**

* Fujitsu Limited
* NVIDIA Corporation
* Intel
* Sony Computer Entertainment Inc.
* Apple, Inc.
* QUALCOMM
* Vivante Corporation
* NVIDIA
* ITRI
* Digital Media Professionals
* Marvell
* ARM
* Imagination Technologies
* Creative Technology Ltd
* MediaTek Inc
* NOKIA OYJ
* Panasonic
* Broadcom Corporation
* Google, Inc.
* HISILICON TECHNOLOGIES CO.,LTD.