

EXPLOITING FRAME TO FRAME COHERENCY IN A SORT-MIDDLE ARCHITECTURE

Background

[0001] This relates to graphics processing.

[0002] It is common to use a sort-middle architecture for a mobile device. As in any graphics application, it is important to reduce power usage.

[0003] Geometry processing may be made parallel by assigning to each processor a subset of the primitives in a scene. Rasterization may be made parallel by assigning to each processor some portion of the necessary pixel calculations. Rendering involves calculating the effect of each primitive on each pixel. A primitive can fall anywhere on or off the actual viewing screen. Rendering sorts the primitives to the screen. The sort can take place during geometry processing in what is called sort-first, between geometry processing and rasterization, called sort-middle, or during rasterization called sort-last. Sort-first means redistributing raw primitives before their screen-space parameters are known, sort-middle means redistributing screen-space primitives, and sort-last means redistributing pixels, samples, or pixel fragments.

Brief Description Of The Drawings

[0004] Some embodiments are described with respect to the following figures:

Figure 1 is a schematic depiction for one embodiment;

Figure 2 is a flow chart for one embodiment;

Figure 3 is a system depiction for one embodiment; and

Figure 4 is a front elevational view for one embodiment.

Detailed Description

[0005] Pixel values that were computed in a previous frame may be reused for the current frame, operating in a sort-middle architecture. A hash or some other compact representation of all the data used in a tile, including all triangles, uniforms, textures, shaders, etc. is computed and stored for each tile. When rendering the

next frame, that compact representation is once again computed for each tile. In a sort-middle architecture, there is a natural break point just before rasterization. At this break point, the compact representation may be compared to the compact representation computed in the previous frame for the same tile. If those compact representations are the same, then there is no need to render anything for this tile. Instead, the contents of the color buffer or other buffers of the previous frame of the tile may be moved, or otherwise re-used, to the same buffer of the tile for the current frame.

[0006] This technique works before rasterization and pixel shading and can be applied to other buffers, in addition to the color buffer.

[0007] In a sort-middle architecture, also sometimes called a tiling architecture, the screen is divided into tiles, which are non-overlapping rectangular regions that, together, cover the entire screen. In a first phase, all primitives, such as triangles that are to be rendered, are sorted into the tiles, so that a reference to a primitive is stored in a primitive list per tile for all tiles that the primitive overlaps. When all sorting is done, each tile can be rasterized serially or in parallel.

[0008] This technique exploits the fact that often large parts of an image are the same from one image to the next. This effect is often referred to as frame to frame coherency. Frame to frame coherency is exploited here to make the graphics processor more efficient from both a power and performance perspective.

[0009] In one embodiment, a compact representation, such as a hash, is accumulated per tile when the primitive list per tile is created. The compact representation may be based on all geometry, input data, and the relevant states, including all primitives, uniforms, textures, shaders, blending modes, etc. The compact representations from the previous frame are stored in memory when rendering the current frame. Then, just before per tile rasterization starts, a determination is made whether the compact representation of the current tile for the current frame is the same as the compact representation of the same tile for the previous frame. If the compact representations are the same, then the rendered content is also assumed to be the same and, consequently, no rendering is needed.

Instead, the colors of all the pixels in the tile are reused from the previous frame. If the compact representations are not the same, then rasterization, depth tests, pixel shade, and blend are done as usual.

[0010] This technique may be used for all render targets and for depth and stencil as well, in addition to the color buffer. However, for the current render targets, such as the color buffer that will be displayed on the screen, some further optimizations are possible. When using, for example, the embedded display port (eDP) specification versions 1.4/1.5 (and higher), these techniques may be used with partial-frame updates while in the panel soft refresh (PSR) mode. Basically, the algorithm detects that a certain tile for the current frame is the same as the corresponding tile for the previous frame and, hence, rasterization and all subsequent steps are not needed. Therefore, when the data is to be transferred to the display, sending data may be avoided for those tiles that are the same as in the previous frame. Finally, when the frame ends, compact representations for the current frame have to be moved to the previous frame compact representation table and are cleared for future usages. Alternatively, one may only swap the a pointer to the current frame and the previous frame. To avoid memory bursts, a lazy replacement can be considered, adding a single bit to identify the current frame.

[0011] The chosen compact representation may comply with one or more of the following criteria: (1) allow processing blocks sequentially, in order to accumulate the overall hash value per tile; (2) minimize false positives, since they could generate visible artifacts; and (3) generate moderately sized hash values, to reduce the storage overhead of the hash table.

[0012] Given the fact that the output hash values per tiles are generated just for comparison purposes, any simple cyclic redundancy check or check sum function may suffice, in some embodiments. If a smaller probability of hash collision is desired, even a cryptographic hash function using the Merkle-Damgaard construction, such as MD5, may be used at the expense of heavier computation requirements.

[0013] In addition, instead of storing a hash, an identifier may be all that is stored for the draw call that fully covered the tile at the end of the frame. If that draw call is static because all uniforms, textures, and shaders, etc. stay the same until the following frame, and it is discovered that the same draw call will cover the tile entirely again, then all processing of that tile may be avoided and the contents from the previous frame reused. For this to work, one could, for example, test if there is a single triangle that overlaps a tile completely, and test if all other triangles in the tile's primitive list are occluded by that triangle. Another way is to determine if a tile is completely covered by a subset of the triangles in a draw call in the current frame, and in the following frame, if the same draw call is exactly the same and all other geometry in that tile are occluded, then nothing needs to be drawn for that tile.

[0014] In still another embodiment, a bloom filter may be used instead of a hash. For that implementation, a minimum of two bloom filters may be used, even though two N smaller bloom filters can also be considered. Each pair of bloom filters comprises the elements for different kinds of information. One pair may could be for geometry and another pair may be for texture addresses and a third one may be for shaders, for example. Out of these pairs, one contains the elements for the previous frame and the second one contains the corresponding elements for the current frame.

[0015] The bloom filter algorithm works like the one with hashes, in one embodiment. When the triangle list is created, several hash functions are applied to the information to compare. The results of these hash functions are used to index both bloom filters. For the one related to the current frame, all bits indexed are set. On the other hand, for the bloom filter of the previous frame, the bits indexed are read and a check determines whether all of them are one. If so, the rendered content is assumed to be the same and the colors are reused from the previous frame. If not, the sequence proceeds normally. Once the frame finishes, the usage of the bloom filters swaps. The one pointing to the current frame becomes the previous frame, and the former previous frame is cleared.

[0016] One advantage of using bloom filters compared to a hash is that, usually, bloom filters require less storage space to keep all the information, although they

may require extra logic for the multiple hash functions. Traditionally, the number of bits required to get a false positive ratio of about 0.5 percent is in the range of eleven to twelve bits per element in the bloom filter and the optimal number of hash functions for this false positive ratio is in the range of six to eight functions.

[0017] Referring to Figure 1, the tiling engine 10 may be coupled to a raster pipeline 12. The tiling engine 10 receives geometry, state information, shader information, and texture addresses in a hash or bloom filter computation unit 13. Any state information is provided to a tile list builder 14 that combines the pipeline result with any state information. This combined data is stored in the tile cache 22.

[0018] The previous frame hashes or bloom filter storage 16 is dumped into the current frame hashes/bloom filter storage 18 at frame change. A tile fetcher 20 fetches tiles from the rasterization pipeline 12 and provides them to a tile cache 22. The tile cache 22 transfers data to the unified cache 24.

[0019] The raster pipeline 12 includes a rasterizer 26 coupled to the tile fetcher 20 and a depth test unit 28. A plurality of shaders may be coupled to the depth test unit 28. The shaders 30 are coupled to a color buffer 32, a texture sampler 34, and a depth buffer 36, which, in turn, are coupled to the unified cache 24.

[0020] A sequence 40, shown in Figure 2, may be implemented in software, firmware and/or hardware. In software and firmware embodiments, it may be implemented by computer executed instructions stored in one or more non-transitory computer readable medium, such as magnetic, optical, or semiconductor storage.

[0021] The sequence begins in the tile list builder 14, which performs tile binning and also determines the tile identifiers that are affected for the input data, as indicated in block 42. Then, the hash or bloom filter is computed for the input data in block 44. For all affected tiles, the current hash or Bloom filter is read and the computed hash or filter is accumulated, as indicated in block 46. Then, a check at diamond 48 determines whether the tile binning process is finished. If not, the flow iterates back to block 42. If so, the flow proceeds to tile fetcher 20.

[0022] In block 50, the next tile identifier to fetch is obtained. For that tile, the current hash or filter is obtained and the ones from the previous frame are obtained

in block 52. A check at 54 determines whether the current and previous hash or filters match. If so, at 56, tile fetching is skipped and rasterization may be avoided for the current tile. If not, at 58, proceed with tile fetching and direct data to raster pipeline for processing. Then, at 60, it is determined if there is any other tile identifier to fetch. If so, the flow iterates back to block 50. Otherwise, at 62, the current hashes or filter values become the previous hashes or filter values.

[0023] Figure 3 illustrates an embodiment of a system 700. In embodiments, system 700 may be a media system although system 700 is not limited to this context. For example, system 700 may be incorporated into a personal computer (PC), laptop computer, ultra-laptop computer, tablet, touch pad, portable computer, handheld computer, palmtop computer, personal digital assistant (PDA), cellular telephone, combination cellular telephone/PDA, television, smart device (e.g., smart phone, smart tablet or smart television), mobile internet device (MID), messaging device, data communication device, and so forth.

[0024] In embodiments, system 700 comprises a platform 702 coupled to a display 720. Platform 702 may receive content from a content device such as content services device(s) 730 or content delivery device(s) 740 or other similar content sources. A navigation controller 750 comprising one or more navigation features may be used to interact with, for example, platform 702 and/or display 720. Each of these components is described in more detail below.

[0025] In embodiments, platform 702 may comprise any combination of a chipset 705, processor 710, memory 712, storage 714, graphics subsystem 715, applications 716 and/or radio 718. Chipset 705 may provide intercommunication among processor 710, memory 712, storage 714, graphics subsystem 715, applications 716 and/or radio 718. For example, chipset 705 may include a storage adapter (not depicted) capable of providing intercommunication with storage 714.

[0026] Processor 710 may be implemented as Complex Instruction Set Computer (CISC) or Reduced Instruction Set Computer (RISC) processors, x86 instruction set compatible processors, multi-core, or any other microprocessor or central processing unit (CPU). In embodiments, processor 710 may comprise dual-core

processor(s), dual-core mobile processor(s), and so forth. The processor may implement the sequence of Figure 2 together with memory 712.

[0027] Memory 712 may be implemented as a volatile memory device such as, but not limited to, a Random Access Memory (RAM), Dynamic Random Access Memory (DRAM), or Static RAM (SRAM).

[0028] Storage 714 may be implemented as a non-volatile storage device such as, but not limited to, a magnetic disk drive, optical disk drive, tape drive, an internal storage device, an attached storage device, flash memory, battery backed-up SDRAM (synchronous DRAM), and/or a network accessible storage device. In embodiments, storage 714 may comprise technology to increase the storage performance enhanced protection for valuable digital media when multiple hard drives are included, for example.

[0029] Graphics subsystem 715 may perform processing of images such as still or video for display. Graphics subsystem 715 may be a graphics processing unit (GPU) or a visual processing unit (VPU), for example. An analog or digital interface may be used to communicatively couple graphics subsystem 715 and display 720. For example, the interface may be any of a High-Definition Multimedia Interface, DisplayPort, wireless HDMI, and/or wireless HD compliant techniques. Graphics subsystem 715 could be integrated into processor 710 or chipset 705. Graphics subsystem 715 could be a stand-alone card communicatively coupled to chipset 705.

[0030] The graphics and/or video processing techniques described herein may be implemented in various hardware architectures. For example, graphics and/or video functionality may be integrated within a chipset. Alternatively, a discrete graphics and/or video processor may be used. As still another embodiment, the graphics and/or video functions may be implemented by a general purpose processor, including a multi-core processor. In a further embodiment, the functions may be implemented in a consumer electronics device.

[0031] Radio 718 may include one or more radios capable of transmitting and receiving signals using various suitable wireless communications techniques.

Such techniques may involve communications across one or more wireless networks. Exemplary wireless networks include (but are not limited to) wireless local area networks (WLANs), wireless personal area networks (WPANs), wireless metropolitan area network (WMANs), cellular networks, and satellite networks. In communicating across such networks, radio 718 may operate in accordance with one or more applicable standards in any version.

[0032] In embodiments, display 720 may comprise any television type monitor or display. Display 720 may comprise, for example, a computer display screen, touch screen display, video monitor, television-like device, and/or a television. Display 720 may be digital and/or analog. In embodiments, display 720 may be a holographic display. Also, display 720 may be a transparent surface that may receive a visual projection. Such projections may convey various forms of information, images, and/or objects. For example, such projections may be a visual overlay for a mobile augmented reality (MAR) application. Under the control of one or more software applications 716, platform 702 may display user interface 722 on display 720.

[0033] In embodiments, content services device(s) 730 may be hosted by any national, international and/or independent service and thus accessible to platform 702 via the Internet, for example. Content services device(s) 730 may be coupled to platform 702 and/or to display 720. Platform 702 and/or content services device(s) 730 may be coupled to a network 760 to communicate (e.g., send and/or receive) media information to and from network 760. Content delivery device(s) 740 also may be coupled to platform 702 and/or to display 720.

[0034] In embodiments, content services device(s) 730 may comprise a cable television box, personal computer, network, telephone, Internet enabled devices or appliance capable of delivering digital information and/or content, and any other similar device capable of unidirectionally or bidirectionally communicating content between content providers and platform 702 and/display 720, via network 760 or directly. It will be appreciated that the content may be communicated unidirectionally and/or bidirectionally to and from any one of the components in system 700 and a content provider via network 760. Examples of content may

include any media information including, for example, video, music, medical and gaming information, and so forth.

[0035] Content services device(s) 730 receives content such as cable television programming including media information, digital information, and/or other content. Examples of content providers may include any cable or satellite television or radio or Internet content providers. The provided examples are not meant to limit the applicable embodiments.

[0036] In embodiments, platform 702 may receive control signals from navigation controller 750 having one or more navigation features. The navigation features of controller 750 may be used to interact with user interface 722, for example. In embodiments, navigation controller 750 may be a pointing device that may be a computer hardware component (specifically human interface device) that allows a user to input spatial (e.g., continuous and multi-dimensional) data into a computer. Many systems such as graphical user interfaces (GUI), and televisions and monitors allow the user to control and provide data to the computer or television using physical gestures.

[0037] Movements of the navigation features of controller 750 may be echoed on a display (e.g., display 720) by movements of a pointer, cursor, focus ring, or other visual indicators displayed on the display. For example, under the control of software applications 716, the navigation features located on navigation controller 750 may be mapped to virtual navigation features displayed on user interface 722, for example. In embodiments, controller 750 may not be a separate component but integrated into platform 702 and/or display 720. Embodiments, however, are not limited to the elements or in the context shown or described herein.

[0038] In embodiments, drivers (not shown) may comprise technology to enable users to instantly turn on and off platform 702 like a television with the touch of a button after initial boot-up, when enabled, for example. Program logic may allow platform 702 to stream content to media adaptors or other content services device(s) 730 or content delivery device(s) 740 when the platform is turned "off." In addition, chip set 705 may comprise hardware and/or software support for 5.1

surround sound audio and/or high definition 7.1 surround sound audio, for example. Drivers may include a graphics driver for integrated graphics platforms. In embodiments, the graphics driver may comprise a peripheral component interconnect (PCI) Express graphics card.

[0039] In various embodiments, any one or more of the components shown in system 700 may be integrated. For example, platform 702 and content services device(s) 730 may be integrated, or platform 702 and content delivery device(s) 740 may be integrated, or platform 702, content services device(s) 730, and content delivery device(s) 740 may be integrated, for example. In various embodiments, platform 702 and display 720 may be an integrated unit. Display 720 and content service device(s) 730 may be integrated, or display 720 and content delivery device(s) 740 may be integrated, for example. These examples are not meant to be scope limiting.

[0040] In various embodiments, system 700 may be implemented as a wireless system, a wired system, or a combination of both. When implemented as a wireless system, system 700 may include components and interfaces suitable for communicating over a wireless shared media, such as one or more antennas, transmitters, receivers, transceivers, amplifiers, filters, control logic, and so forth. An example of wireless shared media may include portions of a wireless spectrum, such as the RF spectrum and so forth. When implemented as a wired system, system 700 may include components and interfaces suitable for communicating over wired communications media, such as input/output (I/O) adapters, physical connectors to connect the I/O adapter with a corresponding wired communications medium, a network interface card (NIC), disc controller, video controller, audio controller, and so forth. Examples of wired communications media may include a wire, cable, metal leads, printed circuit board (PCB), backplane, switch fabric, semiconductor material, twisted-pair wire, co-axial cable, fiber optics, and so forth.

[0041] Platform 702 may establish one or more logical or physical channels to communicate information. The information may include media information and control information. Media information may refer to any data representing content meant for a user. Examples of content may include, for example, data from a voice

conversation, videoconference, streaming video, electronic mail ("email") message, voice mail message, alphanumeric symbols, graphics, image, video, text and so forth. Data from a voice conversation may be, for example, speech information, silence periods, background noise, comfort noise, tones and so forth. Control information may refer to any data representing commands, instructions or control words meant for an automated system. For example, control information may be used to route media information through a system, or instruct a node to process the media information in a predetermined manner. The embodiments, however, are not limited to the elements or in the context shown or described in Figure 3.

[0042] As described above, system 700 may be embodied in varying physical styles or form factors. Figure 4 illustrates embodiments of a small form factor device 800 in which system 700 may be embodied. In embodiments, for example, device 800 may be implemented as a mobile computing device having wireless capabilities. A mobile computing device may refer to any device having a processing system and a mobile power source or supply, such as one or more batteries, for example.

[0043] As described above, examples of a mobile computing device may include a personal computer (PC), laptop computer, ultra-laptop computer, tablet, touch pad, portable computer, handheld computer, palmtop computer, personal digital assistant (PDA), cellular telephone, combination cellular telephone/PDA, television, smart device (e.g., smart phone, smart tablet or smart television), mobile internet device (MID), messaging device, data communication device, and so forth.

[0044] Examples of a mobile computing device also may include computers that are arranged to be worn by a person, such as a wrist computer, finger computer, ring computer, eyeglass computer, belt-clip computer, arm-band computer, shoe computers, clothing computers, and other wearable computers. In embodiments, for example, a mobile computing device may be implemented as a smart phone capable of executing computer applications, as well as voice communications and/or data communications. Although some embodiments may be described with a mobile computing device implemented as a smart phone by way of example, it may be appreciated that other embodiments may be implemented using other

wireless mobile computing devices as well. The embodiments are not limited in this context.

[0045] The following clauses and/or examples pertain to further embodiments:

One example embodiment may be a method comprising accumulating a reduced representation of geometry, input data, and state for tile when a triangle list is created, before per tile rasterization, determining whether the representation for the current tile in the current frame is the same as the representation for the previous frame and the corresponding tile, and if the representations for the tile in previous and current frames are the same, then reusing the prior tile's pixels as the current tile's pixels when rendering the current tile. The method may include using a hash function to produce the reduced representation. The method may include using a bloom filter to produce the reduced representation. The method may include using two bloom filters, one for the current tile and one for a previous tile. The method may include using one pair of bloom filters for each of geometry, texture addresses, and shaders. The method may include accumulating in a sort-middle architecture. The method may include wherein reusing includes reusing the color values. The method may include wherein reusing includes reusing depth or stencil buffers. The method may include changing the current compact representation to the previous compact representation. The method may include determining if the same draw call is used for the same tile in successive frames to determine if the representations are the same.

[0046] Another example embodiment may be one or more non-transitory computer readable media storing instructions to implement a sequence comprising accumulating a reduced representation of geometry, input data, and state for tile when a triangle list is created, before per tile rasterization, determining whether the representation for the current tile in the current frame is the same as the representation for the previous frame and the corresponding tile, and if the representations for the tile in previous and current frames are the same, then reusing the prior tile's pixels as the current tile's pixels when rendering the current tile. The media may include said sequence including using a hash function to produce the reduced representation. The media may include said sequence including using a

bloom filter to produce the reduced representation. The media may include said sequence including using two bloom filters, one for the current tile and one for a previous tile. The media may include said sequence including using one pair of bloom filters for each of geometry, texture addresses, and shaders. The media may include said sequence including accumulating in a sort-middle architecture. The media may wherein reusing includes reusing the color values. The media may wherein reusing includes reusing depth or stencil buffers. The media may include said sequence including determining if the same draw call is used for the same tile in successive frames to determine if the representations are the same.

[0047] In another example embodiment may be an apparatus comprising a processor to accumulate a reduced representation of geometry, input data, and state for tile when a triangle list is created, before per tile rasterization, determine whether the representation for the current tile in the current frame is the same as the representation for the previous frame and the corresponding tile, and if the representations for the tile in previous and current frames are the same, then reuse the prior tile's pixels as the current tile's pixels when rendering the current tile, and a storage coupled to said processor. The apparatus may include said processor to use a hash function to produce the reduced representation. The apparatus may include a bloom filter to produce the reduced representation. The apparatus may include two bloom filters, one for the current tile and one for a previous tile. The apparatus may include one pair of bloom filters for each of geometry, texture addresses, and shaders. The apparatus may include said processor to accumulate in a sort-middle architecture. The apparatus may include said processor to refrain from sending data for display when the compact representations are the same. The apparatus may include said processor to determine if the same draw call is used for the same tile in successive frames to determine if the representations are the same. The apparatus may include a battery. The apparatus may include firmware and a module to update said firmware.

[0048] The graphics processing techniques described herein may be implemented in various hardware architectures. For example, graphics functionality may be integrated within a chipset. Alternatively, a discrete graphics processor may be

used. As still another embodiment, the graphics functions may be implemented by a general purpose processor, including a multicore processor.

[0049] References throughout this specification to “one embodiment” or “an embodiment” mean that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one implementation encompassed within the present disclosure. Thus, appearances of the phrase “one embodiment” or “in an embodiment” are not necessarily referring to the same embodiment. Furthermore, the particular features, structures, or characteristics may be instituted in other suitable forms other than the particular embodiment illustrated and all such forms may be encompassed within the claims of the present application.

[0050] While a limited number of embodiments have been described, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this disclosure.

What is claimed is:

1. An apparatus comprising:
a graphics processor to generate an identification code corresponding to one or more of uniforms, vertex data, texture coordinates or graphics state for a tile in a current frame, determine whether there is a draw call that fully covered the tile, store only the identification code for the draw call that fully covers the tile for a corresponding tile in a previous frame before the tile in the current frame is to be rendered, determine whether the code for the tile in the current frame is the same as a code for the corresponding tile in the previous frame, if the code for the corresponding tile in the previous frame and the code for the tile in the current frame are the same then reuse the image from the previous frame, as pixels for the tile in the current frame to avoid rendering the tile in the current frame; and
a frame buffer coupled to said processor.
2. The apparatus of claim 1 wherein the code for the corresponding tile in the current frame includes at least two of uniforms, triangle data, texture data or state for a tile.
3. The apparatus of claim 1 wherein the code for the corresponding tile in the current frame includes at least three of uniforms, triangle data, texture data or state for a tile.
4. The apparatus of claim 1 wherein the code for the corresponding tile in the current frame includes uniforms, triangle data, texture data and state for a tile.
5. The apparatus of claim 1 wherein said graphics processor uses sort middle processing.
6. The apparatus of claim 5, said processor to generate the code for the corresponding tile in the current frame before or at a natural break point in sort middle processing.

7. The apparatus of claim 1 wherein the code for the corresponding tile in the current frame is a hash.
8. The apparatus of claim 1 wherein said processor to generate the code for the corresponding tile in the current frame using a list of primitives.
9. A method comprising:
 - generating an identification code corresponding to one or more of uniforms, vertex data texturing coordinates or graphics state for a tile in a current frame;
 - determine whether there is a draw call that fully covered the tile,
 - storing only the identification code for the draw call that fully covers the tile for a corresponding tile in a previous frame before the tile in the current frame is to be rendered;
 - determining whether the code for the tile in the current frame is the same as a code for the corresponding tile in the previous frame; and
 - if the code for the corresponding tile in the previous frame and the code for the tile in the current frame are the same then reusing the image from the previous frame, as pixels for the tile in the current frame to avoid rendering the tile in the current frame.
10. The method of claim 9 wherein the code for the corresponding tile in the current frame includes at least two of uniforms, triangle data, texture data or state for a tile.
11. The method of claim 9 wherein the code for the corresponding tile in the current frame includes at least three of uniforms, triangle data, texture data or state for a tile.
12. The method of claim 9 wherein the code for the corresponding tile in the current frame includes uniforms, triangle data, texture data and state for a tile.
13. The method of claim 9 including using sort middle processing.

14. The method of claim 13 including generating the code for the corresponding tile in the current frame before or at a natural break point in sort middle processing.

15. The method of claim 9 wherein the code for the corresponding tile in the current frame is a hash.

16. The method of claim 9 wherein said processor to generate the code using a list of primitives.

17. An apparatus comprising:

means for generating an identification code corresponding to one or more of uniforms, vertex data, texture coordinates or graphics state for a tile in a current frame;

determine whether there is a draw call that fully covered the tile,

means for storing only the identification code for the draw call that fully covers the tile for a tile in a previous frame before the tile in the current frame is to be rendered;

means for determining whether the code for the tile in the current frame is the same as a code for the corresponding tile in the previous frame; and

means for reusing the image from the previous frame, as pixels for the tile in the current frame to avoid rendering the tile in the current frame if the code for the corresponding tile in the previous frame and the code for the tile in the current frame are the same.

18. The apparatus of claim 17 wherein the code for the corresponding tile in the current frame includes at least two of uniforms, triangle data, texture data or state for a tile.

19. The apparatus of claim 17 wherein the code for the corresponding tile in the current frame includes at least three of uniforms, triangle data, texture data or state for a tile.

20. The apparatus of claim 17 wherein the code for the corresponding tile in the current frame includes uniforms, triangle data, texture data and state for a tile.
21. The apparatus of claim 17 wherein said generating means uses sort middle processing.
22. The apparatus of claim 21 including means for generating the code for the corresponding tile in the current frame before or at a natural break point in sort middle processing.
23. The apparatus of claim 17 wherein the code for the corresponding tile in the current frame is a hash.
24. The apparatus of claim 17 wherein said generating means includes means for generating the code using a list of primitives.
25. A system comprising:
a graphics processor to generate an identification code corresponding to one or more of uniforms, vertex data, texture coordinates or graphics state for a tile in a current frame, determine whether there is a draw call that fully covered the tile, store only the identification code for a draw call that fully covers the tile for a corresponding tile in a previous frame before the tile in the current frame is to be rendered, determine whether the code for the tile in the current frame is the same as a code for the corresponding tile in the previous frame, if the code for the corresponding tile in the previous frame and the code for the tile in the current frame are the same then reuse the image from the previous frame, as pixels for the tile in the current frame to avoid rendering the tile in the current frame; and
a display coupled to said processor.
26. The system of claim 25 wherein the code for the corresponding tile in the current frame includes at least two of uniforms, triangle data, texture data or state for a tile.

27. The system of claim 25 wherein the code for the corresponding tile in the current frame includes at least three of uniforms, triangle data, texture data or state for a tile.
28. The system of claim 25 wherein the code for the corresponding tile in the current frame includes uniforms, triangle data, texture data and state for a tile.
29. The system of claim 25 wherein said graphics processor to use sort middle processing.
30. The system of claim 25, said processor to generate the code for the corresponding tile in the current frame before or at a natural break point in sort middle processing.
31. The system of claim 25 wherein the code for the corresponding tile in the current frame is a hash.
32. The system of claim 25 wherein said processor to generate the code for the corresponding tile in the current frame using a list of primitives.

33. A system comprising:
means for generating an identification code corresponding to one or more of uniforms, vertex data, texture coordinates or graphics state for a tile in a current frame;
determine whether there is a draw call that fully covered the tile,
means for storing only the identification code for the draw call that fully covers the tile for a tile in a previous frame before the tile in the current frame is to be rendered;
means for determining whether the code for the tile in the current frame is the same as a code for the corresponding tile in the previous frame;
means for reusing the image from the previous frame, as pixels for the tile in the current frame to avoid rendering the tile in the current frame, if the code for the corresponding tile in the previous frame and the code for the tile in the current frame are the same; and
means for displaying images.
34. The system of claim 33 wherein the code for the corresponding tile in the current frame includes at least two of uniforms, triangle data, texture data or state for a tile.
35. The system of claim 33 wherein the code for the corresponding tile in the current frame includes at least three of uniforms, triangle data, texture data or state for a tile.
36. The system of claim 33 wherein the code for the corresponding tile in the current frame includes uniforms, triangle data, texture data and state for a tile.
37. The system of claim 33 wherein said generating means uses sort middle processing.

38. The system of claim 37 including means for generating the code for the corresponding tile in the current frame before or at a natural break point in sort middle processing.

39. The system of claim 33 wherein the code for the corresponding tile in the current frame is a hash.

40. The apparatus of claim 33 wherein said generating means includes means for generating the code for the corresponding tile in the current frame using a list of primitives.