Software Test Specification

OpenGL ES 2.0

Copyright © 2010, Imagination Technologies Ltd. All Rights Reserved.

This document is confidential. Neither the whole nor any part of the information contained in, nor the product described in, this document may be adapted or reproduced in any material form except with the written permission of Imagination Technologies Ltd. This document can only be distributed to Imagination Technologies employees, and employees of companies that have signed a Non-Disclosure Agreement with Imagination Technologies Ltd.

Filename	:	Software Test Specification.OpenGL ES 2.0.doc
Version	:	1.0.279 External Issue
Issue Date	:	14 Oct 2010
Author	:	PCG

Contents

1. Introduction

This document specifies the tests used during SQA to validate the OpenGL ES 2.0 implementation on Imagination Technologies Hardware IP Cores.

1.1. Related Documentation

Title	Description
OpenGL ES Specification 2.0	The OpenGL ES 2.0 specification.
Software Test Specification.OpenGL ES Shading Language.doc	The OpenGL ES Shading Language test specification. (http://www.khronos.org/opengles/spec.html).

1.2. Assumptions

This document is written with the following assumptions:

The reader is familiar with OpenGL ES in general, OpenGL ES 2.0 in particular and the OpenGL ES Shading Language.

1.3. Document Scope

This specification describes the tests to be performed to verify the implementation of OpenGL ES 2.0.

2. Test Scope

This section details the features that OpenGL ES 2.0 has been split into for testing and the risk associated with each feature not being tested completely.

2.1.1. Features to be tested

Feature	Description	Risk
Vertex Arrays	Specification and use of vertices	High
Buffer Objects	Storage of frequently used data in server memory	High
Coordinate Transformations	Transformations that need to be performed outside of a vertex shader	High
Colours and Colouring	Specification of colours for objects	High
Shaders	Compiling, linking and using Vertex and Fragment shaders	High
Rasterization	Rasterization of lines, polygons and pixels	High
Texturing	Loading and use of textures	High
Per-Fragment Operations	Operations that are performed on individual fragments	High
Framebuffer	Support and manipulation of drawing buffers	High
Synchronisation	Forcing completion of OpenGL drawing	Medium
Hints	Control aspects of OpenGL behaviour	Medium
State and State Requests	Queries that return information about the current OpenGL state	High
Errors	Error behaviour and the current error state	Medium
Extensions	Support for and use of extensions	Low
Shaders compilation performance	Compilation of shaders	Low

Please refer to Appendix A for full details of functions, variables and extensions to be tested

3. Test Catalogue

3.1. Functional Tests

ID	Application	Description	Verif.	Source
GES_FN01	GLSLEStriangle	Minimal GLSL smoke test	M	IMG
OES2_FN02	OGLES2glslapitest	Tests the OpenGL ES ability to use vertex and fragment shaders.	A	IMG
OES2_FN03	OGLES2testkit	Tests for OpenGL ES 2.0.	S/A	IMG
OES2_FN04	Gles2test1	Unit test for OpenGLES2.0	M	IMG
OES2_FN05	gles2_texture_stream	Unit test for OGLES2.0 texture streaming.	M	IMG
CMP_FN10	Offline_compiler_Tester	Script that tests the offline compiler by attempting to compile valid and invalid shaders	M	IMG
OES2_FN06	Ogles2Extension	Tests Olges2 Extensions	M	IMG
EGL_FN03	MixedAPITest2	Mixed APIs (OpenVG & OpenGLES2)	M	IMG
OES2_FN08	Ogles2checksymbols	Script testing the list of exported symbols in the OpenGL ES 2 library to check if unnecessary symbols are exported.	A	IMG

3.2. Non-Functional Tests

3.2.1. Stress Tests

ID	Application	Description	Verif.	Source
OES2_SR01	OGLES2Shaders + systemstress	Run OGLES2Shaders while under heavy CPU load from systemstress.	A	IMG
OES2_SR02	Systemstress + OGLES2Shaders	Run OGLES2Shaders while a lot of memory is allocated by systemstress	A	IMG
OES2_SR03	Systemstress + OGLES2Shaders	Run OGLES2Shaders while memory is constantly allocated and deallocated by systemstress	A	IMG
OES2_SR04	Multi Context	Two applications (OGLES2Shaders and OGLES2Reflections) are started at the same time and run for 30 minutes.	A	IMG
OES2_SR05	OGLES2multithread*	Run multi threaded OpenGL ES 2.0 test.	A	IMG

ID	Application	Description	Verif.	Source
OES2_SR06	Multi Context - Parameter Buffer Resize - test 1	Two applications (OGLES2Skinning And OGLES2SkyBox2) are started at the same time and runs for 30 minutes. OGLES2Skinning is running with 20 pages and OGLES2SkyBox2 with the default number of pages.	A	IMG
OES2_SR07	Multi Context - Parameter Buffer Resize - test 2	One application (OGLES2Skinning with 20 pages) is running in the background and a second application (OGLES2SkyBox2 with default number of pages) is started, stopped and restarted 10 times.	A	IMG
OES2_SR08	Multi Context - Parameter Buffer Resize - test 3	One application (OGLES2Skinning with 20 pages) is running in the background and multiple others (OGLES2SkyBox2 default number of pages, OGLES2Shaders 30 pages, and OGLES2Reflections 25 pages) are started after each other for 1 minute each. This is done 10 times.	A	IMG

Note: tests marked with a star are not implemented yet.

3.2.2. Stability Tests

ID	Application	Description	Verif.	Source
OES2_SB01	OGLES2 SDK Demos	Runs a loop of all OpenGL ES 2.0 demo applications constantly for at least 8 hours.	SA/A	IMG
OES2_SB02	OGLES2 SDK Demos	Runs a single OpenGL ES 2.0 demo application in infinite run mode for at least 8 hours.	SA/A	IMG
OES2_SB03	CCrces*	Allocate and de-allocate OpenGL ES resources for at least 30 minutes.	SA/A	IMG

Note: tests marked with a star are not implemented yet.

3.2.3. Conformance Tests

ID	Application	Description	Verif.	Source
OES2_CC01	GTF	Khronos Conformance Tests*	A	KHR

Note: tests marked with a star are not implemented/finalized yet.

3.2.4. Performance Tests

ID	Application	Description	Verif.	Source
OES2_FN7	Ogles2uniflexbench	Test the shaders compilation times	M	IMG
OES2_PF10	OGLES2GridMark3	Bench 0: Game Scene	A	IMG
OES2_PF11	OGLES2GridMark3	Bench 1: Polygon Throughput	A	IMG
OES2_PF12	OGLES2GridMark3	Bench 2: Visible Pixel Fillrate	A	IMG

ID	Application	Description	Verif.	Source
OES2_PF13	OGLES2GridMark3	Bench 3: Depth Test Fillrate	A	IMG
OES2_PF14	OGLES2GridMark3	Deprecated	A	IMG
OES2_PF15	OGLES2GridMark3	Bench 5: Effective Pixel Fillrate	A	IMG
OES2_PF16	OGLES2GridMark3	Bench 6: Lighting	A	IMG
OES2_PF17	OGLES2GridMark3	Bench 7: Peak Hardware Polygon Throughput	A	IMG
OES2_PF18	3DMarkMobileES2.0	FutureMark Benchmark for 3D graphics performance	SA	Ext
OES2_PF19	SimulationMark	FutureMark Benchmark for 3D graphics performance.	SA	Ext
OES2_PF20	GLBenchmark 2.0	Kishonti Benchmark for 3D graphics performance.	SA	Ext
OES2_PF21	3DMarkMobileES2.0 V2	FutureMark Benchmark for 3D graphics performance	SA	Ext

Note: tests marked with a star are not implemented yet.

3.2.5. Content Tests

ID	Application	Description	Verif.	Source
OES2_CT01	OGLES2Shaders	SDK demo showing use of vertex and fragment shaders	SA/A	IMG
OES2_CT02	OGLES2AlphaBlend	SDK demo showing four different types of blending modes	SA/A	IMG
OES2_CT03	OGLES2AlphaTest	SDK demo showing an alpha test	SA/A	IMG
OES2_CT04	OGLES2AnisotropicLighting	SDK demo showing anisotropic lighting	SA/A	IMG
OES2_CT05	OGLES2BasicTnL	SDK demo showing basic transform and lighting	SA/A	IMG
OES2_CT06	OGLES2Bumpmap	SDK demo showing bump mapping	SA/A	IMG
OES2_CT07	OGLES2CellShading	SDK demo showing cell shading	SA/A	IMG
OES2_CT08	OGLES2ComplexLighting	SDK demo showing complex lighting	SA/A	IMG
OES2_CT09	OGLES2FastTnL	SDK demo showing fast transform and lighting with a vertex shader	SA/A	IMG
OES2_CT10	OGLES2Fog	SDK demo showing a fog effect using a vertex shader	SA/A	IMG
OES2_CT11	OGLES2FresnelReflections	SDK demo showing Fresnel reflections	SA/A	IMG
OES2_CT12	OGLES2IntroducingPFX	SDK demo showing an animation with basic lighting	SA/A	IMG

ID	Application	Description	Verif.	Source
OES2_CT13	OGLES2IntroducingPOD	SDK demo showing an animation with basic lighting	SA/A	IMG
OES2_CT14	OGLES2IntroducingPVRShell	SDK demo showing a simple triangle	SA/A	IMG
OES2_CT15	OGLES2IntroducingPVRTools	SDK demo showing loading of textures and displaying of text	SA/A	IMG
OES2_CT16	OGLES2Iridescence	SDK demo showing an iridescence effect	SA/A	IMG
OES2_CT17	OGLES2LightMap	SDK demo showing light maps using three textures	SA/A	IMG
OES2_CT18	OGLES2PerturbedUvs	SDK demo showing a perturbed Uvs using a texture as a normal map	SA/A	IMG
OES2_CT19	OGLES2Reflections	SDK demo showing reflections, 2D mapping and cube mapping	SA/A	IMG
OES2_CT20	OGLES2Refractions	SDK demo showing a glass like material and Phong's specular reflections	SA/A	IMG
OES2_CT21	OGLES2RenderToTexture	SDK demo showing a frame buffer object being used to render to a texture	SA/A	IMG
OES2_CT22	OGLES2StencilBuffer	SDK demo showing the use of a stencil buffer	SA/A	IMG
OES2_CT23	OGLES2Texturing	SDK demo showing the loading of a texture	SA/A	IMG
OES2_CT24	OGLES2SkyBox2	SDK demo showing vertex and fragment shaders	SA/A	IMG
OES2_CT25	OGLES2Skinning	SDK demo showing how to implement skinning using vertex shaders	SA/A	IMG
OES2_CT26	OGLES2LevelofDetail	SDK demo showing the use of dynamic branching to run a simplified shader	SA/A	IMG
OES2_CT27	OGLES2ShadowVolume	SDK demo showing how to implement Shadow Volumes using 'stencil buffers'	SA/A	IMG
OES2_CT28	OGLES2HelloTriangle	SDK demo showing a yellow triangle against a blue background	SA/A	IMG
OES2_CT29	OGLES2Initialization	SDK demo that initializes the screen with different colours	SA/A	IMG
OES2_CT30	OGLES2Bloom	SDK Training course demonstrates a simple implementation of a Bloom effect	SA/A	IMG

ID	Application	Description	Verif.	Source
OES2_CT31	OGLES2ChameleonMan	SDK demo that shows a matrix skinned character in combination with DOT3 per pixel lighting	SA/A	IMG
OES2_CT32	OGLES2CoverFlow	SDK Demo that demonstrates how to implement a simple version of 'cover flow'	SA/A	IMG
OES2_CT33	OGLES2DisplacementMapping	SDK Demo that shows how to displace geometry in the vertex shader using a texture	SA/A	IMG
OES2_CT34	OGLES2FilmTV	SDK Demo that demonstrates render to texture	SA/A	IMG
OES2_CT35	OGLES2PhantomMask	SDK Demo that shows a mask lit using spherical harmonics and regular diffuse vertex lighting	SA/A	IMG
OES2_CT36	OGLES2ShadowMapping	SDK Demo that demonstrates shadow mapping using GL_OES_depth_texture extension	SA/A	IMG
OES2_CT37	OGLES2Water	SDK Demo that shows how to render a water effect on a given plane	SA/A	IMG

3.2.6. BizDev Content Tests

ID	Application	Description	Verif.	Source
OES2_BZ01	Flowers		SA/A	IMG
OES2_BZ02	OGLES2BlueMarble		SA/A	IMG
OES2_BZ03	OGLES2Coverflow		SA/A	IMG
OES2_BZ04	OGLES2Dash		SA/A	IMG
OES2_BZ05	OGLES2Dash_Poland		SA/A	IMG
OES2_BZ06	OGLES2Dash_Ukraine		SA/A	IMG
OES2_BZ07	OGLES2HDRSkybox		SA/A	IMG
OES2_BZ08	OGLES2MountainView		SA/A	IMG
OES2_BZ09	OGLES2RealCharacters		SA/A	IMG
OES2_BZ10	OGLES2Shaders		SA/A	IMG
OES2_BZ11	WindowEPG		SA/A	IMG

4. Test Cases

4.1. OES2_FN01 - OGLES2triangles

Test Case ID	Function/Variable Under Test	Description	Expected Result
101	ALL	Draw two triangles, one textured and one coloured. Pass them through a vertex and a fragment shader.	Triangles should be textured, coloured and transformed correctly

4.2. OES2_FN02 - OGLES2apitest

Test Groups

Test Group ID	Test Group Name
100	Creating Shader Objects
200	Compiling Shader Objects
300	Linking and Using Shaders
400	Cleaning Up
500	Query Functions
600	Vertex Attributes
700	Uniform Variables
900	Multiple Render Targets
1000	OES Extensions

Test Cases

Test Case ID	Function/Variable Under Test	Description	Expected Result
101	glCreateShader	Query the function in an OpenGL application Pass the shaderTypes *GL_VERTEX_SHADER* and *GL_FRAGMENT_SHADER* to the function	An empty shader object should be created and a non-zero value by which it can be referenced should be returned
102	glShaderSource	Query the function in an OpenGL application Pass a shader object and a number of shader source strings to the function	The shader source should be stored in the shader object

Test Case ID	Function/Variable Under Test	Description	Expected Result
201	glCompileShader	Query the function in an OpenGL application Pass a number of shader objects to the function	The shader source stored in the shader object should be compiled into a shader The compilation status can be verified by calling **glGetShader** with the argument *shader* and *GL_COMPILE_STATUS*
301	glCreateProgram	Query the function in an OpenGL application Call the function with no arguments	An empty program object should be created and a non-zero value by which it can be referenced should be returned
302	glAttachShader	Query the function in an OpenGL application Pass a number of shader objects and programs to the function	The shader object specified by *shader* should be attached to the program object specified by *program*
303	glLinkProgram	Query the function in an OpenGL application Pass a number of program objects to the function	The program object specified by *program* should be used to create an executable to run on the programmable hardware The linking of a program object should fail if: the number of attribute variables supported is exceeded the number of uniform variables supported is exceeded the **main** function is missing a varying variable used in the fragment shader is not declared in the vertex shader a reference to a function or variable is unresolved a shared global is declared with different types or values an attached shader object hasn't been successfully compiled some rows of the matrix when binding a generic attribute matrix fell outside the allowed maximum not enough contiguous vertex attribute slots are available to bind attribute matrices The link status can be verified by calling **glGetProgram** with the arguments *program* and *GL_LINK_STATUS*
304	glUseProgram	Query the function in an OpenGL application Pass a number of linked program objects to the function	The program object specified by *program* should be installed as part of the current rendering state

Test Case ID	Function/Variable Under Test	Description	Expected Result
401	glDeleteShader	Query the function in an OpenGL application Pass a number of shader objects to the function	The shader object specified by *shader* should be invalidated and the memory should be freed If a shader is currently attached to a program object, it should not be deleted until it has been detached
402	glDeleteProgram	Query the function in an OpenGL application Pass a number of program objects to the function	The program object specified by *program* should be invalidated and the memory should be freed
403	glDetachShader	Query the function in an OpenGL application Pass a number of program objects and attached shaders to the function	The shader object specified by *shader* should be detached from the program object specified by *program*
501	glGetShaderiv	Query the function in an OpenGL application Pass a number of shader objects to the function with parameters *GL_SHADER_TYPE*, *GL_DELETE_STATUS*, *GL_COMPILE_STATUS*, *GL_INFO_LOG_LENGTH* and *GL_SHADER_SOURCE_LENGTH*	The value of the parameter for the specified shader object should be returned
502	glGetProgramiv	Query the function in an OpenGL application Pass a number of program objects to the function with parameters *GL_DELETE_STATUS, GL_LINK_STATUS, GL_INFO_LOG_LENGTH, GL_ATTACHED_SHADERS, GL_ACTIVE_ATTRIBUTES, GL_ACTIVE_ATTRIBUTE_MAX_LENGTH, GL_ACTIVE_UNIFORMS* and *GL_ACTIVE_UNIFORM_MAX_LENGTH*	The value of the parameter for the specified program object should be returned
503	glGetShaderSource	Query the function in an OpenGL application Pass a number of shader objects to the function	A concatenation of the source code strings from the shader object specified by *shader* should be returned
504	glGetShaderInfoLog	Query the function in an OpenGL application Pass a number of shader objects to the function	The information log for the specified shader object should be returned

Test Case ID	Function/Variable Under Test	Description	Expected Result
505	glGetProgramInfoLog	Query the function in an OpenGL application Pass a number of program objects to the function	The information log for the specified program object should be returned
506	glGetAttachedShaders	Query the function in an OpenGL application Pass a number of program objects with a varying number of attached shader objects to the function	The handles of the shader objects attached to the program object *program* should be returned
507	glIsShader	Query the function in an OpenGL application Pass a number of shader and non-shader objects to the function	If *shader* is the name of a shader object, *GL_TRUE* should be returned, otherwise *GL_FALSE* should be returned
508	glIsProgram	Query the function in an OpenGL application Pass a number of program and non-program objects to the function	If *program* is the name of a program object, *GL_TRUE* should be returned, otherwise *GL_FALSE* should be returned
601	glVertexAttrib	Query the function in an OpenGL application Pass a number of values, *v*, with the generic vertex attribute *index* to the function, using the suffixes 1,2,3,4 and b,s,I,f,d,ub,us,ui for the different data types and number of components	The generic vertex attribute specified by *index* should be filled with the values *v* of the appropriate data type If the fourth component is not provided, it should be set to 1 If the second and third are not provided, they should be set to 0
602	glVertexAttrib4Nub	Query the function in an OpenGL application Pass a number of normalised values, *v*, with the generic vertex attribute *index*to the function, using the suffixes 1,2,3,4 and b,s,I,f,d,ub,us,ui for the different data types and number of components	The generic vertex attribute specified by *index* should be filled with the values *v* of the appropriate data type, mapped to [-1, 1] for signed data, and [0, 1] for unsigned data. If the fourth component is not provided, it should be set to 1 If the second and third are not provided, they should be set to 0
603	glVertexAttribPointer	Query the function in an OpenGL application Pass a number of generic vertex attribute variable arrays to the function	The generic vertex attribute specified by *index* should be filled with the provided data from the vertex array
604	glEnableVertexAttribArray	Query the function in an OpenGL application Pass a number of generic vertex attribute arrays specified by *index* to the function	The values in the generic vertex attribute array should be accessed and used for rendering

Test Case ID	Function/Variable Under Test	Description	Expected Result
605	glDisableVertexAttribArray	Query the function in an OpenGL application Pass a number of enabled generic vertex attribute variable arrays to the function	The values in the enabled generic vertex attribute array should stop being used for rendering
606	glBindAttribLocation	Query the function in an OpenGL application Pass a number of generic vertex attributes or arrays, specified by *index*, with a program object and variable *name* to the function	The generic vertex attribute specified by *index* should be bound to the program specified by *program* and be accessible within a shader using *name*
607	glGetAttribLocation	Query the function in an OpenGL application Pass an attribute variable, *name*, with a program object *program* to the function	The index of the generic vertex attribute bound to attribute variable *name* in the linked program object *program* should be returned
608	glGetActiveAttrib	Query the function in an OpenGL application	The information about an active attribute variable, specified by *index*, in the program object specified by *program* should be returned
609	glGetVertexAttrib	Query the function in an OpenGL application	The value of a generic vertex attribute parameter specified by *index* should be returned
610	glGetVertexAttribPointerv	Query the function in an OpenGL application	A pointer to information about generic vertex attribute parameters specified by *index* should be returned
701	glGetUniformLocation	Query the function in an OpenGL application Pass a uniform variable, *name*, with a program object *program* to the function	The index of the generic uniform attribute bound to uniform variable *name* in the linked program object *program* should be returned
702	glUniform	Query the function in an OpenGL application Pass a number of values, *v*, and the user-defined uniform variable or uniform variable array *location* to the function using the suffixes 1,2,3,4 and f,I for the different data types and number of components	The user-defined uniform variable specified by *location* should be filled with the values *v* of the appropriate data type

Test Case ID	Function/Variable Under Test	Description	Expected Result
703	glUniformMatrix	Query the function in an OpenGL application Pass a number of values, *v*, and the user-defined uniform matrix variable or uniform matrix variable array *location* to the function using the suffixes 1,2,3,4 and f,I for the different data types and number of components	The user-defined uniform matrix variable specified by *location* should be filled with the values *v* of the appropriate data type
704	glGetUniform	Query the function in an OpenGL application Pass a number of program objects *program* and user-defined uniform variables *location* to the function	The values of the uniform variable specified by *location* should be returned in **params*
705	glGetActiveUniform	Query the function in an OpenGL application	The information about an active uniform variable, specified by *index*, in the program object specified by *program* should be returned
901	glValidateProgram	Query the function in an OpenGL application Pass a number of program objects to the function	This should check whether the executables contained in *program* can execute given the current OpenGL state, and can be checked by calling **glGetProgram** with *GL_VALIDATE_STATUS*
1001	glShaderBinaryOES	Query the function in an OpenGL application Pass a number of compiled shader binaries or binaries containing optimised vertex/fragment shader pairs to the function	The shader binary specified by **binary** should be loaded and set in place as part of the OpenGL pipeline
1002	glReleaseShaderCompilerOES	Query the function in an OpenGL application after using **glCompilerShader** to compile a number of shaders	The resources used by the shader compiler should be released
1003	glGetShaderPrecisionFormatOES	Query the function in an OpenGL application	The range and precision for formats supported should be returned

4.3. OES2_FN03 - OGLES2testkit

Test Groups

Test Group ID	Test Group Name
100	Vertex Arrays
200	Buffer Objects
300	Coordinate Transformations

Test Group ID	Test Group Name
400	Colours and Colouring
500	Rasterization
600	Texturing
700	Per-Fragment Operations
800	Framebuffer
900	Synchronisation
1000	Hints
1100	State and State Requests
1200	Errors
1300	Extensions

Test Cases

Test Case ID	Function/Variable Under Test	Description	Expected Result
101	glVertexAttrib{1234}f[v]	Specify a number of vertices of all types. Use **glDrawArrays** to display the vertices	The specified vertices should correctly be written to the framebuffer and displayed
102	glVertexAttribPointer	After specifying a number of vertices, and for all supported sizes and data types, specify the array to be used to update the value of a variable	The data values for *index* should be updated with the appropriate values from the array pointed to by **pointer*
103	glDrawArrays	After specifying a number of vertices, call **glDrawArrays** with all supported *mode*s	A sequence of geometric primitives of king *mode* should be created using *count* array elements starting at *first*
104	glDrawElements	After specifying a number of elements, whose indices are stored in the array *indices*, call **glDrawElements** for all supported *type*s and *mode*s	A sequence of geometric primitives of kind *mode* should be created using *count* elements whose indices are stored in the array *indices*
105	glEnableVertexAttribArray	After specifying up to **GL_MAX_VERTEX_ATTRIBS** vertex arrays, enable the arrays	Vertex arrays specified by *index* should be enabled
106	glDisableVertexAttribArray	After specifying and enabling up to **GL_MAX_VERTEX_ATTRIBS** vertex arrays, disable the arrays	Vertex arrays specified by *index* should be disabled
201	glBindBuffer	Specify a buffer object, where *buffer* is an unused unsigned integer	A new buffer object should be created and assigned the name *buffer*
202	glBindBuffer	Specify a buffer object, where *buffer* is a previously created buffer object	The buffer object, *buffer*, should become the active buffer object
203	glBindBuffer	Specify a buffer object, where *buffer* is zero	OpenGL should stop using buffer objects

Test Case ID	Function/Variable Under Test	Description	Expected Result
204	glDeleteBuffers	Specify a number of buffers, *n*, named by elements in the array **buffers*	*n* buffer objects should be deleted, with the freed buffer objects now able to be reused If a buffer object was currently bound, the bindings to that object should be reset to the default buffer object Any attempts to delete a nonexistent buffer should be ignored
205	glGenBuffers	Request OpenGL to allocate *n* buffer object identifiers	*n* unused names for buffer objects should be returned in the array **buffers*
206	glBufferData	For a range of storage units *size* (in bytes), and for the possible *target*s, pass a range of pointers **data* to the function specifying a point in client memory	If a valid pointer is passed, *size* units of storage should be copied from the client to the server, with OpenGL possibly optimising the data for better performance based on the hint *usage* If *data* is NULL, *size* units of storage should be reserved for use but left uninitialised GL_OUT_OF_MEMORY should be returned if the requested *size* exceeds what the server is able to allocate
207	glBufferSubData	Update *size* amount of data currently stored in a buffer object using *offset* and the data specified by *data*	*Size* bytes of data should be updated in the currently bound buffer object starting *offset*
208	*glMapBuffer	If **OES_mapbuffer** is supported, attempt to access buffer data with operations indicated by *access* for the currently bound buffer object associated with *target*	A pointer to the data storage for the currently bound buffer object should be returned, and it should be possible to read, write or read and write the data depending on the kind of *access* granted
209	glUnmapBuffer	If **OES_mapbuffer** is supported, calls to this function will indicate to OpenGL that updates to the currently bound buffer object are complete	The buffer should be released
301	glDepthRangef	Specify a range of values, *near* and *far* to define an encoding for z-coordinates used during the viewport transformation	These values should be accessible from within a shader, and they should represent the adjustment to the minimum and maximum values that can be stored in the depth buffer
302	glViewPort	Specify a range of values, *x*, *y*, *width* and *height*	The final image should be mapped to the pixel rectangle represented by a viewport of *width* and *height* with the lower left corner at (*x*,*y*)

Test Case ID	Function/Variable Under Test	Description	Expected Result
401	glActiveTexture	Specify a number of texture units, *texUnit*, up to k-1 where k is the maximum number of texture units and then perform a number of texture routines	Any texture routines should modify the texture unit specified by *texUnit*
402	glFrontFace	Specify the mode to control how front-facing polygons are determined, either GL_CCW or GL_CW, and specify different colours for front and back faces	Front faced polygons and back faced polygons should appear the correct colour
501	glLineWidth	Specify a range of values, *width*, greater than 0, to control the width in pixels of rendered lines	Any rendered lines should be drawn with width *width*
502	glCullFace	Using **glEnable** and **glDisable**, and the modes GL_FRONT, GL_BACK, GL_FRONT_AND_BACK, specify which kind of polygons should be discarded before they're converted to screen coordinates	When enabled, the front, back, or all polygons should be discarded based on the *mode*
503	glPolygonOffset	Using **glEnable** and **glDisable**, and the capability POLYGON_OFFSET_FILL, specify a range of *factor*s and *unit*s to control the depth offset of polygons	Drawn polygons should be drawn with the depth value of each fragment added to the offset specified by m . *factor* + r . *units*
601	glPixelStorei	Specify a range of parameters *pname* with their values *param*	The pixel storage modes which affect the operation of the texture functions should be set to the values specified by *pname*
602	glTexImage2D	Define a range of different sized 2D textures for all targets with all supported formats	The texture image data specified by **data* should be defined and stored correctly
603	glTexSubImage2D	For all targets and supported formats, specify a new set of texel data and a part of the current texture image	The texture image data specified by **texels* should replace all or part of the existing two-dimensional texture
604	glCopyTexImage2D	For all targets and supported formats, specify a screen rectangle with *x*, *y*, *width* and *height*	A two-dimensional texture should be created using the framebuffer specified by the screen rectangle to define the texels, with the pixels read from the current GL_READ_BUFFER
605	glCopyTexSubImage2D	For all targets and supported formats, specify a screen rectangle with *x*, *y*, *width* and *height* and a part of the current texture image	A two-dimensional texture should be created using the framebuffer specified by the screen rectangle to replace all or part of the existing two-dimensional texture

Test Case ID	Function/Variable Under Test	Description	Expected Result
606	glCompressedTexImage2D	Define a range of different sized compressed two-dimensional textures for all targets with all supported formats	The compressed texture image data specified by **data* should be defined and stored correctly
607	glCompressedTexSubImage2D	For all targets and supported formats, specify a new set of compressed texel data and a part of the current texture image	The compressed texture image data specified by **data* should replace all or part of the existing compressed texture image
608	glTexParameter{if}[v]	Specify a range of parameters *pname* with their type *param* for texture specified by *target*	Textures of type *target* should be treated with the specified parameters as they're applied to fragments or stored in a texture object
610	glBindTexture	Specify a texture object, where *textureName* is an unused unsigned integer	A new texture object should be created and assigned the name *textureName*, with dimensionality of *target*
611	glBindTexture	Specify a texture object, where *textureName* is a previously created buffer object	The texture object, *textureName*, should become the active buffer object
612	glBindTexture	Specify a texture object, where *textureName* is zero	OpenGL should stop using texture objects and return to the unnamed default texture
613	glDeleteTextures	Specify a number of textures, *n*, named by elements in the array **textureNames*	*n* texture objects should be deleted, with the freed texture objects now able to be reused If a texture object was currently bound, the bindings to that object should be reset to the default texture object Any attempts to delete a nonexistent texture should be ignored
614	glGenTextures	Request OpenGL to allocate *n* texture object identifiers	*n* unused names for texture objects should be returned in the array **textureNames*
701	glScissor	With SCISSOR_TEST enabled, pass a range of values *x*, *y*, *width* and *height* to the function	A scissor rectangle of size *width* * *height* with lower left corner (*x*,*y*) should be set, with any pixels inside the rectangle passing the test
711	glSampleCoverage	Using **glEnable** and **glDisable**, specify a range of values *value* and set the boolean value *invert*	The parameters specified by *value* and *invert* should be used to interpret alpha values while computing the multisampling coverage depending on what kind of coverage has been enabled

Test Case ID	Function/Variable Under Test	Description	Expected Result
721	glStencilFunc	With STENCIL_TEST enabled, set a range of comparison functions *func*, reference values *ref* and *mask*	If there is a stencil buffer, the reference value should be compared with the value in the stencil buffer using the comparison function *func*, with t he comparison applying to the bits which have the corresponding bits of the mask equal to 1
722	glStencilFunSeparate	Perform the same tests as for **glStencilFunc**, however also specify a parameter for front or back-facing polygons	As for **glStencilFunc**, however different comparisons should be performed depending on whether a polygon is front or back-facing
723	glStencilOp	Specify a range of values *fail, zfail* and *zpass*	If a fragment fails the stencil test, the *fail* function should be applied to the fragment If a fragment passes the stencil test, then the *zfail* function should be applied to the fragment if it fails the depth test, or *zpass* if it passes the depth test
724	glStencilOpSeparate	Specify a range of values *face, fail, zfail* and *zpass*	For front or back facing fragments: If a fragment fails the stencil test, the *fail* function should be applied to the fragment If a fragment passes the stencil test, then the *zfail* function should be applied to the fragment if it fails the depth test, or *zpass* if it passes the depth test
731	glDepthFunc	With DEPTH_TEST enabled, set a number of comparison functions for the depth test	Incoming fragments should have their z-value compared using the specified relation to the value already stored in the depth value
741	glBlendFunc	With BLEND enabled, pass a range of supported blending factors *srcfactor* and *destfactor* to the function	Colour values in the fragment being processed should be combined with those stored in the framebuffer using the specified blending factors
742	glBlendFuncSeparate	Pass a range of supported blending factors *srcRGB*, *destRGB*, *srcAlpha* and *destAlpha* to the function	Colour and alpha values in the fragment being process should be combined with those stored in the framebuffer using the specified blending factors
743	glBlendEquation	Pass a range of supported modes, *mode*, to the function	The framebuffer and source colours should be blended together based on the function specified by *mode*
744	glBlendEquationSeparate	Pass a range of supported modes, *modeRGB*, *modeAlpha* to the function	The framebuffer and source colours and alpha values should be blended together based on the specified modes

Test Case ID	Function/Variable Under Test	Description	Expected Result
745	glBlendColor	Specify a range of values *red, green, blue* and *alpha*	The *rgba* values specified should be used as the constant colour in blending operations
801	glColorMask	Specify a range of values for the boolean variables *red, green* and *blue*	Before OpenGL writes data into the colour buffer, the values should be compared to the current colour mask specified by the values *red, green* and *blue* and masked accordingly
802	glClear	Specify a range of values *mask* that is the bitwise logical OR combination of GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT, GL_STENCIL_BUFFER_BUT	The buffers specified by *mask* should be cleared
803	glClearColor	Specify a range of values *red*, *green*, *blue* and *alpha*	The current clearing colour for use in clearing colour buffers in RGBA should be set to the rgba values specified
804	glDepthMask	Specify values for the boolean variable *flag*	Before OpenGL writes data into the depth buffer, the values should be compared to *flag* and masked accordingly
805	glClearDepthf	Specify a range of values, *depth*	When **glClear** is called, every pixel in the depth buffer should be set to the value *depth*
806	glStencilMask	Specify a range of masks *mask*	Before OpenGL writes data into the stencil buffer, the values should be compared to *mask* and masked accordingly
807	glStencilMaskSeparate	Specify a range of masks *mask* for front or back facing polygons	Before OpenGL writes data into the stencil buffer, the values should be compared to *mask* for the relevant front or back-facing polygons and masked accordingly
808	glClearStencil	Specify a range of values, *s*	When **glClear** is called, every pixel in the stencil buffer should be set to the value *s*
809	glReadPixels	Pass a range of values *x, y, width, height, format* and *type* to the function	The pixels in the framebuffer rectangle specified by *x, y, width* and *height* should be stored in the array pointed to by **pixels*
901	glFlush	Issue a number of OpenGL commands and then call this function	All previously issued commands should begin execution This may have little or no effect because there is no network and all commands should be truly executed immediately

Test Case ID	Function/Variable Under Test	Description	Expected Result
902	glFinish	Issue a number of OpenGL commands and then call this function	All previously issued commands should begin execution This command should not return until all effects are fully realised
1001	glHint	Pass the hint *GENERATE_MIPMAP_HINT* to the function with *GL_FASTEST* and *GL_NICEST*	OpenGL should perform mipmap generation with either the most efficiency or at the highest quality
1002	glHint	Pass the hint *FRAGMENT_SHADER_DERIVATIVE_HINT* to the function with *GL_FASTEST* and *GL_NICEST*	OpenGL should perform shader derivative calculations with either the most efficiency or at the highest quality
1101	glGetBooleanv	Set up the OpenGL state with a range of functions and variables and check that all states are exposed and readable	For each particular state, *pname*, the relevant value should be returned
1102	glGetIntegerv	Set up the OpenGL state with a range of functions and variables and check that all states are exposed and readable	For each particular state, *pname*, the relevant value should be returned
1103	glGetFloatv	Set up the OpenGL state with a range of functions and variables and check that all states are exposed and readable	For each particular state, *pname*, the relevant value should be returned
1104	glIsEnabled	Enable and disable a range of capabilities, specified by *capability*	If *capability* is enabled, GL_TRUE should be returned If capability is disabled, GL_FALSE should be returned
1105	glGetTexParameter	Set up the OpenGL texture state with a range of variables and check that all states are exposed and readable	For each particular state, *pname*, the relevant value should be returned
1106	glGetBufferParameteriv	Set up the OpenGL buffer state with a range of variables and check that all states are exposed and readable	For each particular state, *pname*, the relevant value should be returned
1107	glIsTexture	After calling **glBindTexture**, check to see if a number of texture and non-texture objects, *textureName*, are actually bound texture objects	GL_TRUE should be returned if *textureName* is the name of a bound texture, otherwise GL_FALSE should be returned
1108	glGetString	Call **glGetString** and check the return values	The correct strings for VENDOR, RENDERER, VERSION, SHADING_LANGUAGE_VERSION and EXTENSIONS should be returned

Test Case ID	Function/Variable Under Test	Description	Expected Result
1109	glIsBuffer	After calling **glBindBuffer**, check to see if a number of buffer and non-buffer objects, *buffer*, are actually buffer objects	GL_TRUE should be returned If *buffer* is the name of a buffer object, otherwise GL_FALSE should be returned
1201	glGetError	Force OpenGL into a range of error states and validate the errors that are returned, NO_ERROR, INVALID_ENUM, INALID_VALUE, INVALID_OPERATION, OUT_OF_MEMORY	The correct errors should be detected and returned
1301	OES_fbo_render_mipmap	Allows rendering to any mipmap level of a texture attached to a framebuffer object Check that the extension is exported	The extension should be exported
1302	OES_rgb8_rgba8	Enables rendering to RGB8 and RGBA8 renderbuffers Check that the extension is exported	The extension should be exported
1303	OES_depth24	Enables rendering to 24 bit depth buffers attached to a framebuffer object Check that the extension is exported	The extension should be exported

Test Case ID	Function/Variable Under Test	Description	Expected Result
1304	OES_depth32	Enables rendering to 32 bit depth buffers attached to a framebuffer object Check that the extension is exported	The extension should be exported
1305	OES_vertex_half_float	Allows 16 but floating point data to be supplied for vertex components Check that the extension is exported	The extension should be exported
1306	OES_texture_float	Allows 16 and 32 bit floating point textures to be supplied to the GL Check that the extension is exported	The extension should be exported
1307	OES_element_index_uint extension	Affects DrawElements() Check that the extension is exported	The extension should be exported
1308	OES_mapbuffer	Allows vertex buffer object buffers to be mapped for the application to write directly into. Check that the extension is exported	The extension should be exported
1309	OES_fragment_precision_high	Allows the high precision qualifier to be used in a fragment shader Check that the extension is exported	The extension should be exported
1310	OES_standard_derivatives	Allows standard GLSL derivatives to be used in a fragment shader Check that the extension is exported	**CURRENTLY UNRATIFIED** The extension should be exported
1312	GL_IMG_texture_compression_pvrtx	Allows PVRTC precompressed data to be supplied to the GL Check that the extension is exported	The extension should be exported
1313	OES_depth_texture	Pass DEPTH_COMPONENT As a format and internal format parameter to glTexImage2D, with 1)Target – 2D target 2)type - UNSIGNED_SHORT and UNSIGNED_INT 3)Width,height –64 - MAX 4)Mipmap level -0	Visual output should be correct and should not generate any error.

Test Case ID	Function/Variable Under Test	Description	Expected Result
1314	OES_depth_texture	Pass DEPTH_COMPONENT As a format and internal format parameter to glsubTexImage2D, with 1) Target – 2D target 2)type - UNSIGNED_SHORT and UNSIGNED_INT 3)Width,height –64 - MAX 4)Mipmap level – 0 5)Xoffset,yoffset – 0-within texture array	Visual output should be correct and should not generate any error.
1315	OES_depth_texture	if the <format> and <internalFormat> is DEPTH_COMPONENT and <type> is not UNSIGNED_SHORT, or UNSIGNED_INT.	INVALID_VALUE should be generated
1316	OES_depth_texture	if the <format> and <internalFormat> is not DEPTH_COMPONENT and <type> is UNSIGNED_SHORT, or UNSIGNED_INT.	error INVALID_VALUE should be generated.
1322	OES_get_program_binary	1) After compiling vertex and frag shader manuaaly try to get their binary using GetProgramBinaryOES and Render scene. 2)Save binary 3)load binary again using glProgramBinaryOES and render scene.	Output should match in both case.
1323	OES_get_program_binary	Pass PROGRAM_BINARY_LENGTH_OES as pname in GetProgramiv, begore and after compile and link shaders.	Initial value should be 0 and after compiling and linking it should generate accurate length.

Test Case ID	Function/Variable Under Test	Description	Expected Result
1324	OES_get_program_binary	1) Pass PROGRAM_BINARY_FORMATS_OES as pname in GetIntegerv, GetBooleanv, GetFloatv 2) Pass NUM_PROGRAM_BINARY_FORMATS_OES as pname in GetIntegerv, GetBooleanv, GetFloatv	1) Should return Enumerated program binary formats. It's default value should be N/A 2) Should return Number of program binary formats It's default value should be 0
1325	OES_get_program_binary	GetProgramBinaryOES is called when the program object, <program>, does not contain a valid program binary	INVALID_OPERATION should generate.
1326	OES_get_program_binary	GetProgramBinaryOES is called with <bufSize> which is not big enough to contain the entire program binary	INVALID_OPERATION should generate.
1327	OES_required_internalformat	Pass following as internal format ALPHA8_OES LUMINANCE8_OES LUMINANCE8_ALPHA8_OES LUMINANCE4_ALPHA4_OES RGB565_OES RGB8_OES RGBA4_OES RGB5_A1_OES RGBA8_OES DEPTH_COMPONENT16_OES DEPTH_COMPONENT24_OES DEPTH_COMPONENT32_OES DEPTH24_STENCIL8_OES RGB10_EXT RGB10_A2_EXT In glTexImage2D with 1)Target – 2D target 2) type,format – From OGLES2 khronos spec (table 3.4) 3)Width,height – 64-MAX 4)level- 0 –n 5)border -0	Visual output should correct. Should not generate any error.

Test Case ID	Function/Variable Under Test	Description	Expected Result
1328	OES_required_internalformat	Pass following as internal format ALPHA8_OES LUMINANCE8_OES LUMINANCE8_ALPHA8_OES LUMINANCE4_ALPHA4_OES RGB565_OES RGB8_OES RGBA4_OES RGB5_A1_OES RGBA8_OES DEPTH_COMPONENT16_OES DEPTH_COMPONENT24_OES DEPTH_COMPONENT32_OES DEPTH24_STENCIL8_OES RGB10_EXT RGB10_A2_EXT In glCopyTexImage2D With 1)target - GL_TEXTURE_2D 2)level – 0 - n 3)x,y -0-within texture array 4)width,height – 64-MAX 5)border -0	Visual output should correct. Should not generate any error.
1329	OES_required_internalformat	TexImage2D is called with a combination of <format>, <type>, and <internalformat> not listed in Table 3.4 of OGLES2 Spec.	INVALID_OPERATION should be Generated.
1330	OES_required_internalformat	RenderbufferStorageOES or CopyTexImage2D is called with a value of <internalformat> not listed in Table 3.4.x or 3.4.y of OGLES2 Spec.	INVALID_ENUM should be generated.
1331	GL_IMG_texture_format_BGRA8888	Pass GL_BGRA as internal and external format of glTexImage2D with 1)Target – 2D target 2)Mipmap level – 0-n 3)Width,height – 64-MAX 4)border -0	Visual output should correct. Should not generate any error.

4.4. OES2_FN06 – Extensions

Test Groups

Test Group ID	Test Group Name
100	glMultiDrawArray
200	glVertexArrayObject

Test Cases

Test Case ID	Function/Variable Under Test	Description	Expected Result
101	glMultiDrawArrays	Query the function in an OpenGL application Pass the different modes GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP,GL_LINES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES, GL_QUAD_STRIP, GL_QUADS, and GL_POLYGON to the function	Function should draw appropriate shapes.
102	glMultiDrawArrays	Testcase 101 with different primcount value.(1,MID_VAL,MAX_VALUE)	Function should draw appropriate shapes.
103	glMultiDrawArrays	Pass non accepted mode value.	Error GL_INVALID_ENUM should generate.
104	glMultiDrawArrays	Pass negative primcount value	Error GL_INVALID_VALUE should generate.
105	glMultiDrawArrays	Draw Array using VBO	Function should draw appropriate shapes.
106	glMultiDrawArrays	Performance check – 1) Draw by using glDrawArrays n times 2) Draw by using single glMultiDrawArrays	Check the time required to take to process both case.
201	glVertexArrayObject()	Query the functions to draw some geometry. (Will try two different VAO).	Function should draw appropriate shapes.
202	glVertexArrayObject()	Invalid values passed to the functions	Error GL_INVALID_VALUE should be generated
203	glBindVertexArrayOES	Invalid values passed to the function	Error GL_INVALID_OPERATION should be generated
204	glGetFloatv, glGetBooleanv, glGetIntegerv	Test of the new flag VERTEX_ARRAY_BINDING_OES	Should not produce error. Should return the VAO ID.

4.5. OES2_FN7 – Ogles2uniflexbench

This test is testing the shaders compilation time of a set of shaders, and the results has to be compared with previous release of the driver.

4.6. OES2_FN8 – ogles2checksymbols

This is a shell script (unix platform only) that is checking the list of exported symbols visible from the API library file. If unnecessary symbols are visible the test will fail.

Appendix A. Function Coverage

Feature	Function/Variable
Vertex Arrays	glVertexAttrib{1234}f[v] glVertexAttribPointer glDrawArrays glDrawElements glEnableVertexAttribArray glDisableVertexAttribArray
Buffer Objects	glBindBuffer glDeleteBuffers glGenBuffers glBufferData glBufferSubData *glMapBuffer # glUnmapBuffer #
Coordinate Transformations	glDepthRangef glViewPort glActiveTexture
Colours and Colouring	glFrontFace
Shaders	glAttachShader glBindAttribLocation glCompilerShader # glCreateProgram glCreateShader glDeleteShader glDetachShader glDeleteProgram glGetActiveAttrib glGetActiveUniform glGetAttribLocation glGetShaderiv (#) glGetShaderInfoLog # glGetShaderPrecisionFormat glGetUniformLocation glLinkProgram glReleaseShaderCompiler # glShaderBinary # glShaderSource # glUniform{1234}{if} glUniform{1234}{if}v glUniformMatrix glUseProgram glValidateProgram

Feature	Function/Variable
Rasterization	glLineWidth glCullFace (enable/disable) glPolygonOffset (enable/disable)
Texturing	glPixelStorei glTexImage2D glTexSubImage2D glCopyTexImage2D glCopyTexSubImage2D glCompressedTexImage2D glCompressedTexSubImage2D glTexParameter{if}[v] glGetTexParameter{if}v glBindTexture glDeleteTextures glGenTextures glIsTexture
Per-Fragment Operations	glScissor (enable/disable) glSampleCoverage (enable/disable) glStencilFunc (enable/disable) glStencilFuncSeparate glStencilOp glStencilOpSeparate glDepthFunc (enable/disable) glBlendFunc (enable/disable) glBlendFuncSeparate glBlendEquation glBlendEquationSeparate glBlendColor
Framebuffer	glColorMask glClear glClearColor glDepthMask glClearDepthf glStencilMask glStencilMaskSeparate glClearStencil glReadPixels
Synchronisation	glFlush glFinish
Hints	glHint

Feature	Function/Variable
State and State Requests	glGetBooleanv glGetIntegerv glGetFloatv glIsEnabled glGetTexParameter glGetBufferParameteriv glIsTexture glGetString glIsBuffer
	glIsShader glIsProgram glGetProgramiv glGetAttachedShaders glGetProgramInfoLog glGetShaderiv glGetShaderInfoLog glGetShaderSource glGetUniform glGetVertexAttrib glGetVertexAttribPointeriv
Errors	glGetError
Extensions	GL_OES_fbo_render_mipmap GL_OES_rgb8_rgba8 GL_OES_depth24 GL_OES_depth32 GL_OES_vertex_half_float GL_OES_texture_float GL_OES_element_index_uint extension GL_OES_mapbuffer GL_OES_fragment_precision_high GL_OES_standard_derivatives ~ GL_IMG_texture_compression_pvrtx

 # Support is not required. If it is supported, the appropriate extension will be exported.

~ Extension has not yet been ratified