RayCore® API 사양

Version 1.0

SILICONARTS

목 차

제 1 장 서론 5

1.1 3D Graphics 5
1.2 RayCore® 5
1.3 RayCore® API 6

제 2 장 함수 목록 7

2.1 rcBindBuffer 8
2.2 rcBindMaterial 9
2.3 rcBindTexture 10
2.4 rcBufferData 11
2.5 rcBufferSubData 12
2.6 rcClearColor 13
2.7 rcColor 14
2.8 rcDeleteBuffers 15
2.9 rcDeleteMaterials 16
2.10 rcDeleteTextures 17
2.11 rcDrawArrays 18
2.12 rcDrawElements 19
2.13 rcEnable 20
2.14 rcEnableClientState 22
2.15 rcFinish 23
2.16 rcFlush 24
2.17 rcFrustum 25
2.18 rcGenBuffers 26
2.19 rcGenMaterials 27
2.20 rcGenTextures 28
2.21 rcGet 29
2.22 rcGetBufferParameteriv 34
2.23 rcGetError 35
2.24 rcGetLight 36
2.25 rcGetMaterial 38
2.26 rcGetPointerv 40
2.27 rcGetString 41
2.28 rcGetTexParameter 42
2.29 rcHint 43
2.30 rcIsBuffer 45

2.31 rcIsEnabled 46
2.32 rcIsMaterial 47
2.33 rcIsTexture 48
2.34 rcLight 49
2.35 rcLoadIdentity 52
2.36 rcLoadMatrix 53
2.37 rcMaterial 54
2.38 rcMatrixMode 56
2.39 rcMultMatrix 57
2.40 rcNormalPointer 58
2.41 rcPushMatrix, rcPopMatrix 59
2.42 rcRotate 60
2.43 rcScale 61
2.44 rcTexCoordPointer 62
2.45 rcTexImage2D 63
2.46 rcTexParameter 65
2.47 rcTranslate 67
2.48 rcVertexPointer 68
2.49 rcViewport 69

제 3 장 확장 함수 목록 70

3.1 rcuLookAt 71
3.2 rcuPerspective 72
3.3 rcCurrentPaletteMatrixOES 73
3.4 rcLoadPaletteFromModelViewMatrixOES 74
3.5 rcMatrixIndexPointerOES 75
3.6 rcWeightPointerOES 76
3.7 rcSceneAllInit 77
3.8 rcStaticSceneBegin, rcStaticSceneEnd 78
3.9 rcTextureAlpha 79
3.10 rcDepthBounce 80

제 1장 서론

1.1 3D Graphics

OpenGL ES 는 임베디드 시스템을 위한 그래픽 하드웨어에 대한 소프트웨어 인터페이스이다. 이 인터페이스는 프로그래머가 높은 품질의 그래픽 이미지(특히, 3D 컬러 이미지) 생성에 관련된 객체와 동작들을 설정할 수 있는 기능과 절차에 대한 집합으로 구성되어 있다.

대부분의 OpenGL ES 는 프레임 버퍼가 포함된 그래픽 하드웨어를 필요로 한다. OpenGL ES 는 대개 점, 선, 다각형 등의 객체 그리기에 관련된 호출이지만, 그림의 일부를 발생시키는 방식(예를 들어 안티 앨리어싱(antialiasing) 및 텍스쳐가 활성화되어 있는 경우)에 있어서는 프레임 버퍼에 의존하고 있다. 그래서, OpenGL ES 의 일부 기능은 프레임 버퍼 조작과 관련이 많다.

OpenGL ES 는 일반적으로 래스터 기반의 하드웨어 가속을 지원한다. 3D 그래픽에서 래스터 하드웨어는 이미지를 생성하기 위해 모든 폴리곤을 처리하는 포워드 렌더링(forward rendering) 방식을 사용한다. 이 방식은 널리 사용되기는 하지만 이미지 품질이 매우 낮다. 이러한 약점을 극복하기 위해서 하드웨어는 쉐이더(shader)같은 추가적인 기능을 지원하게 되었다. 이런 이유로 응용 프로그램을 개발하는 데 많은 시간이 걸리며, 막대한 비용이 발생하게 되었다.

1.2 RayCore®

RayCore®는 주식회사 실리콘아츠에 의해 개발된 레이트레이싱 렌더링 엔진의 등록 상표이다. RayCore®는 래스터 방식처럼 추가적인 처리가 필요 없는 반면, 실시간으로 고품질의 3D 그래픽 이미지를 생성할 수 있다. 따라서, 이를 이용하면 새로운 3D 응용 프로그램 개발이 매우 쉬운 편이다.

지원되는 주요 기능은 다음과 같다.

- 백워드 랜더링(Backward rendering) 방식
- 조명
- 퐁쉐이딩(Phong shading)
- 그림자
- 반사 및 굴절
- 텍스쳐 맵핑(Texture Mapping)
- 안티 앨리어싱(Antialiasing)

래스터 하드웨어에서 일반적으로 사용하는 포워드 렌더링(forward rendering)은 이미지를 생성하기 위해 모든 다각형을 화면 좌표로 변환해야만 하는데, 이는 폴리곤이 랜더링될 때까지 랜더링하기 전의 위치를 알 수 없기 때문이다. 따라서 이런 가시성을 해결하기 위해서는 깊이 테스트(depth test)가 반드시 필요하게 된다. 그러나, RayCore®에서 사용하는 백워드 랜더링(backward rendering)은 이런 깊이 테스트없이 문제의 가시성을 해결할 수 있고, 모든 다각형을 처리하지 않기 때문에 다각형의 처리량을 줄일 수 있다.

기본적으로 조명은 객체가 현실 세계에서처럼 표현되도록 설정되는데, 이로 인해 그림자, 반사, 굴절이 나타난다. 텍스쳐 매핑은 사실적인 3D 그래픽을 위한 기본 기능이다. 안티 앨리어싱은 더 좋은 화면 품질을 지원한다.

RayCore®는 트리(tree)와 같은 가속 구조를 가진 레이트레이싱 알고리즘을 사용하여 3D 그래픽 이미지를 랜더링한다. 트리 가속 구조는 레이트레이싱 처리 과정에서 폴리곤 검색을 빠르게 해 준다. RayCore® API 에는 트리 생성기(Tree Builder)가 소프트웨어적으로 구현되어 있다.

1.3 RayCore® API

RayCore® API 는 OpenGL ES Version 1.1 사양을 참조하여 새로 구현되었다. 레이트레이싱의 특정 기능을 위해서는 새로운 함수나 매개변수를 필요로 하는데, 가능하면 OpenGL ES Version 1.1 의 기능과 매개변수를 그대로 참조하였다. 기본적으로 함수 이름의 접두사를 변경하였고, 매개변수는 최소한으로만 변경하도록 하였다. 또한, 레이트레이싱의 특정 기능에 대해서는 몇 가지 함수를 추가하였다.

지원되는 주요 기능은 다음과 같다.

- 정점, 법선, 텍스쳐 좌표 목록
- Mipmap 텍스쳐
- 삼각형(Triangle), 삼각형 스트립(Triangle strip), 삼각형 팬(Triangle fan)
- 재질 속성(주변광, 확산광, 반사광, 반사, 굴절 등)
- 광원 속성(주변광, 확산광, 반사광 등)
- 매트릭스 모드(Matrix mode) (Push, pop, load 등)

제 2장 함수 목록

이 장에서는 OpenGL ES 1.1 을 참조하여 구성된 RayCore® API 함수 목록을 설명한다. 대부분 이미 정의된 함수를 유사하게 사용하며, 몇몇 함수들은 RayCore®에 적용하기 위해 수정하였다.

2.1 rcBindBuffer

void rcBindBuffer(RCenum *target*, RCuint *buffer*);	
현재 버퍼 대상(*target*)으로 해당 이름(*buffer*)의 버퍼 객체를 설정한다.	
target	버퍼 객체를 설정할 현재 버퍼 대상 (RC_ARRAY_BUFFER, RC_ELEMENT_ARRAY_BUFFER)
buffer	설정할 버퍼 객체의 이름

오류 코드

RC_INVALID_ENUM : *target* 이 허용 값이 아닌 경우
RC_OUT_OF_MEMORY : 버퍼 객체 생성에 실패한 경우

관련 함수

rcBufferData, rcBufferSubData, rcDeleteBuffers, rcGenBuffers
rcGet
    RC_ARRAY_BUFFER_BINDING
    RC_ELEMENT_ARRAY_BUFFER_BINDING

설명

현재 버퍼 대상으로 설정된 버퍼 객체는 데이터 수정 및 조회가 가능해 진다. 이 버퍼 객체는 다른 이름의 버퍼 객체가 같은 버퍼 대상에 설정되거나, 해당 버퍼 객체를 삭제(rcDeleteBuffers 참조)할 때까지 사용이 가능하다.

버퍼 객체의 이름(*buffer*)은 음이 아닌 정수이며, 실제로 사용되는 이름은 양의 정수이다. 0 은 약속된 값으로 *buffer* 에 0 을 설정하면, 현재 버퍼 대상에 대한 버퍼 객체 설정을 초기화한다. 해당 이름의 버퍼 객체가 존재하지 않는 경우 자동으로 버퍼 객체를 생성한다. 새로운 이름의 버퍼 객체들은 rcGenBuffers 를 사용하여 생성할 수 있다. 한번 생성된 이름의 버퍼 객체는 필요에 따라 현재 버퍼 대상에 다시 설정될 수 있다.

rcBindBuffer 를 사용하면, rcVertexPointer, rcTexCoordPointer, rcNormalPointer 의 호출에서 메모리 포인터로 표현되던 rcDrawArrays 의 정점, 법선, 텍스쳐 좌표 배열 포인터 파라미터와 rcDrawElement 의 인덱스 배열 포인터 파라미터가 RayCore® API 에서 관리되는 배열 버퍼(RC_ARRAY_BUFFER) 객체와 인덱스 배열 버퍼(RC_ELEMENT_ARRAY_BUFFER)객체로 표현된다.

2.2 rcBindMaterial

void rcBindMaterial(RCuint *material*);	
현재 물성(material) 객체로 해당 이름(*material*)의 물성 객체를 설정한다.	
material	설정할 물성 객체의 이름

오류 코드
RC_OUT_OF_MEMORY : 물성 객체 생성에 실패한 경우
관련 함수
rcDeleteMaterials, rcGenMaterials

주의

rcBindMaterial 은 RC_TEXTURE_2D 또는 RC_TEXTURE_2D_NORMAL 로 rcEnable, rcDisable 및 rcBindTexture 이 호출되거나, rcMaterial 이 호출되기 전에 반드시 먼저 호출되어야 한다.

설명

현재 물성 객체로 설정된 물성 객체는 데이터 수정 및 조회가 가능해 진다. 이 물성 객체는 다른 이름의 물성 객체가 설정되거나, 해당 물성 객체를 삭제(rcDeleteMaterials 참조)할 때까지 사용이 가능하다.

물성 객체 이름(*material*)은 음이 아닌 정수이며, 0 은 약속된 값으로 물성 연결 상태를 초기화하는 데 사용할 수 있는 기본 물성 객체를 의미한다. 해당 이름의 물성 객체가 존재하지 않는 경우 자동으로 물성 객체를 생성한다. 새로운 이름의 물성 객체들은 rcGenMaterials 를 사용하여 생성할 수 있다. 한번 생성된 이름의 물성 객체는 필요에 따라 현재 물성 객체에 다시 설정될 수 있다.

2.3 rcBindTexture

void rcBindTexture(RCenum *target*, RCuint *texture*);	
텍스쳐 대상(*target*)에 해당 이름(*texture*)의 텍스쳐 객체를 설정한다.	
target	텍스쳐 객체를 설정할 텍스쳐 대상 (RC_TEXTURE_2D, RC_TEXTURE_2D_NORMAL)
texture	설정할 텍스쳐 객체의 이름

오류 코드

RC_INVALID_ENUM : *target* 이 허용값이 아닌 경우
RC_OUT_OF_MEMORY : 텍스쳐 객체 생성에 실패한 경우

관련 함수

rcDeleteTextures, rcGenTextures, rcTexImage2D, rcTexParameter
rcGet
RC_TEXTURE_BINDING_2D

설명

현재 텍스쳐 객체로 설정된 텍스쳐 객체는 데이터 수정 및 조회가 가능해 진다. 이 텍스쳐 객체는 다른 이름의 텍스쳐 객체가 설정되거나, 해당 텍스쳐 객체를 삭제(rcDeleteTextures 참조)할 때까지 사용이 가능하다. 텍스쳐 객체가 설정될 때, 현재 물성 객체에 적용된다. (rcBindMaterial 참조)

텍스쳐 객체 이름(*texture*)은 음이 아닌 정수이며, 0 은 약속된 값으로 텍스쳐 연결 상태를 초기화하는 데 사용할 수 있는 기본 텍스쳐 객체를 의미한다. 해당 이름의 텍스쳐 객체가 존재하지 않는 경우 자동으로 텍스쳐 객체를 생성한다. 새로운 이름의 텍스쳐 객체들은 rcGenTextures 를 사용하여 생성할 수 있다. 한번 생성된 이름의 텍스쳐 객체는 필요에 따라 현재 텍스쳐 객체에 다시 설정될 수 있다.

2.4 rcBufferData

void rcBufferData(RCenum *target*, RCsizeiptr *size*, const RCvoid * *data*, RCenum *usage*);

현재 버퍼 대상(*target*)에 설정된 버퍼 객체의 데이터 저장소를 초기화하거나, 재생성하여 버퍼 데이터 정보를 설정한다.

target	버퍼 데이터 정보를 설정할 현재 버퍼 대상 (RC_ARRAY_BUFFER, RC_ELEMENT_ARRAY_BUFFER)
size	버퍼 객체의 데이터 저장소에 대한 바이트 단위 크기
data	데이터 저장소에 복사할 원본 데이터의 포인터 (데이터가 없는 경우 NULL)
usage	데이터 저장소의 사용 패턴 (RC_STATIC_DRAW, RC_DYNAMIC_DRAW)

오류 코드

RC_INVALID_ENUM
: *target* 이 허용값이 아닌 경우
: *usage* 가 허용값이 아닌 경우
RC_INVALID_VALUE : *size* 가 0 보다 작은 경우
RC_INVALID_OPERATION : *target* 에 버퍼 객체 이름 0 이 연결되는 경우
RC_OUT_OF_MEMORY : 데이터 저장소 생성에 실패한 경우

관련 함수

rcBufferSubData, rcBindBuffer
rcGetBufferParameteriv
    RC_BUFFER_SIZE
    RCL_BUFFER_USAGE

주의

data 가 NULL 인 경우 지정된 크기의 데이터 저장소는 여전히 생성되지만, 그 내용은 초기화되지 않은 상태이며, 따라서 정의되지 않는다.

설명

이전에 존재하던 데이터 저장소는 삭제되고, 새로운 크기(*size*)의 데이터 저장소를 생성한다. 원본 데이터의 포인터(*data*)가 NULL 인 경우 데이터 저장소는 초기화된다.

2.5 rcBufferSubData

void rcBufferSubData(RCenum *target*, RCintptr *offset*, RCsizeiptr *size*, const RCvoid * *data*);

현재 버퍼 대상(*target*)에 설정된 버퍼 객체의 데이터 저장소에 대해서 일부분 또는 전체 버퍼 데이터 정보를 재설정한다.

target	버퍼 데이터 정보를 설정할 현재 버퍼 대상 (RC_ARRAY_BUFFER, RC_ELEMENT_ARRAY_BUFFER)
offset	재설정이 시작될 데이터 저장소의 바이트 단위 오프셋
size	재설정할 데이터 저장소의 바이트 단위 크기
data	데이터 저장소에 복사할 원본 데이터의 포인터

오류 코드

RC_INVALID_ENUM : *target* 이 허용값이 아닌 경우

RC_INVALID_VALUE : *offset* 또는 *size* 가 0 보다 작거나, 두 값이 버퍼 객체의 할당된 데이터 저장소 범위를 벗어나는 경우

RC_INVALID_OPERATION : *target* 에 버퍼 객체 이름 0 이 연결되거나, 버퍼 객체의 *usage* 가 RC_STATIC_DRAW 인 경우

관련 함수

rcBindBuffer, rcBufferData

설명

지정된 메모리 포인터(*data*)의 데이터를 데이터 저장소의 *offset* 에서부터 *size* 만큼 복사한다.

2.6 rcClearColor

void rcClearColor(RCclampf *red*, RCclampf *green*, RCclampf *blue*, RCclampf *alpha*); **void rcClearColorx(RCclampx *red*, RCclampx *green*, RCclampx *blue*, RCclampx *alpha*);**
배경색을 설정한다.

red, green, blue, alpha	배경색의 적색, 녹색, 청색, 알파값

관련 함수

rcClear
rcGet
    RC_COLOR_CLEAR_VALUE

설명

rcClearColor 는 배경색을 설정하기 위해 사용되는 적색, 녹색, 청색, 알파 값들을 설정한다. 초기값은 모두 0 이다. rcClearColor 로 설정되는 값들의 범위는 0 에서 1 이다.

2.7 rcColor

void rcColor4f(RCfloat *red*, RCfloat *green*, RCfloat *blue*, RCfloat *alpha*);
void rcColor4x(RCfixed *red*, RCfixed *green*, RCfixed *blue*, RCfixed *alpha*);
void rcColor4ub(RCubyte *red*, RCubyte *green*, RCubyte *blue*, RCubyte *alpha*);

기본 색상을 설정한다.

red, green, blue, alpha	기본 색상의 적색, 녹색, 청색, 알파값

관련 함수

rcBindMaterial, rcGenMaterial, rcMaterial
rcGet
    RC_CURRENT_COLOR

주의

기본 색상의 초기값은 (1, 1, 1, 1)이다.

설명

현재 물성 객체 속성에서 주변광(ambient)과 확산광(diffuse) 값을 해당 값으로 설정하고, 반사광(specular) 값을 (0,0,0,0)으로 초기화한다.

rcColor4ub 로 설정되는 부호가 없는 바이트 색상 구성은 부동소수점으로 255 를 1.0(전체 강도)에, 0 을 0.0(강도 0)에 각각 선형적으로 매칭된다.

2.8 rcDeleteBuffers

void rcDeleteBuffers(RCsizei *n*, const RCuint * *buffers*);

버퍼 객체 이름 배열(*buffers*)에 들어 있는 *n* 개의 버퍼 객체 이름에 대한 버퍼 객체를 삭제한다.

n	삭제할 버퍼 객체들의 수
buffers	삭제할 버퍼 객체 이름들의 배열

오류 코드

RC_INVALID_VALUE : *n* 이 0 보다 작은 경우

관련 함수

rcBufferData, rcBindBuffer, rcGenBuffers, rcIsBuffer

설명

버퍼 객체 삭제 시 예약된 이름 0 과 아직 생성되어 있지 않은 객체 이름들은 무시된다. 버퍼 객체가 삭제되면, 그 객체에는 아무런 내용이 없으며, 그 이름은 재사용을 위해 자유로워진다. (rcGenBuffers 참조)

버퍼 대상으로 설정된 버퍼 객체가 삭제되는 경우, 해당 객체에 대한 모든 연결 설정은 0 으로 재설정된다.

2.9 rcDeleteMaterials

void rcDeleteMaterials(RCsizei *n*, const RCuint * *materials*);
물성 객체 이름 배열(*materials*)에 들어 있는 *n* 개의 물성 객체 이름에 대한 물성 객체를 삭제한다.

N	삭제할 물성 객체들의 수
materials	삭제할 물성 객체 이름들의 배열

오류 코드
RC_INVALID_VALUE : *n* 이 0 보다 작은 경우
관련 함수
rcBindMaterial, rcGenMaterials, rcIsMaterial

설명

물성 객체 삭제 시 예약된 이름 0 과 아직 생성되어 있지 않은 객체 이름들은 무시된다. 물성 객체가 삭제되면, 그 객체에는 아무런 내용이 없으며, 그 이름은 재사용을 위해 자유로워진다. (rcGenMaterials 참조)

현재 물성 객체로 설정된 물성 객체가 삭제되는 경우, 현재 물성 객체는 0(기본 물성 객체)으로 환원된다.

2.10 rcDeleteTextures

void rcDeleteTextures(RCsizei *n*, const RCuint * *textures*);

텍스쳐 객체 이름 배열(*textures*)에 들어 있는 *n* 개의 텍스쳐 객체 이름에 대한 텍스쳐 객체를 삭제한다.

n	삭제할 텍스쳐 객체들의 수
textures	삭제할 텍스쳐 객체 이름들의 배열

오류 코드

RC_INVALID_VALUE : *n* 이 0 보다 작은 경우

관련 함수

rcBindTexture, rcGenTextures, rcIsTexture

설명

텍스쳐 객체 삭제 시 예약된 이름 0 과 아직 생성되어 있지 않은 객체 이름들은 무시된다. 텍스쳐 객체가 삭제되면, 그 객체에는 아무런 내용이 없으며, 그 이름은 재사용을 위해 자유로워진다. (rcGenTextures 참조)

현재 텍스쳐 객체로 설정된 텍스쳐 객체가 삭제되는 경우, 현재 텍스쳐 객체는 0(기본 텍스쳐 객체)으로 환원된다.

2.11 rcDrawArrays

void rcDrawArrays(RCenum *mode*, RCint *first*, RCsizei *count*);	
미리 설정된 정점, 법선, 색상 및 텍스쳐 좌표에 대한 배열 데이터로부터 렌더링할 primitive 를 생성한다.	
mode	렌더링할 primitive 의 종류 (RC_TRIANGLE_STRIP, RC_TRIANGLE_FAN, RC_TRIANGLES, RC_QUADS)
first	사용될 배열에서의 시작 인덱스
count	primitive 생성에 사용할 인덱스의 수

오류 코드

RC_INVALID_ENUM : *mode* 가 허용 값이 아닌 경우
RC_INVALID_VALUE : *count* 가 0 보다 작은 경우

관련 함수

rcDrawElements, rcNormalPointer, rcTexCoordPointer, rcVertexPointer

설명

별도의 정점, 법선, 색상, 텍스쳐 좌표에 대한 배열을 사전에 미리 설정한 후, rcDrawArrays 라는 하나의 호출로 렌더링할 primitive 를 구성할 수 있다.

각 배열에서 구성요소의 시작 인덱스(*first*)로부터 *count* 만큼 연속적인 요소를 사용하여 기하 primitive 를 생성한다. 여기서, *mode* 는 구성될 primitive 의 종류, 즉 요소 배열에 대한 primitive 의 구성 방법을 의미한다. 만약 정점 배열이 비활성화 상태이면, 아무런 기하 primitive 도 구성되지 않는다.

2.12 rcDrawElements

void rcDrawElements(RCenum *mode*, RCsizei *count*, RCenum *type*, const RCvoid * *indices*);

미리 설정된 정점, 법선, 색상 및 텍스쳐 좌표에 대한 배열 및 인덱스 배열 데이터로부터 렌더링할 primitive 를 생성한다.

mode	렌더링할 primitive 의 종류 (RC_TRIANGLE_STRIP, RC_TRIANGLE_FAN, RC_TRIANGLES, RC_QUADS)
count	primitive 생성에 사용할 인덱스의 수
type	인덱스 배열(*indices*)의 구성요소에 대한 데이터 유형 (RC_BYTE, RC_UNSIGNED_BYTE, RC_SHORT, RC_UNSIGNED_SHORT, RC_INT, RC_UNSIGNED_INT)
indices	인덱스들이 적재된 저장소에 대한 포인터

오류 코드

RC_INVALID_ENUM
: *mode* 가 허용 값이 아닌 경우
: *type* 이 허용 값이 아닌 경우
RC_INVALID_VALUE : *count* 가 0 보다 작은 경우

관련 함수

rcDrawArrays, rcNormalPointer, rcTexCoordPointer, rcVertexPointer

설명

별도의 정점, 법선, 색상 및 텍스쳐 좌표에 대한 배열과 그와 관련된 인덱스 배열을 사전에 미리 설정한 후, rcDrawElements 라는 하나의 호출로 렌더링할 primitive 를 구성할 수 있다.

각 배열의 구성요소에 대한 인덱스 배열 포인터(*indices*)로부터 *count* 만큼 연속적인 인덱스를 사용하여 기하 primitive 를 생성한다. 여기서, *mode* 는 구성될 primitive 의 종류, 즉 요소 배열에 대한 primitive 의 구성 방법을 의미한다. 만약 정점 배열이 비활성화 상태이면, 아무런 기하 primitive 도 구성되지 않는다.

2.13 rcEnable

void rcEnable(RCenum *cap*);
void rcDisable(RCenum *cap*);

RayCore® API 의 다양한 성능을 활성화 및 비활성화시킨다.

cap	RayCore® API 의 성능

오류 코드

RC_INVALID_ENUM : *cap* 이 허용 값이 아닌 경우

관련 함수

rcEnableClientState, rcGet, rcIsEnabled, rcLight, rcMaterial, rcTexImage2D, rcTexSubImage2D

설명

다음은 RayCore® API 의 성능(*cap*)에 대한 설명이다.

- RC_LIGHT*i*
  i 번째 광원에 대한 활성화 상태를 설정한다. (rcLight 참조)

- RC_LIGHTING
  조명에 대한 활성화 상태를 설정한다. (rcLight 참조)

- RC_MATRIX_PALETTE_OES
  palette matrix 에 대한 활성화 상태를 설정한다.
  (rcCurrentPaletteMatrixOES 및 rcLoadPaletteFromModelViewMatrixOES 참조)

- RC_TEXTURE_2D
  2D 텍스쳐링에 대한 활성화 상태를 설정한다. 현재 텍스쳐 객체와 현재 물성객체에 적용된다. (rcBindMaterial 및 rcTexImage2D 참조)

- RC_TEXTURE_2D_NORMAL
  2D 노멀맵 텍스쳐링에 대한 활성화 상태를 설정한다. 현재 텍스쳐 객체와 현재 물성객체에 적용된다. (rcBindMaterial 및 rcTexImage2D 참조)

- RC_USE_COLOR_SHADOW
  그림자에 대한 물성 객체의 색상 적용 유무를 설정한다. 비활성화 상태이면, 단순희 무채색을 적용한다. (rcMaterial 및 rcLight 참조)

- RC_USE_SHADOW
  그림자에 대한 렌더링 유무를 설정한다. 비활성화 상태이면, 그림자를 렌더링하지 않는다.

- RC_USE_TEXTURE_ALPHA_SHADOW
  그림자에 대한 텍스쳐의 알파값 적용 유무를 설정한다. 비활성화 상태이면, 그림자에 단순히 알파값 없는 텍스쳐를 적용한다. (rcMaterial 및 rcTexImage2D

참조)

- RC_USE_TEXTURE_ONLY:
  텍스쳐 객체에 대한 반사 및 그림자 적용 유무를 설정한다. 비활성화 상태이면, 그림자와 반사가 적용된다. (rcMaterial 및 rcTexImage2D 참조)

- RC_USE_TRANSMITTANCE_SHADOW
  그림자에 대한 물성 객체의 투과값 적용 유무를 설정한다. 비활성화 상태이면 투과값이 적용되지 않아 어두운 그림자가 생성된다. (rcMaterial 및 rcTexImage2D 참조)

2.14 rcEnableClientState

void rcEnableClientState(RCenum *array*);

void rcDisableClientState(RCenum *array*);

개별적인 클라이언트 상태를 활성화 또는 비활성화시킨다.

array	클라이언트 상태 (RC_COLOR_ARRAY, RC_MATRIX_INDEX_ARRAY_OES, RC_NORMAL_ARRAY, RC_TEXTURE_COORD_ARRAY, RC_VERTEX_ARRAY, and RC_WEIGHT_ARRAY_OES)

오류 코드

RC_INVALID_ENUM : *array* 가 허용 값이 아닌 경우

관련 함수

rcDrawArrays, rcDrawElements, rcEnable, rcIsEnabled , rcNormalPointer, rcTexCoordPointer, rcVertexPointer, rcMatrixIndexPointerOES, rcWeightPointerOES

설명

기본적으로 모든 클라이언트 상태는 비활성화되어 있다. 다음은 클라이언트 상태(*array*)에 대한 설명이다.

- RC_MATRIX_INDEX_ARRAY_OES
  rcDrawArrays 또는 rcDrawElements 호출 시 렌더링에 사용될 palette matrix 인덱스 배열에 대한 사용 유무를 설정한다. (rcMatrixIndexPointerOES 참조)

- RC_NORMAL_ARRAY
  rcDrawArrays 또는 rcDrawElements 호출 시 렌더링에 사용될 법선 배열에 대한 사용 유무를 설정한다. (rcNormalPointer 참조)

- RC_TEXTURE_COORD_ARRAY
  rcDrawArrays 또는 rcDrawElements 호출 시 렌더링에 사용될 텍스쳐좌표 배열에 대한 사용 유무를 설정한다. (rcTexCoordPointer 참조)

- RC_VERTEX_ARRAY
  rcDrawArrays 또는 rcDrawElements 호출 시 렌더링에 사용될 정점 배열에 대한 사용 유무를 설정한다. (rcVertexPointer 참조)

- RC_WEIGHT_ARRAY_OES
  rcDrawArrays 또는 rcDrawElements 호출 시 렌더링에 사용될 가중치 배열에 대한 사용 유무를 설정한다. (rcWeightPointerOES 참조)

2.15 rcFinish

void rcFinish(void);
레이트레이싱 렌더링을 수행한다.

관련 함수
rcFlush, eglSwapBuffers

설명

RayCore®를 통하여 레이트레이싱 렌더링을 수행한다.

2.16 rcFlush

void rcFlush(void);
rcFinish 와 같은 동작을 수행한다.

관련 함수
rcFinish

설명

rcFlush 는 rcFinish 를 호출한다.

2.17 rcFrustum

void rcFrustumf(RCfloat *left*, RCfloat *right*, RCfloat *bottom*, RCfloat *top*, RCfloat *near*, RCfloat *far*);

void rcFrustumx(RCfixed *left*, RCfixed *right*, RCfixed *bottom*, RCfixed *top*, RCfixed *near*, RCfixed *far*);

절두체(frustum)를 설정한다.

left, right	좌우 수직 절단(clipping) 평면에 대한 좌표
bottom, top	상하 수평 절단(clipping) 평면에 대한 좌표
near, far	근거리와 원거리에 있는 깊이 절단(clipping) 평면까지의 거리 (둘다 반드시 0 보다 커야 한다.)

오류 코드

RC_INVALID_VALUE

: *near* 또는 ***far*** 가 0 보다 크지 않은 경우

: *near* 와 ***far*** 가 같은 경우

: *left* 와 *right* 가 같거나, *bottom* 과 *top* 이 같은 경우

관련 함수

rcViewport

설명

해당 설정값을 통하여 광선 생성이 시작되는 카메라 위치, 화면을 나타낼 절단(clipping) 평면의 픽셀 크기, 카메라와 절단(clipping) 평면의 거리를 계산한다. 여기서, *near* 는 카메라와 화면 사이의 거리이고, *far* 는 사용하지 않는다.

2.18 rcGenBuffers

void rcGenBuffers(RCsizei *n*, RCuint * *buffers*);	
n 개의 새로운 이름들(*buffers*)을 가진 버퍼 객체들을 생성한다.	
n	생성할 버퍼 객체 이름들의 수
buffers	생성된 버퍼 객체 이름들이 적재될 배열

오류 코드

RC_INVALID_VALUE : *n* 이 0 보다 작은 경우
RC_OUT_OF_MEMORY : 버퍼 객체 생성에 실패한 경우

관련 함수

rcBindBuffer, rcBufferData, rcBufferSubData, rcDeleteBuffers, rcIsBuffer

설명

반환되는 버퍼 객체 이름들은 아직 생성되지 않았거나 이미 삭제된 버퍼 객체들로서 항상 연속적인 것은 아니다. 왜냐하면, 이미 생성된 버퍼 객체들은 rcDeleteBuffers 로 삭제하지 않는 한 rcGenBuffers 로 반환되지 않기 때문이다.

2.19 rcGenMaterials

void rcGenMaterials(RCsizei *n*, RCuint * *materials*);

n 개의 새로운 이름들(*materials*)을 가진 물성 객체들을 생성한다.

n	생성할 물성 객체 이름들의 수
materials	생성된 물성 객체 이름들이 적재될 배열

오류 코드

RC_INVALID_VALUE : *n* 이 0 보다 작은 경우
RC_OUT_OF_MEMORY : 물성 객체 생성에 실패한 경우

관련 함수

rcBindMaterial, rcDeleteMaterials, rcIsMaterial

설명

반환되는 물성 객체 이름들은 아직 생성되지 않았거나 이미 삭제된 물성 객체들로서 항상 연속적인 것은 아니다. 왜냐하면, 이미 생성된 물성 객체들은 rcDeleteMaterials 로 삭제하지 않는 한 rcGenMaterials 로 반환되지 않기 때문이다.

2.20 rcGenTextures

void rcGenTextures(RCsizei *n*, RCuint * *textures*);
n 개의 새로운 이름들(*textures*)을 가진 텍스쳐 객체들을 생성한다.

n	생성할 텍스쳐 객체 이름들의 수
textures	생성된 텍스쳐 객체 이름들이 적재될 배열

오류 코드

RC_INVALID_VALUE : *n* 이 0 보다 작은 경우
RC_OUT_OF_MEMORY : 텍스쳐 객체 생성에 실패한 경우

관련 함수

rcBindTexture, rcDeleteTextures, rcIsTexture, rcTexImage2D, rcTexParameter

설명

반환되는 텍스쳐 객체 이름들은 아직 생성되지 않았거나 이미 삭제된 텍스쳐 객체들로서 항상 연속적인 것은 아니다. 왜냐하면, 이미 생성된 텍스쳐 객체들은 rcDeleteTextures로 삭제하지 않는 한 rcGenTextures 로 반환되지 않기 때문이다.

2.21 rcGet

void rcGetBooleanv(RCenum *pname*, RCboolean * *params*);
void rcGetFixedv(RCenum *pname*, RCfixed * *params*);
void rcGetFloatv(RCenum *pname*, RCfloat * *params*);
void rcGetIntegerv(RCenum *pname*, RCint * *params*);

RayCore® API 의 정적 상태 변수(*pname*)에 대한 값(*params*)을 반환한다.

pname	반환할 정적 상태 변수
params	반환될 데이터 유형의 배열 포인터

오류 코드

RC_INVALID_ENUM : *pname* 이 허용 값이 아닌 경우

관련 함수

rcGetError, rcGetString, rcCurrentPaletteMatrixOES,
rcLoadPaletteFromModelViewMatrixOES, rcMatrixIndexPointerOES, rcWeightPointerOES

설명

획득된 값의 유형과 반환값의 유형이 다른 경우, 형 변환을 수행한다.

- GetBooleanv
  부동 소수점 또는 정수는 0 인 경우에만 RC_FALSE 로 변환하고, 그 외에는 RC_TRUE 로 변환한다.

- GetIntegerv
  해당 값이 RGBA 색상 구성 요소가 아니라면, 논리값은 1 또는 0 중 하나로 변환하고, 부동 소수점은 가장 가까운 정수로 반올림한다. 해당 값이 RGBA 색상 구성 요소인 경우, 부동 소수점 1.0 을 255 의 정수로, 부동 소수점 0.을 0 의 정수로 맵핑하는 선형 매핑을 수행한다.

- GetFloatv
  논리값은 부동소수점 1.0 또는 0.0 으로 변환한다.

다음은 정적 상태 변수(*pname*)에 대한 반환값을 설명한 것이다.

- RC_ARRAY_BUFFER_BINDING
  RC_ARRAY_BUFFER 대상에 현재 설정된 버퍼 객체의 이름을 반환한다. 설정된 버퍼 객체가 없는 경우, 0 을 반환한다. (rcBindBuffer 참조)

- RC_COLOR_CLEAR_VALUE
  배경색에 사용되는 적색, 녹색, 청색, 알파 값을 반환한다. (rcClearColor 참조)

- RC_CURRENT_COLOR

기본 색상의 적색, 녹색, 청색, 알파 값을 반환한다. 정수를 요청한 경우, 내부적으로 부동 소수점 1.0 이 255 의 정수로, 부동 소수점 0.0 이 0 의 정수로 맵핑되는 선형 매핑을 통하여 반환한다. (rcColor 참조)

- RC_ELEMENT_ARRAY_BUFFER_BINDING
  RC_ELEMENT_ARRAY_BUFFER 대상에 설정된 버퍼 객체의 이름을 반환한다. 설정된 버퍼 객체가 없을 경우, 0 을 반환한다. (rcBindBuffer 참조)

- RC_LIGHT*i*
  i 번째 광원의 활성화 상태를 반환한다. (rcLight 참조)

- RC_LIGHTING
  조명에 대한 활성화 상태를 반환한다. (rcLight 및 rcMaterial 참조)

- RC_MAX_LIGHTS
  광원의 최대수를 반환한다. 그 값은 8 이다. (rcLight 참조)

- RC_MAX_MODELVIEW_STACK_DEPTH
  모델뷰 행렬 스택에 대한 최대 지원 심도를 반환한다. 그 값은 32 이다. (rcPushMatrix 참조)

- RC_MAX_PALETTE_MATRICES_OES
  팔레트(palette) 행렬의 최대수를 반환한다. 그 값은 128 이다. (rcCurrentPaletteMatrixOES 참조)

- RC_MAX_PROJECTION_STACK_DEPTH
  투영 행렬 스택의 최대 지원 심도를 반환한다. 그 값은 32 이다. (rcPushMatrix 참조)

- RC_MAX_TEXTURE_SIZE
  RayCore®에서 지원하는 최대 텍스쳐 크기를 반환한다. 그 값은 1024 이다. (rcTexImage2D 참조)

- RC_MAX_TEXTURE_STACK_DEPTH
  텍스쳐 행렬 스택의 최대 지원 심도를 반환한다. 그 값은 32 이다. (rcPushMatrix 참조)

- RC_MAX_TEXTURE_UNITS
  RayCore®에서 지원되는 텍스쳐 유닛의 수를 반환한다. 그 값은 1 이다. RayCore® API 에서는 다중 텍스쳐를 지원하지 않는다.

- RC_MAX_THRESHOLD_LEVELS
  레이 바운스(ray bounce)에 대한 임계 레벨의 최대수를 반환한다. 그 값은 10 이다. (rcHint 참조)

- RC_MAX_VERTEX_UNITS_OES
  팔레트(palette) 행렬의 정점 단위의 최대수를 반환한다. 그 값은 128 이다. (rcMatrixIndexPointerOES 및 rcWeightPointerOES 참조)

- RC_MAX_VIEWPORT_DIMS
  뷰포트(viewport) 화면의 해상도로 지원되는 최대 너비 및 높이를 반환한다. 그 값은 2048 이다. (rcViewport 참조)

- RC_MATRIX_INDEX_ARRAY_OES
  팔레트(palette) 행렬 인덱스 배열에 대한 활성화 상태를 반환한다. (rcMatrixIndexPointerOES 참조)

- RC_MATRIX_INDEX_ARRAY_BUFFER_BINDING_OES
  팔레트(palette) 행렬 인덱스 배열 버퍼에 설정된 버퍼 객체의 이름을 반환한다. (rcMatrixIndexPointerOES 참조)

- RC_MATRIX_INDEX_ARRAY_SIZE_OES
  팔레트(palette) 행렬 인덱스 배열에서 정점 당 팔레트(palette) 행렬 인덱스 구성 요소의 수를 반환한다. (rcMatrixIndexPointerOES 참조)

- RC_MATRIX_INDEX_ARRAY_STRIDE_OES
  팔레트(palette) 행렬 인덱스 배열에서 정점 당 연속된 팔레트(palette) 행렬 인덱스 사이의 바이트 단위 간격을 반환한다. (rcMatrixIndexPointerOES 참조)

- RC_MATRIX_INDEX_ARRAY_TYPE_OES
  팔레트(palette) 행렬 인덱스 배열에서 각 구성 요소의 데이터 유형을 반환한다. (rcMatrixIndexPointerOES 참조)

- RC_MATRIX_MODE
  현재 행렬 모드를 반환한다. (rcMatrixMode 참조)

- RC_MATRIX_PALETTE_OES
  팔레트(palette) 행렬에 대한 활성화 상태를 반환한다. (rcEnable 및 rcDisable 참조)

- RC_MODELVIEW_MATRIX
  현재 모델뷰 행렬의 16 개의 값을 반환한다. (rcPushMatrix 참조)

- RC_MODELVIEW_STACK_DEPTH
  모델뷰 행렬 스택에 있는 행렬의 개수를 반환한다. (rcPushMatrix 참조)

- RC_NORMAL_ARRAY
  법선 배열에 대한 활성화 상태를 반환한다. (rcNormalPointer 참조)

- RC_NORMAL_ARRAY_BUFFER_BINDING
  법선 배열에 설정된 버퍼 객체의 이름을 반환한다. (rcNormalPointer 참조)

- RC_NORMAL_ARRAY_STRIDE
  법선 배열에서 연속적인 법선 사이의 바이트 단위 간격을 반환한다. (rcNormalPointer 참조)

- RC_NORMAL_ARRAY_TYPE
  법선 배열에서 각 법선의 데이터 유형을 반환한다. (rcNormalPointer 참조)

- RC_PROJECTION_MATRIX
  현재 투영 행렬의 16 개의 값을 반환한다. (rcPushMatrix 참조)

- RC_PROJECTION_STACK_DEPTH
  투영 행렬 스택에 있는 행렬의 개수를 반환한다. (rcPushMatrix 참조)

- RC_TEXTURE_2D
  2 차원 텍스쳐링에 대한 활성화 상태를 반환한다. (rcTexImage2D 참조)

- RC_TEXTURE_2D_NORMAL
  2 차원 노멀맵 텍스쳐링에 대한 활성화 상태를 반환한다. (rcTexImage2D 참조)

- RC_TEXTURE_BINDING_2D
  현재 텍스쳐 대상(RC_TEXTURE_2D 또는 RC_TEXTURE_2D_NORMAL)에 현재 설정된 텍스쳐 객체의 이름을 반환한다. (rcBindTexture 참조)

- RC_TEXTURE_COORD_ARRAY
  텍스쳐 좌표 배열에 대한 활성화 상태를 반환한다. (rcTexCoordPointer 참조)

- RC_TEXTURE_COORD_ARRAY_BUFFER_BINDING
  텍스쳐 좌표 배열에 설정된 버퍼 객체의 이름을 반환한다. (rcTexCoordPointer 참조)

- RC_TEXTURE_COORD_ARRAY_SIZE
  텍스쳐 좌표 배열에서 구성 요소 당 좌표의 수를 반환한다. (rcTexCoordPointer 참조)

- RC_TEXTURE_COORD_ARRAY_STRIDE
  텍스쳐 좌표 배열에서 연속적인 구성 요소 사이의 바이트 단위 간격을 반환한다. (rcTexCoordPointer 참조)

- RC_TEXTURE_COORD_ARRAY_TYPE
  텍스쳐 좌표 배열에서 각 좌표의 데이터 유형을 반환한다. (rcTexCoordPointer 참조)

- RC_TEXTURE_MATRIX
  현재 텍스쳐 행렬의 16 개의 값을 반환한다. (rcPushMatrix 참조)

- RC_TEXTURE_STACK_DEPTH
  텍스쳐 행렬 스택에 있는 행렬의 개수를 반환한다. (rcPushMatrix 참조)

- RC_USE_COLOR_SHADOW
  색상 그림자에 대한 활성화 상태를 반환한다. (rcEnable 및 rcDisable 참조)

- RC_USE_SHADOW
  그림자에 대한 활성화 상태를 반환한다. (rcEnable 및 rcDisable 참조)

- RC_USE_TEXTURE_ALPHA_SHADOW

알파 텍스쳐 그림자에 대한 활성화 상태를 반환한다. (rcEnable 및 rcDisable 참조)

- RC_USE_TEXTURE_ONLY
  배경 텍스쳐에 대한 활성화 상태를 반환한다. (rcEnable 및 rcDisable 참조)

- RC_USE_TRANSMITTANCE_SHADOW
  투과 그림자에 대한 활성화 상태를 반환한다. (rcEnable 및 rcDisable 참조)

- RC_VIEWPORT
  뷰포트(viewport) 화면에 대한 윈도우 x, y 좌표 및 폭과 너비를 반환한다. (rcViewport 참조)

- RC_VERTEX_ARRAY
  정점 배열에 대한 활성화 상태를 반환한다. (rcVertexPointer 참조)

- RC_VERTEX_ARRAY_BUFFER_BINDING
  정점 배열에 설정된 버퍼 객체의 이름을 반환한다. (rcVertexPointer 참조)

- RC_VERTEX_ARRAY_SIZE
  정점 배열에서 정점 당 좌표의 수를 반환한다. (rcVertexPointer 참조)

- RC_VERTEX_ARRAY_STRIDE
  정점 배열에서 연속적인 정점 사이의 바이트 단위 간격을 반환한다. (rcVertexPointer 참조)

- RC_VERTEX_ARRAY_TYPE
  정점 배열에서 각 정점의 데이터 유형을 반환한다. (rcVertexPointer 참조)

- RC_WEIGHT_ARRAY_OES
  팔레트(palette) 행렬 가중치 배열에 대한 활성화 상태를 반환한다. (rcWeightPointerOES 참조)

- RC_WEIGHT_ARRAY_BUFFER_BINDING_OES
  팔레트(palette) 행렬 가중치 배열에 설정된 퍼버 객체의 이름을 반환한다. (rcWeightPointerOES 참조)

- RC_WEIGHT_ARRAY_SIZE_OES
  팔레트(palette) 행렬 가중치 배열에서 정점 당 팔레트(palette) 행렬 가중치 구성 요소의 수를 반환한다. (rcWeightPointerOES 참조)

- RC_WEIGHT_ARRAY_STRIDE_OES
  팔레트(palette) 행렬 가중치 배열에서 정점 당 연속적인 팔레트(palette) 행렬 가중치 사이의 바이트 단위 간격을 반환한다. (rcWeightPointerOES 참조)

- RC_WEIGHT_ARRAY_TYPE_OES
  팔레트(palette) 행렬 가중치 배열에서 각 구성 요소의 데이터 유형을 반환한다. (rcWeightPointerOES 참조)

2.22 rcGetBufferParameteriv

void rcGetBufferParameteriv(RCenum *target*, RCenum *pname*, RCint * *params*);	
현재 버퍼 대상(*target*)에 설정된 버퍼 객체의 정보를 반환한다.	
target	정보를 요청할 현재 버퍼 대상 (RC_ARRAY_BUFFER, RC_ELEMENT_ARRAY_BUFFER)
pname	버퍼 객체의 정보 파라미터 (RC_BUFFER_SIZE, RC_BUFFER_USAGE)
params	반환할 데이터 유형의 배열 포인터

오류 코드
RC_INVALID_ENUM : *pname* 이 허용 값이 아닌 경우 RC_INVALID_OPERATION *: target* 에 버퍼 객체 이름 0 이 연결되어 있는 경우
관련 함수
rcBufferData, rcBindBuffer

설명

다음은 버퍼 객체의 정보 파라미터(*pname*)에 대한 반환값을 설명한 것이다.

- RC_BUFFER_SIZE
  바이트 단위로 측정되는 버퍼 객체의 크기를 반환한다.

- RC_BUFFER_USAGE
  버퍼 객체의 사용 패턴을 반환한다.

2.23 rcGetError

RCenum rcGetError(void);
현재 설정된 오류 코드를 반환한다.

설명

처음에 오류 플래그는 RC_NO_ERROR 로 설정되어 있다. 동작 중 오류가 발생하면, 오류 플래그는 적절한 오류 코드 값으로 설정된다.

rcGetError 가 호출될 때까지 다른 오류는 기록되지 않는다. 오류 코드를 반환하고 나면, 오류 플래그는 RC_NO_ERROR 로 재설정된다. rcGetError 로 RC_NO_ERROR 가 반환되었다면, rcGetError 가 마지막에 호출된 이후, 또는 RayCore® API 가 초기화된 이후 발생한 오류가 없다는 것을 의미한다. 다음은 오류에 대한 정의이다.

- RC_NO_ERROR
  발생된 오류가 없다. 그 값은 0 이다.
- RC_INVALID_ENUM
  해당 인자에 허용되지 않은 값이 지정되어 있다.
- RC_INVALID_VALUE
  숫자 인수가 범위를 벗어났다.
- RC_INVALID_OPERATION
  지정된 작업이 현재 상태에서 허용되지 않는다.
- RC_STACK_OVERFLOW
  해당 명령이 스택 오버플로우(stack overflow)를 야기한다.
- RC_STACK_UNDERFLOW
  해당 명령이 스택 언더플로우(stack underflow)를 야기한다.
- RC_OUT_OF_MEMORY
  명령을 수행할 충분한 메모리가 남아 있지 않다.

2.24 rcGetLight

void rcGetLightfv(RCenum *light*, RCenum *pname*, RCfloat * *params*);
void rcGetLightxv(RCenum *light*, RCenum *pname*, RCfixed * *params*);

광원에 대한 정보를 반환한다.

light	정보를 요청할 광원, RC_LIGHT*i* (0<= *i* < RC_MAX_LIGHTS) (지원되는 광원의 최대수는 8 이다.)
pname	광원에 대한 정보 파라미터 (RC_AMBIENT, RC_DIFFUSE, RC_SPECULAR, RC_POSITION, RC_SPOT_DIRECTION, RC_SPOT_EXPONENT, RC_SPOT_INNER_CONE, RC_SPOT_OUTER_CONE, RC_SPOT_CUTOFF, RC_ATTENUATION_RANGE, RC_START_ATTENUATION, RC_END_ATTENUATION, RC_CONSTANT_ATTENUATION, RC_LINEAR_ATTENUATION, RC_QUADRATIC_ATTENUATION)
params	반환할 데이터 유형의 배열 포인터

오류 코드

RC_INVALID_ENUM : *light* 또는 *pname* 이 허용 값이 아닌 경우

관련 함수

rcLight

주의

RC_LIGHT*i* 와 RC_LIGHT0+*i* 는 같은 의미이다.

설명

다음은 광원의 정보 파라미터(*pname*)에 대한 반환값을 설명한 것이다.

- RC_AMBIENT
  광원의 주변광에 대한 RGBA 강도를 반환한다.
- RC_DIFFUSE
  광원의 확산광에 대한 RGBA 강도를 반환한다.
- RC_SPECULAR
  광원의 반사광에 대한 RGBA 강도를 반환한다.
- RC_POSITION
  광원의 위치와 유형을 나타내는 4 개의 값을 반환한다. 처음 3 개의 값은 eye 좌표계에서의 광원의 위치에 대한 좌표값이다. 마지막 네번째 값은 광원의 유형(0 : 방향성 광원, 1 : 점 광원)을 나타낸다.
- RC_SPOT_DIRECTION
  광원의 방향 벡터에 대한 3 개의 값을 반환한다.

- RC_SPOT_EXPONENT
  광원의 강도 분포에 대한 집중 지수(spot exponent)를 반환한다.

- RC_SPOT_INNER_CONE
  광원의 강도 분포에 대한 감쇠 시작 각도를 반환한다.

- RC_SPOT_OUTER_CONE, RC_SPOT_CUTOFF
  광원의 강도 분포에 대한 최대 확산 각도를 반환한다.

- RC_ATTENUATION_RANGE
  광원의 세기가 감쇠되는 광원으로부터의 거리 범위를 반환한다. 첫 번째 값은 시작 거리이고, 두 번째 값은 종료 거리이다.

- RC_START_ATTENUATION
  광원의 세기가 감쇠되기 시작하는 광원으로부터 거리를 반환한다.

- RC_END_ATTENUATION
  광원의 세기에 대한 감쇠가 종료되는 광원으로부터 거리를 반환한다.

- RC_CONSTANT_ATTENUATION, RC_LINEAR_ATTENUATION, RC_QUADRATIC_ATTENUATION
  광원의 세기에 대한 세 가지 감쇠 계수 중 하나를 반환한다.

2.25 rcGetMaterial

void rcGetMaterialfv(RCenum *face*, RCenum pname, RCfloat * *params*);
void rcGetMaterialxv(RCenum *face*, RCenum *pname*, RCfixed * *params*);

물성 객체에 대한 정보를 반환한다.

face	정보를 요청할 현재 물성 객체의 면 정보 (RC_FRONT, RC_BACK, RC_FRONT_AND_BACK)
pname	물성 객체에 대한 정보 파라미터 (RC_AMBIENT, RC_DIFFUSE, RC_SHININESS, RC_SPECULAR, RC_SPECULAR_LEVEL, RC_REFLECTION, RC_TRANSMITTANCE, RC_REFRACTION_INDEX)
params	반환할 데이터 유형의 배열 포인터

오류 코드

RC_INVALID_ENUM : *face* 또는 pname 이 허용 값이 아닌 경우

관련 함수

rcBindMaterial, rcGenMaterial, rcMaterial

주의

RayCore® API 에서는 앞면과 뒷면을 공유한 단 하나의 물성 객체만 존재한다. 따라서, RC_FRONT 와 RC_BACK 및 RC_FRONT_AND_BACK 을 쿼리하면 항상 같은 값을 반환한다.

설명

다음은 물성 객체의 정보 파라미터(*pname*)에 대한 반환값을 설명한 것이다.

- RC_AMBIENT
  물성 객체의 주변광 RGBA 반사율을 반환한다.
- RC_DIFFUSE
  물성 객체의 확산광 RGBA 반사율을 반환한다.
- RC_REFLECTION
  물성 객체의 반사율을 반환한다.
- RC_REFRACTION_INDEX
  물성 객체의 굴절률을 반환한다.
- RC_SHININESS
  물성 객체의 정반사 지수를 반환한다.
- RC_SPECULAR
  물성 객체의 반사광 RGBA 반사율을 반환한다.

- RC_SPECULAR_LEVLE
  물성 객체의 정반사 세기를 반환한다.

- RC_TRANSMITTANCE
  물성 객체의 투과율을 반환한다.

2.26 rcGetPointerv

void rcGetPointerv(RCenum *pname*, RCvoid ** *params*);	
현재 배열에 대한 포인터의 메모리 주소를 반환한다.	
pname	현재 배열에 대한 포인터 유형 (RC_COLOR_ARRAY_POINTER, RC_MATRIX_INDEX_ARRAY_POINTER_OES, RC_NORMAL_ARRAY_POINTER, RC_TEXTURE_COORD_ARRAY_POINTER, RC_VERTEX_ARRAY_POINTER, RC_WEIGHT_ARRAY_POINTER_OES)
params	해당 포인터 유형(*pname*)에 설정되어 있는 포인터의 메모리 주소

오류 코드

RC_INVALID_ENUM : *pname* 이 허용 값이 아닌 경우

관련 함수

rcBindBuffer, rcDrawArrays, rcMatrixIndexPointerOES, rcNormalPointer, rcTexCoordPointer, rcVertexPointer, rcWeightPointerOES

2.27 rcGetString

const RCubyte * rcGetString(RCenum *name*);	
RayCore® API 에 대한 정보를 문자열로 반환한다.	
name	정보 유형 (RC_VENDOR, RC_RENDERER, RC_VERSION)

오류 코드
RC_INVALID_ENUM : *name* 이 허용 값이 아닌 경우

주의

오류가 발생하면, rcGetString 은 NULL 을 반환한다.

설명

모든 문자열의 끝은 널(null)로 종료한다. 다음은 정보 유형(*name*)에 대한 반환 문자열을 설명한 것이다.

- RC_VENDOR
  RayCore® API 를 구현한 회사의 이름을 반환한다.

- RC_RENDERER
  렌더러의 이름을 반환한다.

- RC_VERSION
  버전 번호를 반환한다. 그 형태는 "RayCore API <major>.<minor>"이며, <major>, <minor>는 정수로 표현한다. 예를 들어, "RayCore API 1.0"은 <major>가 1 이고, <minor>가 0 이라는 것을 의미한다.

2.28 rcGetTexParameter

void rcGetTexParameterfv(RCenum *target*, RCenum *pname*, RCfloat * *params*);
void rcGetTexParameteriv(RCenum *target*, RCenum *pname*, RCint * *params*);
void rcGetTexParameterxv(RCenum *target*, RCenum *pname*, RCfixed * *params*);

텍스쳐 대상에 대한 정보를 반환한다.

target	정보를 요청할 텍스쳐 대상 (RC_TEXTURE_2D, RC_TEXTURE_2D_NORMAL)
pname	텍스쳐 대상의 정보 파라미터 (RC_TEXTURE_WRAP_S, RC_TEXTURE_WRAP_T, RC_GENERATE_MIPMAP)
params	반환할 데이터 유형의 배열 포인터

오류 코드

RC_INVALID_ENUM : *target* 또는 *pname* 이 허용 값이 아닌 경우

관련 함수

rcTexParameter

설명

다음은 텍스쳐 대상의 정보 파라미터(*pname*)에 대한 반환값을 설명한 것이다.

- RC_TEXTURE_WRAP_S
  텍스쳐 좌표 *s* 의 래핑 기능에 대한 정보를 반환한다. (RC_CLAMP_TO_EDGE, RC_REPEAT)

- RC_TEXTURE_WRAP_T
  텍스쳐 좌표 *t* 의 래핑 기능에 대한 정보를 반환한다. (RC_CLAMP_TO_EDGE, RC_REPEAT)

- RC_GENERATE_MIPMAP
  자동 mipmap 레벨 업데이트에 대한 활성화 상태를 반환한다. (rcTexParameter 참조)

2.29 rcHint

void rcHint(RCenum *target*, RCenum *mode*);	
제어할 동작에 대한 힌트를 설정한다.	
target	제어할 동작 유형 (RC_RENDERING_HINT, RC_RAYBOUNCE_THRESHOLD_HINT, RC_MIPMAP_HINT)
mode	원하는 동작 상태 (RC_FASTEST, RC_NICEST, RC_FASTEST_AND_NICEST, RC_THRESHOLD_LEVELi, RC_DONT_CARE) RC_THRESHOLD_LEVEL*i* (0<= *i* < RC_MAX_THRESHOLD_LEVELS)

오류 코드

RC_INVALID_ENUM : *target* 또는 *mode* 가 허용 값이 아닌 경우

주의

RC_THRESHOLD_LEVEL*i* 와 RC_THRESHOLD_LEVEL0+*i* 는 같은 의미이다.

설명

RayCore® API 에서는 일부 동작에 대한 제어를 힌트로 처리할 수 있다. 다음은 원하는 동작 상태(*mode*)에 대한 설명이다.

- RC_FASTEST
  가장 효율적인 동작 상태를 선택한다.
- RC_NICEST
  최상의 품질이 나타나는 동작 상태를 선택한다.
- RC_FASTEST_AND_NICEST
  가장 효율적이며, 최상의 품질이 나타나는 동작 상태를 선택한다.
- RC_DONT_CARE
  기본 동작 상태를 선택한다.

각 *target* 에 대한 초기값은 RC_DONT_CARE 이다. 동작 유형(*target*)별 동작 상태(*mode*)에 대한 해석은 다음과 같다.

- RC_RENDERING_HINT
  픽셀 샘플링의 렌더링 품질을 나타낸다. 동작 상태 (RC_DONT_CARE, RC_FASTEST, RC_NICEST)를 힌트하여 샘플링 효과의 픽셀당 렌더링 결과를 나타낸다.
- RC_RAYBOUNCE_THRESHOLD_HINT

픽셀 값에 대한 레이 바운스(ray bounce)의 임계 수준을 나타낸다. 임계 수준을 적용하지 않을 경우, RC_DONE_CARE 를 힌트한다. 그렇지 않으면, RC_THRESHOLD_LEVEL*i* 를 힌트하여 임계 수준 효과의 픽셀당 렌더링 결과를 나타낸다.

- RC_MIPMAP_HINT
  물성 객체를 렌더링하는 텍스쳐 품질을 나타낸다. 동작 상태 (RC_DONT_CARE, RC_FASTEST, RC_NICEST)를 힌트하여, 물성 텍스쳐당 mipmap 레벨 사용 결과를 나타낸다.

2.30 rcIsBuffer

RCboolean rcIsBuffer(RCuint *buffer*);	
현재 생성되어 있는 버퍼 객체 중 해당 이름의 버퍼 객체가 존재하는지 여부를 반환한다.	
buffer	버퍼 객체의 이름

관련 함수

rcBindBuffer, rcBufferData, rcBufferSubData, rcDeleteBuffers, rcGenBuffers

설명

buffer 가 현재 생성되어 있는 버퍼 객체의 이름인 경우 RC_TRUE 를 반환한다. 그렇지 않으면, 0 이 아닌 경우 RC_FALSE 를 반환한다.

2.31 rcIsEnabled

RCboolean rcIsEnabled(RCenum *cap*);	
RayCore® API 의 다양한 성능에 대한 활성화 상태를 반환한다.	
cap	RayCore® API 의 성능

오류 코드
RC_INVALID_ENUM : *cap* 이 허용 값이 아닌 경우
관련 함수
rcEnable, rcEnableClientState

주의

오류가 발생하면, rcIsEnabled 는 0 을 반환한다.

설명

다음은 RayCore® API 의 성능(*cap*)에 대한 설명이다.

기호 상수	참조 함수
RC_LIGHT*i*	rcLight
RC_LIGHTING	rcLight, rcMaterial
RC_MATRIX_INDEX_ARRAY_OES	rcEnableClientState
RC_MATRIX_PALETTE_OES	rcEnable
RC_NORMAL_ARRAY	rcNormalPointer
RC_TEXTURE_2D	rcEnable, rcTexImage2D, rcMaterial
RC_TEXTURE_2D_NORMAL	rcEnable, rcTexImage2D, rcMaterial
RC_TEXTURE_COORD_ARRAY	rcTexCoordPointer
RC_USE_COLOR_SHADOW	rcEnable
RC_USE_SHADOW	rcEnable
RC_USE_TEXTURE_ALPHA_SHADOW	rcEnable
RC_USE_TEXTURE_ONLY	rcEnable
RC_USE_TRANSMITTANCE_SHADOW	rcEnable
RC_VERTEX_ARRAY	rcVertexPointer
RC_WEIGHT_ARRAY_OES	rcEnableClientState

2.32 rcIsMaterial

RCboolean rcIsMaterial(RCuint *material*);
현재 생성되어 있는 물성 객체 중 해당 이름의 물성 객체가 존재하는지 여부를 반환한다.

material	물성 객체의 이름

관련 함수

rcBindMatreial, rcDeleteMaterials, rcGenMaterials

설명

material 이 현재 생성되어 있는 물성 객체의 이름인 경우 RC_TRUE 를 반환한다. 그렇지 않으면, 0 이 아닌 경우 RC_FALSE 를 반환한다.

2.33 rcIsTexture

RCboolean rcIsMaterial(RCuint *material*);	
현재 생성되어 있는 물성 객체 중 해당 이름의 물성 객체가 존재하는지 여부를 반환한다.	
material	물성 객체의 이름

관련 함수

rcBindMatreial, rcDeleteMaterials, rcGenMaterials

설명

texture 가 현재 생성되어 있는 텍스쳐 객체의 이름인 경우 RC_TRUE 를 반환한다. 그렇지 않으면, 0 이 아닌 경우 RC_FALSE 를 반환한다.

2.34 rcLight

void rcLightf(RCenum *light*, RCenum *pname*, RCfloat *param*); **void rcLightx(RCenum *light*, RCenum *pname*, RCfixed *param*);**	
광원에 대한 정보를 설정한다.	
light	정보를 설정할 광원, RC_LIGHT*i* (0<= *i* < RC_MAX_LIGHTS) (지원되는 광원의 최대수는 8 이다.)
pname	광원에 대한 정보 파라미터 (RC_SPOT_EXPONENT, RC_SPOT_INNER_CONE, RC_SPOT_OUTER_CONE, RC_SPOT_CUTOFF, RC_START_ATTENUATION, RC_END_ATTENUATION, RC_CONSTANT_ATTENUATION, RC_LINEAR_ATTENUATION, RC_QUADRATIC_ATTENUATION)
param	광원의 정보 파라미터에 설정할 값

void rcLightfv(RCenum *light*, RCenum *pname*, const RCfloat * *params*); **void rcLightxv(RCenum *light*, RCenum *pname*, const RCfixed * *params*);**	
광원에 대한 정보를 설정한다.	
light	정보를 설정할 광원, RC_LIGHT*i* (0<= *i* < RC_MAX_LIGHTS) (지원되는 광원의 최대수는 8 이다.)
pname	광원에 대한 정보 파라미터 (RC_AMBIENT, RC_DIFFUSE, RC_SPECULAR, RC_POSITION, RC_SPOT_DIRECTION, RC_SPOT_EXPONENT, RC_SPOT_INNER_CONE, RC_SPOT_OUTER_CONE, RC_SPOT_CUTOFF, RC_START_ATTENUATION, RC_END_ATTENUATION, RC_CONSTANT_ATTENUATION, RC_LINEAR_ATTENUATION, RC_QUADRATIC_ATTENUATION)
params	광원의 정보 파라미터에 설정할 데이터 유형의 배열 포인터

오류 코드

RC_INVALID_ENUM : *light* 또는 *pname* 이 허용 값이 아닌 경우
RC_INVALID_VALUE
: 집중 지수(spot exponent) 값이 0 보다 작은 경우
: 최대 확산 각도가 0~90 사이 범위(특별히 180 은 제외)가 아닌 경우
: 감쇠 계수가 0 보다 작은 경우

관련 함수

rcEnable, rcMaterial

주의

RC_LIGHT*i* 와 RC_LIGHT0+*i* 는 같은 의미이다. RayCore®의 조명 방정식에서는 알파값을 사용하지 않는다.

설명

활성화된 광원은 조명이 활성화되어 있을 때만 조명 계산에 포함된다. (rcEnable 및 rcDisable 참조) 다음은 광원의 정보 파라미터(*pname*)에 대한 설명이다.

- RC_AMBIENT
  광원의 주변광 RGBA 강도를 설정한다. 초기 주변광 강도는 (1, 1, 1, 0)이다. 설정되는 값들의 범위는 0 에서 1 이다.

- RC_DIFFUSE
  광원의 확산광 RGBA 강도를 설정한다. RC_LIGHT0 에 대한 초기값은 (0.1, 0.1, 0.1, 0)이고, 다른 광원에 대한 초기값은 (0.1, 0.1, 0.1, 0)이다. 설정되는 값들의 범위는 0 에서 1 이다.

- RC_SPECULAR
  광원의 반사광 RGBA 강도를 설정한다. RC_LIGHT0 에 대한 초기값은 (0, 0, 0, 0)이고, 다른 광원에 대한 초기값은 (0, 0, 0, 0)이다. 설정되는 값들의 범위는 0 에서 1 이다.

- RC_POSITION
  광원의 위치와 유형을 설정한다. 처음 3 개의 값인 위치는 모델뷰 행렬에 의해 변환되어 eye 좌표계에서의 좌표값으로 저장된다. 마지막 네번째 값인 광원의 유형(w)은 0 인 경우, 방향성 광원으로 간주되고, 1 인 경우 점 광원으로 간주된다. eye 좌표계에서 광원이 방향 벡터와 실제 위치를 기반으로 하고 있으면, 빛의 감쇠를 사용할 수 있다. 위치와 유형에 대한 초기값은 (0, 0, 1, 0)이다. 즉, $-z$ 축에 평행한 방향성 광원이다.

- RC_SPOT_DIRECTION
  객체 좌표계에서 광원의 방향 벡터를 설정한다. 방향 벡터는 모델뷰 행렬에 의해 변환되어 eye 좌표계에서의 좌표값으로 저장된다. 방향 벡터는 점 광원의 경우에 광원의 최대 확산 각도(RC_SPOT_CUTOFF 참조)가 180(초기값)이 아닌 경우에만 의미가 있다. 초기 방향 벡터는 (0, 0, 0)이다.

- RC_SPOT_EXPONENT
  광원의 강도 분포에 대한 집중 지수(spot exponent)를 설정한다. 광원의 방향 벡터와 정점에 비춰지는 빛 사이의 각도를 θ라고 가정할 때, 광원의 강도 분포는 θ의 코사인 값에 대한 집중 지수(spot exponent)의 거듭제곱승(power)에 의해 감쇠된다. 따라서, 높은 집중 지수(spot exponent)는 좀더 집중된 광원을 나타낼 수 있다. 초기 집중 지수(spot exponent)의 값은 0 으로 균일한 빛 분포를 나타낸다.

- RC_SPOT_INNER_CONE
  광원의 감쇠 시작 각도를 설정한다. 광원의 방향 벡터와 정점에 비춰지는 빛 사이의 각도가 이 값보다 커지게 되면, 빛이 감쇠되기 시작한다. 이 감쇠 강도는 집중 지수(spot exponent)와 감쇠 계수에 의해 결정된다. 이 값의 범위는 0~90 사이이며, 특별히 180 의 값이 허용된다. 초기값은 0 이다.

- RC_SPOT_OUTER_CONE, RC_SPOT_CUTOFF
  광원의 최대 확산 각도를 설정한다. 광원의 방향 벡터와 정점에 비춰지는 빛 사이의 각도가 이 값보다 작은 경우 빛의 감쇠는 계속 이루어진다. 둘 사이의 각도가 이 값보다 커지게 되면, 빛은 완전히 차단된다. 이 감쇠 강도는 집중 지수(spot exponent)와 감쇠 계수에 의해 결정된다. 이 값의 범위는 0~90 사이이며, 특별히 180 의 값이 허용된다. 초기값은 180 으로 균일한 빛 분포를 나타낸다.

- RC_ATTENUATION_RANGE
  광원의 세기가 감쇠되는 광원으로부터의 거리 범위를 설정한다. 첫 번째 값은 시작 거리이고, 두 번째 값은 종료 거리이다.

- RC_START_ATTENUATION
  광원의 세기가 감쇠되기 시작하는 광원으로부터 거리를 설정한다. 초기값은 0 이다.

- RC_END_ATTENUATION
  광원의 세기에 대한 감쇠가 종료되는 광원으로부터 거리를 설정한다. 초기값은 0 이다.

- RC_CONSTANT_ATTENUATION, RC_LINEAR_ATTENUATION, RC_QUADRATIC_ATTENUATION
  광원의 세기에 대한 세 가지 감쇠 계수 중 하나를 설정한다. 이 값들은 0 보다 작지 않은 값이어야 하며, 광원의 위치와 관련되어 있다. 광원과 정점 사이의 거리를 r 이라고 가정할 때, 광원의 세기는 상수(*constant*) 인자, r 을 곱한 선형(*linear*) 인자, r 을 제곱하여 곱한 2 차항(*quadratic*) 인자의 합에 대한 역수의 형태로 감쇠한다. 초기 감쇠 인자는 $(1, 0, 0)$으로 거리에 따른 광원의 세기가 균일하다.

2.35 rcLoadIdentity

void rcLoadIdentity(void);

현재 행렬 모드의 행렬을 4x4 단위행렬로 교체한다.

관련 함수

rcLoadMatrix, rcMatrixMode, rcMultMatrix, rcPushMatrix
rcGet
RC_MATRIX_MODE
RC_MODELVIEW_MATRIX
RC_PROJECTION_MATRIX
RC_TEXTURE_MATRIX
RC_MATRIX_PALETTE_OES

설명

4x4 단위행렬을 rcLoadMatrix 로 호출하는 것과 같다.

현재 행렬 모드의 행렬은 투영 행렬, 모델뷰 행렬, 텍스쳐 행렬, 팔레트(palette) 행렬 중 하나이다. (rcMatrixMode 참조)

2.36 rcLoadMatrix

void rcLoadMatrixf(const RCfloat * *m*);
void rcLoadMatrixx(const RCfixed * *m*);

현재 행렬 모드의 행렬을 m 으로 설정되는 특정 행렬로 교체한다.

m	4x4 열기준 행렬의 구성 요소인 16 개의 값에 대한 배열 포인터

관련 함수

rcLoadIdentity, rcMatrixMode, rcMultMatrix, rcPushMatrix
rcGet
RC_MATRIX_MODE
RC_MODELVIEW_MATRIX
RC_PROJECTION_MATRIX
RC_TEXTURE_MATRIX
RC_MATRIX_PALETTE_OES

설명

현재 행렬 모드의 행렬 M 은 설정되는 배열 m 에 의해 다음과 같이 교체된다.

$$M = \begin{bmatrix} m[0] & m[4] & m[8] & m[12] \\ m[1] & m[5] & m[9] & m[13] \\ m[2] & m[6] & m[10] & m[14] \\ m[3] & m[7] & m[11] & m[15] \end{bmatrix}$$

현재 행렬 모드의 행렬은 투영 행렬, 모델뷰 행렬, 텍스쳐 행렬, 팔레트(palette) 행렬 중 하나이다. (rcMatrixMode 참조)

2.37 rcMaterial

void rcMaterialf(RCenum *face*, RCenum *pname*, RCfloat *param*); **void rcMaterialx(RCenum *face*, RCenum *pname*, RCfixed *param*);**	
현재 물성 객체에 대한 정보를 설정한다.	
face	정보를 설정할 현재 물성 객체의 면 정보 (RC_FRONT_AND_BACK)
pname	물성 객체에 대한 정보 파라미터 (RC_REFLECTION, RC_REFRATION_INDEX, RC_SHININESS, RC_SPECULAR_LEVEL, RC_TRANSMITTANCE)
param	물성 객체의 정보 파라미터에 설정할 값

void rcMaterialfv(RCenum *face*, RCenum *pname*, const RCfloat * *params*); **void rcMaterialxv(RCenum *face*, RCenum *pname*, const RCfixed * *params*);**	
현재 물성 객체에 대한 정보를 설정한다.	
face	정보를 설정할 현재 물성 객체의 면 정보 (RC_FRONT_AND_BACK)
pname	물성 객체에 대한 정보 파라미터 (RC_AMBIENT, RC_AMBIENT_AND_DIFFUSE, RC_DIFFUSE, RC_REFLECTION, RC_REFRATION_INDEX, RC_SHININESS, RC_SPECULAR, RC_SPECULAR_LEVEL, RC_TRANSMITTANCE)
param	물성 객체의 정보 파라미터에 설정할 데이터 유형의 배열 포인터

오류 코드

RC_INVALID_ENUM : *face* 또는 *pname* 이 허용 값이 아닌 경우
RC_INVALID_VALUE
: 정반사 지수(specular exponent)가 0 보다 작은 경우
: 정반사 세기(specular intensity)가 0 보다 작은 경우

관련 함수

rcEnable, rcGetMaterial, rcLight

주의

RayCore®의 조명 방정식에서는 알파값을 사용하지 않는다.

설명

물성 객체의 정보들은 관련된 primitive 에 적용되는 조명 방정식에서 사용된다. 다음은 물성 객체의 정보 파라미터(*pname*)에 대한 설명이다.

- RC_AMBIENT
  물성 객체의 주변광 RGBA 반사율을 설정한다. 초기 주변광 반사율은 (0.2, 0.2, 0.2, 0)이다. 설정되는 값들의 범위는 0 에서 1 이다.

- RC_AMBIENT_AND_DIFFUSE

RC_AMBIENT 및 RC_DIFFUSE 각각에 대해 같은 파라미터 값들로 rcMaterial 을 호출한 것과 같다.

- RC_DIFFUSE
  물성 객체의 확산광 RGBA 반사율을 설정한다. 초기 확산광 반사율은 (0.8, 0.8, 0.8, 0)이다. 설정되는 값들의 범위는 0 에서 1 이다.

- RC_REFLECTION
  물성 객체의 반사율을 설정한다. 초기 반사율은 0 이다.

- RC_REFRACTION_INDEX
  물성 객체의 굴절률을 설정한다. 초기 굴절률은 1 이다.

- RC_SHININESS
  물성 객체의 정반사 지수를 설정한다. 초기 정반사 지수는 0 이다.

- RC_SPECULAR
  물성 객체의 반사광 RGBA 반사율을 설정한다. 초기 반사광 반사율은 (0, 0, 0, 0)이다. 설정되는 값들의 범위는 0 에서 1 이다.

- RC_SPECULAR_LEVEL
  물성 객체의 정반사 세기를 설정한다. 초기 정반사 세기는 0 이다.

- RC_TRANSMITTANCE
  물성 객체의 투과율을 설정한다. 초기값은 0 이다.

2.38 rcMatrixMode

void rcMatrixMode(RCenum *mode*);	
현재 행렬이 되는 현재 행렬 모드를 설정한다.	
mode	현재 행렬을 설정할 행렬 모드 (RC_MODELVIEW, RC_PROJECTION, RC_TEXTURE)

오류 코드

RC_INVALID_ENUM : *mode* 가 허용 값이 아닌 경우

관련 함수

rcLoadMatrix, rcMultMatrix, rcPushMatrix

rcGet

RC_MATRIX_MODE

설명

현재 행렬 모드(*mode*)를 설정하면, 후속 행렬 작업에서 사용할 현재 행렬 및 행렬 스택이 결정된다. 초기 행렬 모드는 RC_MODELVIEW 이다.

다음은 행렬 모드(*mode*)에 대한 설명이다.

- RC_MODELVIEW
  후속 행렬 작업을 현재 모델뷰 행렬 및 행렬 스택에 적용한다.
- RC_PROJECTION
  후속 행렬 작업을 현재 투영 행렬 및 행렬 스택에 적용한다.
- RC_TEXTURE
  후속 행렬 작업을 현재 텍스쳐 행렬 및 행렬 스택에 적용한다.
- RC_MATRIX_PALETTE_OES
  후속 행렬 작업을 현재 팔레트(palette) 행렬 및 행렬 스택에 적용한다.

2.39 rcMultMatrix

void rcMultMatrixf(const RCfloat * *m*);
void rcMultMatrixx(const RCfixed * *m*);

현재 행렬 모드의 행렬에 m 으로 설정되는 특정 행렬을 곱한다.

m	4x4 열기준 행렬의 구성 요소인 16 개의 값에 대한 배열 포인터

관련 함수

rcLoadIdentity, rcLoadMatrix, rcMatrixMode, rcPushMatrix
rcGet
RC_MATRIX_MODE
RC_MODELVIEW_MATRIX
RC_PROJECTION_MATRIX
RC_TEXTURE_MATRIX
RC_MATRIX_PALETTE_OES

설명

현재 행렬 모드의 행렬은 투영 행렬, 모델뷰 행렬, 텍스쳐 행렬 중 하나이다. (rcMatrixMode 참조) 현재 행렬 모드의 행렬 M 에 배열 m 으로 구성되는 행렬을 곱하여 현재 행렬 모드의 새로운 행렬 M'을 생성한다.

$$M' = \begin{bmatrix} M[0] & M[4] & M[8] & M[12] \\ M[1] & M[5] & M[9] & M[13] \\ M[2] & M[6] & M[10] & M[14] \\ M[3] & M[7] & M[11] & M[15] \end{bmatrix} \begin{bmatrix} m[0] & m[4] & m[8] & m[12] \\ m[1] & m[5] & m[9] & m[13] \\ m[2] & m[6] & m[10] & m[14] \\ m[3] & m[7] & m[11] & m[15] \end{bmatrix}$$

2.40 rcNormalPointer

void rcNormalPointer(RCenum *type*, RCsizei *stride*, const RCvoid * *pointer*);	
렌더링에 사용할 법선 배열의 데이터 및 메모리 주소를 설정한다.	
type	배열 내 각 좌표의 데이터 유형 (RC_BYTE, RC_SHORT, RC_FLOAT, RC_FIXED)
stride	연속적인 법선들 사이의 바이트 단위 간격 (만약 값이 0 이면, 법선들이 배열 내에 빈 공간 없이 적재되어 있음을 의미한다.)
pointer	배열에서 첫 번째 법선의 첫 번째 구성 요소에 대한 포인터

오류 코드

RC_INVALID_ENUM : *type* 이 허용 값이 아닌 경우
RC_INVALID_VALUE : *stride* 가 0 보다 작은 경우

관련 함수

rcDrawArrays, rcDrawElements, rcEnable, rcTexCoordPointer, rcVertexPointer

설명

법선 정보는 색상, 정점, 텍스쳐 좌표와 함께 하나의 배열 안에 모두 적재되거나, 서로 구분된 배열에 따로 적재되어 사용될 수 있다. 법선 배열이 활성화 상태이면, rcDrawArrays 또는 rcDrawElements 가 호출될 때 이 배열이 사용된다. (rcEnableClientState 및 rcDisableClientState 참조)

2.41 rcPushMatrix, rcPopMatrix

void rcPushMatrix(void);
void rcPopMatrix(void);

현재 행렬 모드의 행렬 스택을 푸시(push) 및 팝(pop) 시킨다.

오류 코드

RC_STACK_OVERFLOW : rcPushMatrix 호출 시 현재 행렬 스택이 차 있을 경우
RC_STACK_UNDERFLOW : rcPopMatrix 호출 시 현재 행렬 스택이 비어 있을 경우

관련 함수

rcLoadIdentity, rcLoadMatrix, rcMatrixMode, rcMultMatrix, rcRotate, rcScale, rcTranslate
rcGet
RC_MAX_MODELVIEW_STACK_DEPTH
RC_MAX_PROJECTION_STACK_DEPTH
RC_MAX_TEXTURE_STACK_DEPTH
RC_MAX_TEXTURE_UNITS

설명

rcPushMatrix 는 현재 행렬을 복사하여 현재 행렬 스택에 푸시다운(push-down)시킨다. rcPopMatrix 는 현재 행렬 스택의 최상단에 있는 행렬로 현재 행렬을 교체하고, 현재 행렬 스택을 팝(pop)시킨다. 초기에 모든 행렬 스택은 비어 있다.

2.42 rcRotate

void rcRotatef(RCfloat *angle*, RCfloat *x*, RCfloat *y*, RCfloat *z*);
void rcRotatex(RCfixed *angle*, RCfixed *x*, RCfixed *y*, RCfixed *z*);

현재 행렬 모드의 행렬에 회전 행렬을 곱한다.

angle	60 진법 각도(degree)로 표현된 회전각
x, y, z	회전의 기준이 되는 벡터의 *x, y, z* 좌표

관련 함수

rcMatrixMode, rcMultMatrix, rcPushMatrix, rcScale, rcTranslate
rcGet
RC_MATRIX_MODE
RC_MODELVIEW_MATRIX
RC_PROJECTION_MATRIX
RC_TEXTURE_MATRIX

주의

RayCore® API 에서 회전은 기본적으로 오른손 법칙을 따른다. 즉, 벡터 (*x, y, z*)가 사용자 쪽을 향할 경우, 회전 방향은 반시계 방향이다.

설명

현재 행렬 모드의 행렬은 투영 행렬, 모델뷰 행렬, 텍스쳐 행렬 중 하나이다. (rcMatrixMode 참조) 정의되는 회전 행렬 R 은 벡터 (*x, y, z*)를 기준으로 회전각(angle)만큼 회전하는 행렬이다. 벡터 (*x, y, z*)는 크기가 1 이 되도록 정규화한다. cos(*angle*)를 *a*, sin(*angle*)를 *b* 라고 하면, 회전행렬 *R* 은 다음과 같이 표현된다.

$$R = \begin{bmatrix} (1-a)x^2 + a & (1-a)xy - bz & (1-a)xz + by & 0 \\ (1-a)xy + bz & (1-a)y^2 + ay & (1-a)yz - bx & 0 \\ (1-a)xz - by & (1-a)yz + bx & (1-a)z^2 + a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

회전을 적용하기 전의 좌표계를 보관하였다가 다시 복원하고 싶으면, rcPushMatrix 및 rcPopMatrix 를 사용하면 된다.

2.43 rcScale

void rcScalef(RCfloat *x*, RCfloat *y*, RCfloat *z*);
void rcScalex(RCfixed *x*, RCfixed *y*, RCfixed *z*);

현재 행렬 모드의 행렬에 스케일링 행렬(scaling matrix)을 곱한다.

x, y, z	*x, y, z* 축에 대한 스케일(scale) 계수

관련 함수

rcEnable, rcMatrixMode, rcMultMatrix, rcPushMatrix, rcRotate, rcTranslate
rcGet
RC_MATRIX_MODE
RC_MODELVIEW_MATRIX
RC_PROJECTION_MATRIX
RC_TEXTURE_MATRIX

주의

스케일링 행렬이 모델뷰 행렬에 적용되는 경우, 광원에 대한 정보가 간혹 잘못 나타날 수 있다.

설명

현재 행렬 모드의 행렬은 투영 행렬, 모델뷰 행렬, 텍스쳐 행렬 중 하나이다. (rcMatrixMode 참조) 정의되는 스케일링 행렬 S 는 *x, y, z* 축에 따라 균일하지 않게 스케일링한다. 스케일링 행렬 S 는 다음과 같이 표현된다.

$$S = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

스케일링을 적용하기 전의 좌표계를 보관하였다가 다시 복원하고 싶으면, rcPushMatrix 및 rcPopMatrix 를 사용하면 된다.

2.44 rcTexCoordPointer

void rcTexCoordPointer(RCint *size*, RCenum *type*, RCsizei *stride*, const RCvoid * *pointer*);

렌더링에 사용할 텍스쳐 좌표 배열의 데이터 및 메모리 주소를 설정한다.

size	한 텍스쳐 좌표에 대한 구성 요소의 수 (반드시 2)
type	배열 내 각 텍스쳐 좌표의 데이터 유형 (RC_BYTE, RC_SHORT, RC_FLOAT, RC_FIXED)
stride	연속적인 텍스쳐 좌표들 사이의 바이트 단위 간격 (만약 값이 0 이면, 텍스쳐 좌표들이 배열 내에 빈 공간 없이 적재되어 있음을 의미한다.)
pointer	배열에서 첫 번째 텍스쳐 좌표의 첫 번째 구성 요소에 대한 포인터

오류 코드

RC_INVALID_VALUE *: size* 가 2 가 아닌 경우
RC_INVALID_ENUM : *type* 이 허용 값이 아닌 경우
RC_INVALID_VALUE : *stride* 가 0 보다 작은 경우

관련 함수

rcDrawArrays, rcDrawElements, rcEnable, rcNormalPointer, rcVertexPointer

설명

텍스쳐 좌표는 색상, 정점, 법선과 함께 하나의 배열 안에 모두 적재되거나, 서로 구분된 배열에 따로 적재되어 있을 수 있다. 텍스쳐 좌표 배열이 활성화 상태이면, rcDrawArrays 또는 rcDrawElements 가 호출될 때 이 배열이 사용된다. (rcEnableClientState 및 rcDisableClientState 참조)

2.45 rcTexImage2D

void rcTexImage2D(RCenum *target*, RCint *level*, RCint *internalformat*, RCsizei *width*, RCsizei *height*, RCint *border*, RCenum *format*, RCenum *type*, const RCvoid * *pixels*);

현재 텍스쳐 객체에 2 차원 텍스쳐 이미지를 설정한다.

target	텍스쳐 대상 (RC_TEXTURE_2D, RC_TEXTURE_2D_NORMAL)
level	LOD(level-of-detail) 레벨 (반드시 0)
internalformat	텍스쳐의 색상 구성 유형 (RC_RGB, RC_RGBA)
width	텍스쳐 이미지의 너비 (반드시 2 의 배수, 최소 16 이상, 최대 RC_MAX_TEXTURE_SIZE 이하)
height	텍스쳐 이미지의 높이 (반드시 2 의 배수이며, 최소 16 이상, 최대 RC_MAX_TEXTURE_SIZE 이하)
border	윤곽선의 너비 (반드시 0)
format	픽셀 데이터의 색상 구성 유형 (RC_RGB, RC_RGBA)
type	픽셀 데이터의 데이터 유형 (RC_BYTE, RC_UNSIGNED_BYTE, RC_SHORT, RC_UNSIGNED_SHORT)
pixels	픽셀 데이터가 적재되어 있는 메모리 주소

오류 코드

RC_INVALID_ENUM
: *target* 이 허용값이 아닌 경우
: *internalformat* 이나 *format* 이 허용 값이 아닌 경우
: *type* 이 허용 값이 아닌 경우

RC_INVALID_VALUE
: *level* 이 0 이 아닌 경우
: *width* 및 *height* 가 16 보다 작거나, RC_MAX_TEXTURE_SIZE 보다 큰 경우 또는 2 의 배수가 아닌 경우
: *border* 가 0 이 아닌 경우

RC_INVALID_OPERATION : *internalformat* 과 *format* 이 동일한 값이 아닌 경우

관련 함수

rcBindTexture,
rcGet
    RC_MAX_TEXTURE_SIZE

주의

pixels 가 NULL 인 경우 텍스쳐가 적용되지 않는다.

설명

색상 구성 유형이 RC_RGB 인 경우, RayCore® API 에서는 rcTextureAlpha 로 설정되는 알파값을 조합하여 RGBA 형태로 변경한다.

색상 구성 유형(*internalformat, format*)은 다음과 같다.

- RC_RGB
  각 색상이 RGB(적색, 녹색, 청색)로 구성되어 있다.

- RC_RGBA
  각 색상이 RGBA(적색, 녹색, 청색, 알파값)로 구성되어 있다.

2.46 rcTexParameter

void rcTexParameterf(RCenum *target*, RCenum *pname*, RCfloat *param*);
void rcTexParameteri(RCenum *target*, RCenum *pname*, RCint *param*);
void rcTexParameterx(RCenum *target*, RCenum *pname*, RCfixed *param*);

텍스쳐 대상에 대한 정보를 설정한다.

target	정보를 설정할 텍스쳐 대상 대상 (RC_TEXTURE_2D, RC_TEXTURE_2D_NORMAL)
pname	텍스쳐 대상의 정보 파라미터 (RC_TEXTURE_WRAP_S, RC_TEXTURE_WRAP_T, RC_GENERATE_MIPMAP)
param	텍스쳐 대상의 정보 파라미터에 설정할 값

void rcTexParameterfv(RCenum target, RCenum pname, RCfloat * params);
void rcTexParameteriv(RCenum target, RCenum pname, RCint * params);
void rcTexParameterxv(RCenum target, RCenum pname, RCfixed * params);

텍스쳐 대상에 대한 정보를 설정한다.

target	정보를 설정할 텍스쳐 대상 대상 (RC_TEXTURE_2D, RC_TEXTURE_2D_NORMAL)
pname	텍스쳐 대상의 정보 파라미터 (RC_TEXTURE_WRAP_S, RC_TEXTURE_WRAP_T, RC_GENERATE_MIPMAP)
params	텍스쳐 대상의 정보 파라미터에 설정할 데이터 유형의 배열 포인터

오류 코드

RC_INVALID_ENUM : *target* 및 *pname* 이 허용 값이 아닌 경우

관련 함수

rcBindTexture, rcEnable, rcTexImage2D, rcTexSubImage2D

설명

다음은 텍스쳐 대상의 정보 파라미터(*pname*)에 대한 설명이다.

- RC_TEXTURE_WRAP_S
  텍스쳐 좌표 *s* 의 래핑 모드를 RC_CLAMP_TO_EDGE 또는 RC_REPEAT 로 설정한다. 초기값은 RC_REPEAT 이다.

- RC_CLAMP_TO_EDGE
  텍스쳐 좌표 *s* 가 $[1/2N, 1 - 1/2N]$ 범위로 고정된다. 여기서, N 은 고정 방향으로의 텍스쳐 크기이다.

- RC_REPEAT
  텍스쳐 좌표 *s* 의 정부 부분은 무시되고, 소수 부분만 사용한다. 결과적으로 반복적인 이미지 패턴이 생성된다.

- RC_TEXTURE_WRAP_T

텍스쳐 좌표 *t* 의 래핑 모드를 RC_CLAMP_TO_EDGE 또는 RC_REPEAT 로 설정한다. (RC_TEXTURE_WRAP_S 내용 참조) 초기값은 RC_REPEAT 이다.

- RC_GENERATE_MIPMAP
  자동 mipmap 레벨 업데이트에 대한 활성화 상태를 설정한다. 상태를 활성화시키면, 기본 레벨의 mipmap 이 수정될 경우 mipmap 배열의 모든 레벨이 자동으로 업데이트 된다. 기본값은 RC_TRUE 이다.

2.47 rcTranslate

void rcTranslatef(RCfloat *x*, RCfloat *y*, RCfloat *z*);
void rcTranslatex(RCfixed *x*, RCfixed *y*, RCfixed *z*);

현재 행렬 모드의 행렬에 이동 행렬을 곱한다.

x, y, z	이동 벡터의 *x, y, z* 좌표

관련 함수

rcMatrixMode, rcMultMatrix, rcPushMatrix, rcRotate, rcScale
rcGet
RC_MATRIX_MODE
RC_MODELVIEW_MATRIX
RC_PROJECTION_MATRIX
RC_TEXTURE_MATRIX

설명

현재 행렬 모드의 행렬은 투영 행렬, 모델뷰 행렬, 텍스쳐 행렬 중 하나이다. (rcMatrixMode 참조) 정의되는 이동 행렬 T 는 벡터(*x, y, z*)에 의해 이동되는 행렬이다. 이동 행렬 *T* 는 다음과 같다.

$$T = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

이동을 적용하기 전의 좌표계를 보관하였다가 다시 복원하고 싶으면, rcPushMatrix 및 rcPopMatrix 를 사용하면 된다.

2.48 rcVertexPointer

void rcVertexPointer(RCint *size*, RCenum *type*, RCsizei *stride*, const RCvoid * *pointer*);

렌더링에 사용할 정점 좌표 배열의 데이터 및 메모리 주소를 설정한다.

size	정점당 좌표의 수 (반드시 3)
type	배열 내 각 정점 좌표의 데이터 유형 (RC_BYTE, RC_SHORT, RC_FLOAT, RC_FIXED)
stride	연속적인 정점 사이의 바이트 단위 간격 (만약 값이 0 이면, 정점들이 배열 내에 빈 공간 없이 적재되어 있음을 의미한다.)
pointer	배열에서 첫 번째 정점의 첫 번째 좌표에 대한 포인터

오류 코드

RC_INVALID_ENUM : *type* 이 허용 값이 아닌 경우
RC_INVALID_VALUE
: *size* 가 3 이 아닌 경우
: *stride* 가 0 보다 작은 경우

관련 함수

rcDrawArrays, rcDrawElements, rcEnable, rcNormalPointer, rcTexCoordPointer

설명

정점 좌표는 색상, 법선, 텍스쳐 좌표와 함께 하나의 배열 안에 모두 적재되거나, 서로 구분된 배열에 따로 적재되어 있을 수 있다. 정점 좌표 배열이 활성화되면, rcDrawArrays 또는 rcDrawElements 가 호출될 때 이 배열이 사용된다. (rcEnableClientState 및 rcDisableClientState 참조)

2.49 rcViewport

void rcViewport(RCint *x*, RCint *y*, RCsizei *width*, RCsizei *height*);	
뷰포트(viewport)에 대한 화면 정보를 설정한다.	
x, y	화면의 좌하단 픽셀 좌표 (반드시 (0, 0))
width, height	화면에 대한 픽셀단위 너비 및 높이

오류 코드
RC_INVALID_VALUE : *width* 또는 *height* 가 0 보다 작은 경우
관련 함수
rcGet

설명

너비와 높이는 화면 해상도를 나타낸다. 이를 통하여 생성될 광선의 수가 결정된다. 너비 및 높이의 허용 범위는 RC_MAX_VIEWPORT_DIMS 인자로 rcGet 을 호출하면 조회할 수 있다.

제 3장 확장 함수 목록

이 장에서는 RayCore® API 확장 함수 목록을 설명한다. OpenGL ES 1.1 에서 이미 정의된 확장 함수를 유사하게 사용하였으며, 몇몇 함수들은 RayCore®에 적용하기 위해 수정 및 추가하였다.

3.1 rcuLookAt

void rcuLookAt(RCfloat *eyex*, RCfloat *eyey*, RCfloat *eyez*, RCfloat *centerx*, RCfloat *centery*, RCfloat *centerz*, RCfloat *upx*, RCfloat *upy*, RCfloat *upz*);	
시점에 대한 정보를 통하여 카메라 좌표계를 설정한다.	
eyeX, eyeY, eyeZ	시점(eye point)의 위치
centerX, centerY, centerZ	장면의 중심을 나타내는 기준점(reference point)의 위치
upX, upY, upZ	업(*UP*) 벡터의 방향성

관련 함수

rcFrustum , rcuPerspective

설명

설정되는 시점, 기준점, 업(*UP*) 벡터를 이용하여 카메라 좌표계를 정의한다. 여기서 카메라 좌표계의 X 축은 우(*RIGHT*) 벡터, Y 축은 업(*UP*) 벡터, Z 축은 시선(*GAZE*) 벡터라고 부른다.

G 를 다음과 같이 표현하고,

$$G = \begin{bmatrix} eyeX - centerX \\ eyeY - centerY \\ eyeZ - centerZ \end{bmatrix}$$

벡터 (*upX, upY, upZ*)를 UP 라고 하면, 정규화는 다음과 같다.

$$g = \frac{G}{||G||}$$

$$u = \frac{UP}{||UP||}$$

마지막으로 u 와 f 의 외적을 R 이라 하면,

$$R = u \times f$$

$$r = \frac{R}{||R||}$$

이 된다. 이는 카메라 좌표계의 각 기준 축을 나타내는데, r 이 X 축을, u 가 Y 축을, f 가 Z 축을 나타내는 단위 벡터이다.

3.2 rcuPerspective

void rcuPerspective(RCfloat *fovy*, RCfloat *aspect*, RCfloat *zNear*, RCfloat *zFar*);	
절대 좌표계에서의 카메라 절두체(viewing frustum) 정보를 설정한다.	
fovy	*y* 방향에서의 60 진법 각도(degree) 단위의 시야각
aspect	*x* 방향에서 시야각을 결정하는 종횡비 (*y*(높이)에 대한 *x*(너비)의 비율)
zNear	관찰자(viewer)로부터 근거리 절단(clipping) 평면까지의 거리 (항상 0 보다 크다.)
zFar	관찰자(viewer)로부터 원거리 절단(clipping) 평면까지의 거리 (항상 0 보다 크다.)

오류 코드

RC_INVALID_VALUE
: *zNear* 또는 ***zFar*** 가 0 보다 크지 않은 경우
: *zNear* 와 *zFar* 가 같은 경우

관련 함수

rcFrustum, rcLoadIdentity, rcMultMatrix

주의

zNear 는 반드시 0 이 아니어야 한다.

설명

일반적으로 종횡비는 뷰포트(viewport)의 종횡비와 일치한다. *aspect* 가 2 라는 것은 관찰자(viewer)의 바라보는 각도가 *y* 에서 보다 *x* 에서 2 배 넓다는 것을 의미한다. 만약 뷰포트(viewport)가 너비가 높이보다 2 배 넓게 되면, 왜곡되지 않은 이미지를 표시한다.

주어진 f 를 다음과 같이 정의하면,

$$f = tangent\left(\frac{fovy}{2}\right)$$

화면의 너비값과 높이값을 각각 *Xmax*, *Ymax* 라고 하면,

$$Ymax = 2 \times zNear \times f$$

$$Xmax = Ymax \times aspect$$

이 값을 이용하여 화면을 나타낼 절단(clipping) 평면의 픽셀 크기를 계산할 수 있다. 여기서, *zNear* 는 카메라와 화면 사이의 거리이고, *zFar* 는 사용하지 않는다.

3.3 rcCurrentPaletteMatrixOES

void rcCurrentPaletteMatrixOES(RCuint *index*);	
후속 행렬 작업에 의해 영향을 받는 팔레트(palette) 행렬을 설정한다.	
index	팔레트(palette) 행렬에 대한 인덱스

오류 코드

RC_INVALID_VALUE : *index* 가 0 과 RC_MAX_PALETTE_MATRICES_OES – 1 사이가 아닌 경우

관련 함수

rcLoadPaletteFromModelViewMatrixOES, rcMatrixIndexPointerOES, rcMatrixMode, rcWeightPointerOES

설명

현재 행렬 모드가 RC_MATRIX_PALETTE_OES 인 경우 사용할 수 있다.

3.4 rcLoadPaletteFromModelViewMatrixOES

void rcLoadPaletteFromModelViewMatrixOES(void);

현재 팔레트(palette) 행렬에 현재 모델뷰 행렬을 복사한다.

관련 함수

rcCurrentPaletteMatrixOES, rcMatrixIndexPointerOES, rcMatrixMode, rcWeightPointerOES

설명

현재 팔레트(palette) 행렬은 rcCurrentPaletteMatrixOES 로 설정된다.

3.5 rcMatrixIndexPointerOES

void rcMatrixIndexPointerOES(RCint *size*, RCenum *type*, RCsizei *stride*, const RCvoid **pointer*);

렌더링에 사용할 행렬 인덱스 배열의 데이터 및 메모리 주소를 설정한다.

size	정점 당 행렬 인덱스들의 수 (반드시 RC_MAX_VERTEX_UNITS_OES 보다 작거나 같아야 한다.)
type	배열에서 각 행렬 인덱스의 데이터 유형 (RC_UNSIGNED_BYTE, RC_UNSIGNED_INT)
stride	연속적인 행렬 인덱스 사이의 바이트 단위 간격 (만약 값이 0 이면, 행렬 인덱스들이 배열 내에 빈 공간 없이 적재되어 있음을 의미한다.)
pointer	배열에서 첫 번째 정점의 첫 번째 행렬 인덱스에 대한 포인터

오류 코드

RC_INVALID_ENUM : *type* 이 허용 값이 아닌 경우
RC_INVALID_VALUE
: *size* 가 0 보다 크지 않거나 RC_MAX_VERTEX_UNITS_OES 보다 큰 경우
: *stride* 가 0 보다 작은 경우

관련 함수

rcCurrentPaletteMatrixOES, rcDrawArrays, rcDrawElements, rcLoadPaletteFromModelViewMatrixOES, rcMatrixMode, rcWeightPointerOES

설명

행렬 인덱스는 주어진 정점에 해당하는 행렬을 혼합하는데 사용된다.

행렬 인덱스 배열이 활성화되면, rcDrawArrays 또는 rcDrawElements 가 호출될 때 이 배열이 사용된다. (rcEnableClientState 및 rcDisableClientState 참조)

3.6 rcWeightPointerOES

void rcWeightPointerOES(RCint *size*, RCenum *type*, RCsizei *stride*, const RCvoid **pointer*);

렌더링에 사용할 가중치(weight) 배열의 데이터 및 메모리 주소를 설정한다.

size	정점 당 가중치의 수 (RC_MAX_VERTEX_UNITS_OES 보다 작거나 같아야 한다.)
type	배열 내 각 가중치의 데이터 유형 (RC_FIXED, RC_FLOAT)
stride	연속적인 가중치들 사이의 바이트 단위 간격 (만약 값이 0 이면, 가중치들이 배열 내에 빈 공간 없이 적재되어 있음을 의미한다.)
pointer	배열에서 첫 번째 정점의 첫 번째 가중치에 대한 포인터

오류 코드

RC_INVALID_ENUM : *type* 이 허용 값이 아닌 경우
RC_INVALID_VALUE
: *size* 가 0 보다 크지 않거나 RC_MAX_VERTEX_UNITS_OES 보다 큰 경우
: *stride* 가 0 보다 작은 경우

관련 함수

rcCurrentPaletteMatrixOES, rcDrawArrays, rcDrawElements, rcLoadPaletteFromModelViewMatrixOES, rcMatrixIndexPointerOES, rcMatrixMode

설명

가중치는 주어진 정점에 해당하는 행렬을 혼합하는데 사용된다.

가중치 배열이 활성화되면, rcDrawArrays 또는 rcDrawElements 가 호출될 때 이 배열이 사용된다. (rcEnableClientState 및 rcDisableClientState 참조)

3.7 rcSceneAllInit

void rcSceneAllInit(void);
정적 및 동적 장면(scene) 데이터를 모두 초기화한다.

관련 함수
rcStaticSceneBegin, rcStaticSceneEnd, rcFinish

설명

rcSceneAllInit 은 rcStaticSceneBegin 와 rcStaticSceneEnd 사이에서 primitive 를 규정하는 정적 장면(static scene)을 포함하여 모든 규정되는 primitive 의 렌더링 데이터를 초기화한다.

모든 장면(scene)을 다시 그려야 하는 경우, rcSceneAllInit 를 호출한다.

3.8 rcStaticSceneBegin, rcStaticSceneEnd

void rcStaticSceneBegin(void);
void rcStaticSceneEnd(void);

정적 장면(static scene)을 위한 primitive 또는 같은 primitive 그룹의 정점들, 물성들, 텍스쳐들을 규정한다.

관련 함수

rcVertexPointer, rcTexCoordPointer, rcNormalPointer, rcDrawArrays, rcDrswElements, rcBindMaterial, rcMaterial, rcBindTexture, rcFinish

주의

rcStaticSceneBegin 이 호출된 후 rcStaticScenenEnd 는 반드시 호출되어야 한다. rcSceneAllInit 또는 rcStaticSceneBegin 가 다시 호출되기 전까지 정적 장면(static scene)은 한번만 규정한 후에는 다시 규정하지 않아도 계속 렌더링에 반영된다.

설명

rcStaticSceneBegin 과 rcStaticSceneEnd 는 primitive 또는 같은 primitive 그룹을 정의하는 정점들을 규정한다.

rcStaticSceneBegin 과 rcStaticSceneEnd 사이에서는 RC 명령의 하위 집합만을 사용할 수 있다. 그 명령은 rcVertexPointer, rcTexCoordPointer, rcNormalPointer, rcDrawArrays, rcDrswElements, rcBindMaterial, rcMaterial, rcBindTexture 이다.

rcStaticSceneBegin 과 rcStaticSceneEnd 사이에서 정의할 수 있는 정점들의 개수에 대한 제한은 없다. 불완전하게 명시된 삼각형 및 사각형은 그려지지 않는다.

각 primitive 에 대한 정점들의 최소 사양은 다음과 같다.

- 삼각형에 대해서는 정점이 3 개이며, 사각형에 대해서는 정점이 4 개이다.
- 정점들의 특정 개수가 요구되는 모드는 RC_TRIANGLES(3), RC_TRIANGLE_STRIP(3), RC_TRIANGLE_FAN(3), RC_QUADS(4)이다.

3.9 rcTextureAlpha

void rcTextureAlpha(RCbyte *value*);	
RGB 데이터 형식을 가진 텍스쳐의 알파 채널에 대한 알파값을 설정한다.	
value	물성 텍스쳐가 알파값을 가지고 있지 않은 경우에 사용되는 추가 알파값 (초기값 16)

관련 함수

rcBindMaterial, rcMaterial, rcBindTexture, rcTexImage2D

설명

rcTextureAlpha 는 rcTexImage2D 로 알파값이 없는 RGB 형식의 텍스쳐에 추가 알파값을 설정하는 데 사용되는 알파값을 설정한다. rcTextureAlpha 에 의해 설정되는 값의 범위는 0 에서 255 이다.

3.10 rcDepthBounce

void rcDepthBounce(RCuint *value*);	
레이 바운스(ray bounce)의 깊이(depth)를 설정한다.	
value	레이트레이싱에서 광선이 생성되어질 때 사용되는 레이 바운스(ray bounce)의 깊이(depth) (초기값 10)

.관련 함수

rcFinish

설명

rcDepthBounce 는 레이트레이싱 렌더링 과정에서 생성되는 광선의 최대 바운스 깊이(bounce depth)를 설정한다. rcDepthBounce 에 의해 설정되는 값의 범위는 0 에서 14 이다.