

# **RayStorm Documentation**

Andreas Heumann

**COLLABORATORS**

	<i>TITLE :</i> RayStorm Documentation		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Andreas Heumann	January 7, 2023	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>RayStorm Documentation</b>	<b>1</b>
1.1	indexnode . . . . .	1
1.2	RayStorm Documentation . . . . .	4
1.3	Introduction . . . . .	5
1.4	General . . . . .	6
1.5	Octree . . . . .	6
1.6	Antialiasing . . . . .	7
1.7	Depth of field . . . . .	8
1.8	Soft shadows . . . . .	8
1.9	Surfaces . . . . .	8
1.10	Internals . . . . .	10
1.11	Virtual Memory . . . . .	10
1.12	Blur . . . . .	10
1.13	Requirements . . . . .	11
1.14	Features . . . . .	11
1.15	Installation . . . . .	12
1.16	ARexx Interface . . . . .	12
1.17	General ARexx-commands . . . . .	15
1.18	antialias . . . . .	17
1.19	brushpath . . . . .	17
1.20	cleanup . . . . .	17
1.21	display . . . . .	17
1.22	distrib . . . . .	18
1.23	geterrorstr . . . . .	18
1.24	objectpath . . . . .	18
1.25	pointlight . . . . .	19
1.26	quit . . . . .	19
1.27	savepic . . . . .	19
1.28	setcamera . . . . .	20
1.29	setscreen . . . . .	20

---

---

1.30	setworld	20
1.31	spotlight	21
1.32	startrender	22
1.33	texturepath	22
1.34	wintofront	22
1.35	ARexx-commands for creating objects	23
1.36	loadobj	23
1.37	plane	23
1.38	sphere	24
1.39	triangle	24
1.40	ARexx-commands for setting attributes	25
1.41	ambient	26
1.42	brush	26
1.43	difftrans	28
1.44	diffuse	28
1.45	foglen	28
1.46	imtexture	29
1.47	newsurface	29
1.48	refexp	29
1.49	reflect	30
1.50	refrindex	30
1.51	spectrans	31
1.52	specular	31
1.53	transexp	32
1.54	transluc	32
1.55	transpar	32
1.56	ARexx-commands for animation control	33
1.57	alignment	33
1.58	newactor	33
1.59	position	34
1.60	size	34
1.61	ARexx-errors	35
1.62	Examples	37
1.63	Tutorials	39
1.64	Simple scene	39
1.65	Bouncing ball	41
1.66	Tips&Tricks	45
1.67	Textures	46
1.68	Bump	46

---

---

1.69	Checker	47
1.70	Linear	47
1.71	Wood	47
1.72	Marble	48
1.73	Radial	49
1.74	Stars	49
1.75	Known Bugs	49
1.76	Legal Stuff	50
1.77	Credits	50
1.78	Register	50
1.79	Author	51
1.80	History of Changes	51
1.81	PC-version	53
1.82	Homepage	53
1.83	Future	53

---

## Chapter 1

# RayStorm Documentation

### 1.1 indexnode

-A-

ALIGNMENT

Altitude brush

AMBIENT

ANTIALIAS

Antialiasing

Author

-B-

Backdrop

BRUSH

BRUSHPATH

Bump

-C-

Checker

CLEANUP

Color brush

Credits

Cylinder mapping

-D-

Depth of Field

DIFFTRANS

---

DIFFUSE

DISPLAY

DISTRIB

-E-

Errors

Examples

-F-

Features

Filter brush

Flat mapping

FOGLEN

Future

-G-

GETERRORSTR

-H-

History

Homepage

-I-

IMTEXTURE

Installation

Internals

Introduction

-L-

Legal Stuff

Linear

LOADOBJ

-M-

Marble

Motion Blur

-N-

NEWACTOR

NEWSURFACE

-O-

OBJECTPATH

-P-

PC-Version

PLANE

POINTLIGHT

POSITION

-Q-

QUIT

-R-

Radial

REFEXP

REFLECT

Reflect brush

Reflection map

REFRINDEX

Register

Requirements

-S-

SAVEPIC

SETCAMERA

SETSCREEN

SETWORLD

SIZE

Soft shadows

SPECTRANS

SPECULAR

Specular brush

SPHERE

Sphere mapping

SPOTLIGHT



Stars

STARTRENDER

Surface

-T-

TEXTUREPATH

Textures

Tips&Tricks

TRANSEXP

TRANSLUC

TRANSPAR

TRIANGLE

Tutorials

-V-

Virtual Memory

-W-

WINTOFRONT

Wood

WWW

## 1.2 RayStorm Documentation

29 ←  
February ←  
1996

RayStorm  
v1.2  
Demoverision  
by Andreas Heumann and Mike Hesser

Introduction  
What is RayStorm?

Requirements  
What do I need to run it?

Features  
What can RayStorm do?

Installation

---

How can I install it?

ARexx interface  
Which commands doe it have?

Examples  
How do I use the examples?

Tutorials  
Some tutorials

Tips&Tricks  
Useful tips,tricks and hints

Textures  
How do I use textures?

Known bugs  
Bugs

Legal Stuff  
Legal stuff

Register  
What must I do to register?

Credits  
Thanks go to...

Authors  
Who had written it?

PC-Version  
Where can I get the PC-version?

Homepage  
Where to find us on the World Wide Web

History  
What happened in the past?

Future  
What is planned for the future?

## 1.3 Introduction

### INTRODUCTION

RayStorm has been written to be as fast as possible, and use as less memory as possible. Thus we have implemented a octree algorithm, and optimized all calculations as much as we could.

Originally, RayStorm has been developed on Amiga using Maxon C++ 3.0 Developer.  
The

---

PC version  
has been compiled with WATCOM C++ 10.5.

This demo version is limited to a resolution of 160x128.

The full version is unlimited. Use the registration program to register.

#### FUNDAMENTALS ABOUT RAYTRACING

General

Octree

Antialiasing

Depth of field

Soft shadows

Surfaces

Internals

Virtual Memory

Motion Blur

## 1.4 General

### GENERAL

Raytracing makes it possible to generate fotorealistic pictures of objects.

A raytracer casts a ray form the position of the viewer through a scene and calculates the intersections with the objects in that scene. If a intersection is found, the raytracer decides which color the object at this position has. If the object is reflective or transparent, the raytracer casts new rays from this positon and tests the intersections again and so on.

To make the surfaces of the objects more realistic, textures which simulate marble or clouds or water or other surfaces can be used.

## 1.5 Octree

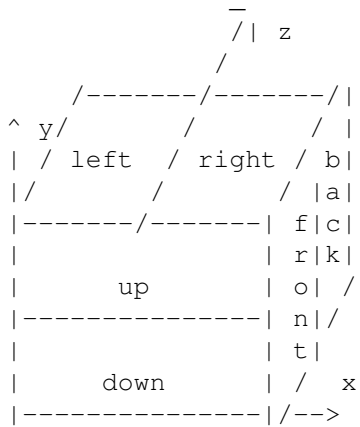
### OCTREE

Simple raytracers determine the intersections with objects by testing all objects. This can lead to long rendering times if there are a lot of objects in the scene.

---

One solution of this problem is the Octree algorithm. This algorithm divides the scene in eight child-cells and every child-cell again in eight cells and so on until there are less than one objects in the cell or the maximum depth of the tree is reached.

Division of space with the octree algorithm:



## 1.6 Antialiasing

### ANTIALIASING

RayStorm uses a algorithm called 'Adaptive Supersampling' to do antialiasing. This algorithm cast for each pixel with a high contrast against it's four neighbours new rays which are close to the ray used for the pixel itself. The new color of the pixel is calculated with the supersampled pixels and the gaussian filter.

Supersampling is also used to do

depth of field

and

soft shadows

. So if

you want to use this features you have to set a antialiasing value greater than one. (->

ANTIALIAS

)

Example:

Settings: squareroot of number of samples per pixel: 3

```

|-----|-----|-----|
| super- | super- | super- |
| sampled| sampled| sampled|
|         |         |         |
|-----|-----|-----|
| super- |         | super- |
| sampled| pixel  | sampled|
|         |         |         |
|-----|-----|-----|
| super- | super- | super- |
| sampled| sampled| sampled|

```

```
|          |          |          |
|-----|-----|-----|

|- Gaussian filter width -|
```

The rendering time increases dramatically if you use antialiasing. The values below depend on the contrast of the picture.

Samplesetting	rendering time
1	x1
2	x4
3	x9
4	x16
...	...

Setting higher than 3 are not leading to significant better results.

## 1.7 Depth of field

### DEPTH OF FIELD

Objects in computer graphics are normally rendered in an image plane using a pinhole camera model. That is to say, no matter how far or how near the objects are from the camera, they are always in sharp focus. Depth of field means that only objects at a certain distance from the camera lens are in sharp focus. Further and nearer objects produce a blurred image on the film plane.

[From 'Advanced Animation and Rendering Techniques']

To use depth of field you have to set  
 ANTIALIAS  
 to a value bigger  
 than one.

Example for DOF

## 1.8 Soft shadows

### SOFT SHADOWS

Real Light sources never have a zero size, therefore the shadows behind objects are never hard edged, they are soft. RayStorm generates this shadows by jittering the position of the light source. To use soft shadows you have to set

DISTRIB  
 to a value bigger than one.

## 1.9 Surfaces

---

## SURFACES

Ambient (set with AMBIENT)

This determines the color of the object in sections which are in shadow.

Diffuse reflection (set with DIFFUSE)

The diffuse reflection falls off as the cosine of the angle between the normal and the ray to the light. Diffuse reflection determines the main color of the object (color in Imagine).

Specularly reflected highlights (set with SPECULAR)

Specularly reflected highlights fall off as the cosine of the angle between the reflected ray and the ray to the light source (specular in Imagine)

Specular reflection exponent (set with REFEXP)

Determines the size of the specularly reflected highlights, the higher the smaller the highlight (hardness in Imagine)

Diffuse transmission (set with DIFFTRANS)

Same as diffuse reflection, but only used if the lightsource is on opposite side of surface. Only applied if tranlucency is not 0.

Specular transmission (set with SPECTRANS)

Same as specular reflection, but only used if the lightsource is on opposite side of surface. Only applied if tranlucency is not 0.

Specular transmission exponent (set with TRANSEXP)

Same as specular reflection exponent, but only used if the lightsource is on opposite side of surface.

Specular transmittance (set with TRANSLUC)

Specular transmittance.

Transparency (set with TRANSPAR)

Transparent color (filter in Imagine).

Reflectivity (set with REFLECT)

Reflective color (reflect in Imagine).

Fog lenght (set with FOGLEN) (fog in Imagine).

Index of refraction (set with REFRINDEX)

Determines how the ray through transparent objects is refracted, the higher the more (index of refraction in Imagine).

Is calculated with the formula

$$\text{index} = \frac{\text{lightspeed in vacuum}}{\text{lightspeed in object}} .$$

## 1.10 Internals

### INTERNALS

Memory requirements

Triangle:	156 Bytes (flat shaded)
	192 Bytes (Phong shaded)
Sphere:	70 Bytes
Plane:	78 Bytes
Surface:	122 Bytes + length of name
Screenbuffer:	4 Bytes per pixel

Memory requirements of the octree depends on the scene.

## 1.11 Virtual Memory

### VIRTUAL MEMORY

RayStorm has been tested succesfully with VMM 3.1 from Martin Apel. If you want use RayStorm with virtual memory notice follwing hints:

- set Minimum VM allocation to 100 bytes if you define large scenes with many objects, because RayStorm only allocates small pieces of memory for single objects (less then 200 bytes). If you're loading Imagine objects RayStorm allocates big blocks of memory so you don't have to set Minimum VM allocation to 100.
- use a partition or a pseudo-partition for VMM, this is faster

## 1.12 Blur

### MOTION BLUR

Motion blur is temporal anti-aliasing. In animated sequences, the normal rendering process functions like a camera that possesses an infinitely short exposure time and this eliminates the blurring of the image due to relative motion between an object and the film plane. When a series of images, generated without motion blur, is displayed as an animated sequence, the illusion of smooth motion is diminished by strobing effects. As human beings we expect to see loss of detail in moving images.

Motion blur is accounted for in distributed ray tracing by extending the distributed sampling and jittering into the time domain and computing a

solution that extracts information from the scen over the duration of the shutter exposure time. Objects are moved as required in the time period and visibility consequently changes over this time intervall. This method ensures that highlights and shadows are blurred or not, depending on the nature of the motion.

[From 'Advanced Animation and Rendering Techniques']

## 1.13 Requirements

REQUIREMENTS

REQUIREMENTS

- (1) You will need at least Kickstart 2.0.
- (2) 881-version: 68020 processor and a mathematical coprocessor (68881/882 or internal 68040/060 version).
- (2) 020-version: 68020 processor (no math coprocessor needed)
- (4) 000-version: 68000 processor (should even run on a Amiga 500 (not tested))
- (5) 1MB RAM minimum
- (6) RayStorm was written using MUI. So you need muimaster.library V2.3 or better to run RayStorm.

recommended: 68030, 68882, Harddisk, GFX-Board

The faster the better :-).

Tested with:

A1200 68030/50, 6MB, 200MB HD  
A2000 68040/30, 9MB, 250+250MB HD, Merlin Gfx-board  
A2000 68030/14, 68882/20, 4MB, 730+52MB HD  
A4000 68030/25, 68882/57, 4MB, 730+80MB HD

## 1.14 Features

FEATURES

- Up to 30% faster than Imagine (in trace mode).
- ARexx-port. RayStorm can be used by all programs with ARexx-port.
- Imagine compatible. RayStorm is designed to be almost compatible to Imagine. It can load Imagine objects and use Imagine textures.
- Octree algorithm used for rendering.
- Color, reflectivity, filter, altitude and specular mapping.
- Flat, cylinder and sphere mapping.
- Soft brush mapping.
- Mathematical textures: wood, marble, bumps, checker, linear, radial,



- stars.
- Transparency and physically correct refractions.
- 8 levels of antialiasing (adaptive supersampling).
- Rendering box.
- Three builtin object types: sphere, plane and triangle.
- Three light types: ambient, point and spot.
- Depth of field with adjustable focal distance and aperture.
- Soft shadows.
- Backdrop picture.
- Global fog and foggy objects.
- Material attributes for realistic objects: ambient color, diffuse color, specular color, specular reflection exponent, diffuse transmission color, specular transmission color, specular transmission exponent, specular transmittance, transparent color, reflective color, index of refraction, foglength.
- Bright objects.
- Motion blur for realistic animations.
- Quick rendering.
- Global reflection map.
- Image formats: IFF-ILBM, PNG, TGA.
- Object format: Imagine-TDDD, Autodesk 3DS
- New image- and object-formats can be easily included because of the modular concept.

## 1.15 Installation

### INSTALLATION

There is a installation script included in the archive which uses the Commodore Installer. Run it to install RayStorm.

## 1.16 ARexx Interface

### AREXX INTERFACE

#### Introduction

RayStorm is completely controlled through it's ARexx interface. We recommend that you have a look at the example script files in the 'ARexx' directory. These examples cover most of the features of RayStorm. We encourage you to create your own files and make them available for the public. You can send them to us and we might add them as an example files in the next version of RayStorm or we include them to our Homepage.

In one of the next versions of RayStorm we'll create a more powerful language, which has a similar syntax to C++.

It's the same if you write the the commands in upper case or lower case. But it's important to enclose all commans in quotes because ARexx tries to interpret the line before it sends it to ARexx. It may happen that the line is changed and RayStorm don't do this what you want.

A typical structure of a scene file is:

---

```
/* title, comments, ... */

/* setting resolution, world, camera, lightsources */
'SETSCREEN 160 128'
'SETWORLD [0,0,0] [40,40,40]'
'SETCAMERA <0,0,80> <0,0,0> <0,1,0> 25 20'
'POINTLIGHT <10,-10,100> [255,255,255] SHADOW'

/* define surfaces and actors */
'NEWSURFACE TEST1'
'AMBIENT [255,0,0]'
'DIFFUSE [255,0,0]'
'SPECULAR [255,255,255]'

'NEWSURFACE TEST2'
'AMBIENT [0,0,255]'

/* creating objects */
'SPHERE TEST1 <0,0,0> 10'
'SPHERE TEST2 <0,0,0> 10'

/* finally start to render the scene */
'STARTRENDER'

/* save the image */
'SAVEPIC "test.iff"'

'CLEANUP'
```

The parameters of a command can be FLOATs, INTEGERs, VECTORs, COLORs, STRINGs, and IDENTIFIERS.

**FLOAT** An FLOAT is a floating point number with single precision

**NUMBER** A NUMBER is a simple integer number

**VECTOR** A VECTOR is embedded in '< ' >' and consists of three FLOATs

**COLOR** A COLOR is embedded in '[' ']' and consists of three INTEGERs with a range of 0 to 255

**STRING** A STRING consists of characters

**KEYWORD** An KEYWORD is a switch and consists of uppercase characters

Address

The ARexx-address of RayStorm is 'RAYSTORM'.

Parameter conventions:

- /S - Switch.
- /N - Number.
- /A - Required.

All other numeric parameters are floating point numbers.

---

ARexx commands

General

Objects

Attributes

Animation

Errors

Alphabetically sorted

-A-

ALIGNMENT

AMBIENT

ANTIALIAS

-B-

BRUSH

BRUSHPATH

-C-

CLEANUP

-D-

DIFFTRANS

DIFFUSE

DISPLAY

DISTRIB

-F-

FOGLEN

-G-

GETERRORSTR

-I-

IMTEXTURE

-L-

LOADOBJ

-N-

NEWSURFACE

-O-

OBJECTPATH

-P-

PLANE  
POINTLIGHT  
POSITION  
-Q-  
QUIT  
-R-  
REFEXP  
REFLECT  
REFRINDEX  
-S-  
SAVEPIC  
SETCAMERA  
SETSCREEN  
SETWORLD  
SIZE  
SPECTRANS  
SPECULAR  
SPHERE  
SPOTLIGHT  
STARTRENDER  
-T-  
TEXTUREPATH  
TRANSEXP  
TRANSLUC  
TRANSPAR  
TRIANGLE  
-W-  
WINTOFRONT

## 1.17 General ARexx-commands

---

## GENERAL AREXX-COMMANDS

## ANTIALIAS

sets antialiasing parameters

## BRUSHPATH

sets brush path

## CLEANUP

cleanups scene

## DISPLAY

displays scene

## DISTRIB

sets parameters for distributive sampling

## GETERRORSTR

gets a error string for a given number

## OBJECTPATH

sets object path

## POINTLIGHT

creates point lightsource

## QUIT

quits RayStorm

## SAVEPIC

saves rendered picture

## SETCAMERA

sets camera parameters

## SETSCREEN

sets screen parameters

## SETWORLD

sets world parameters

## SPOTLIGHT

creates spot lightsource

## STARTRENDER

starts rendering

## TEXTUREPATH

sets texture path

## WINTOFRONT

brings window to front

## 1.18 antialias

ANTIALIAS

Template:

SAMPLES/N/A,WIDTH,CONTRIB

Arguments:

NUMBER SAMPLES

    squareroot of number of samples per pixel (max. 8)

FLOAT WIDTH

    width of gaussian filter

COLOR CONTRIB

    max. allowed contrast

Description:

Sets antialiasing parameters (see  
    Antialiasing  
    )

Default:

ANTIALIAS 1 1.3 [51,38,76]

## 1.19 brushpath

BRUSHPATH

Template:

PATH/A

Arguments:

STRING PATH

    pathname (format: 'path1;path2;...;pathn')

Description:

Defines the path where to search brushes.

## 1.20 cleanup

CLEANUP

Template:

none

Arguments:

none

Description:

Deletes all defined objects, lightsources, surfaces and actors

## 1.21 display

DISPLAY

!!! CAUTION !!!

THIS COMMAND ISN'T RELEASED IN THIS VERSION YET

---

!!! CAUTION !!!

Template:  
FLOYD/S  
Arguments:  
KEYWORD FLOYD/S  
dither with Floyd-Steinberg algorithm  
Description:  
Displays rendered pic on screen

## 1.22 distrib

DISTRIB

Template:  
SAMPLES/N,SOFTSHADOW/N  
Arguments:  
NUMBER SAMPLES/N  
squareroot of number of samples per pixel for motionblur  
NUMBER SOFTSHADOW/N  
squareroot of number of samples per pixel for softshadows  
Description:  
Sets number of samples per pixel for distributive sampling (used for  
and )  
Default:  
DISTRIB 1,1

## 1.23 geterrorstr

GETERRORSTR

Template:  
ERRNUM/N/A  
Arguments:  
NUMBER ERRNUM  
error number  
Description:  
Returns the error string for the given error number

## 1.24 objectpath

OBJECTPATH

Template:  
PATH/A  
Arguments:  
PATH  
pathname (format: 'path1;path2;...;pathn')  
Description:  
Defines the path where to search Imagine objects.

---

## 1.25 pointlight

### POINTLIGHT

Template:

POS, COLOR, SIZE, SHADOW/S, ACTOR, FALLOFF

Arguments:

VECTOR POS

position

COLOR COLOR

color of light

VECTOR SIZE

size of light source (used for  
soft shadows  
)

KEYWORD SHADOW/S

cast shadows if keyword given

STRING ACTOR

name of actor

FLOAT FALLOFF

distance where the brightness of the light is zero

Description:

Creates a point lightsource

Default:

POINTLIGHT <0,0,0> [255,255,255] 0 ?? ?? 0

## 1.26 quit

### QUIT

Template:

none

Arguments:

none

Description:

Quits Raystorm

## 1.27 savepic

### SAVEPIC

Template:

NAME/A, FORMAT

Arguments:

STRING NAME

name of file to save

STRING FORMAT

image format (default ILBM)

Description:

Saves rendered picture. If an error occurs the command returns an error string.

---



## 1.28 setcamera

SETCAMERA

Template:

POS/A, VIEWPOINT, VIEWUP, FOVX, FOVY, FOCALDIST, APERTURE, POSACTOR, VIEWACTOR

Arguments:

VECTOR POS

position

VECTOR VIEWPOINT

viewpoint

VECTOR VIEWUP

viewupvector

FLOAT FOVX, FOVY

field of view (in degree) (20 degree creates camera like Imagine default camera)

FLOAT FOCALDIST

distance from eye to focal plane

FLOAT APERTURE

aperture width (0 = pinhole) (-> depth of field)

STRING POSACTOR

name of position actor

STRING VIEWACTOR

name of point of view actor

Description:

Sets the parameters of the camera

Default:

SETCAMERA <0,0,-10> <0,0,0> <0,1,0> 45 45 1. 0.

## 1.29 setscreen

SETSCREEN

Template:

RESX/N/A, RESY/N/A, COLORS/N

Arguments:

NUMBER RESX, RESY

resolution

NUMBER COLORS

number of colors (not yet implemented)

Description:

Sets the screen parameters. Note that in the demo-version the resolution is limited to 160x128!

Default:

SETSCREEN 128 128 32

## 1.30 setworld

SETWORLD

---

Template:  
 BACK/A, AMBIENT, RANDJIT/S, BACKDROP, FOGLEN, FOGHEIGHT, FOGCOLOR, REFLMAP

Arguments:  
 COLOR BACK  
     backgroundcolor  
 COLOR AMBIENT  
     ambientcolor  
 KEYWORD RANDJIT  
     use random jitter for  
         depth of field  
         and  
         soft shadows  
         STRING BACKDROP  
         name of backdrop picture  
 FLOAT FOGLEN  
     global fog length  
 FLOAT FOGHEIGHT  
     highest fog y-coordinate  
 COLOR FOGCOLOR  
     fogcolor  
 STRING REFLMAP  
     name of reflection map

Description:  
 Sets world parameters

Default:  
 SETWORLD [0,0,0] [0,0,0] ?? 32 0 [255,255,255]

## 1.31 spotlight

### SPOTLIGHT

Template:  
 POS, COLOR, LOOKPOINT, ANGLE, SIZE, SHADOW/S, ACTOR, LOOKP\_ACTOR, FALLOFF

Arguments:  
 VECTOR POS  
     position  
 COLOR COLOR  
     color  
 VECTOR LOOKPOINT  
     lookpoint  
 FLOAT ANGLE  
     opening angel (in degree max. 180)  
 FLOAT SIZE  
     size of light source (used for  
         soft shadows  
     )  
 KEYWORD SHADOW  
     cast shadows if keyword given  
 STRING ACTOR  
     name of position actor  
 STRING LOOKP\_ACTOR  
     name of lookpoint actor  
 FLOAT FALLOFF  
     distance where the brightness of the light is zero

Description:

---

Creates a spot lightsource. The rays emitted from a spotlight are constrained by a cone. The LOOKPOINT vector gives the center of the illuminated area.

Default:

```
SPOTLIGHT <0,0,0> [255,255,255] <0,0,1> 45 0 ?? ?? ?? 0
```

## 1.32 startrender

STARTRENDER

Template:

```
QUICK/S,DEPTH/N,FROM,TO/N,LEFT/N,TOP/N,RIGHT/N,BOTTOM/N
```

Arguments:

KEYWORD QUICK

render quick (no shadows, reflections and transparency)

NUMBER DEPTH

depth of generated octree (default 3)

FLOAT FROM,TO

time code (default 0,0). If you want motion blur you have to

set FROM and TO to different values, if not only set FROM.

NUMBER LEFT, TOP, RIGHT, BOTTOM

coordinates for rendering box. Picture is rendered only inside of rectangle.

Description:

Starts rendering process. If you set QUICK shadows, reflections and transparency are not calculated. In very complex scenes it is useful to increase the octree depth in order to reach a better performance during the rendering process. But this can only be done with enough memory!

## 1.33 texturepath

TEXTUREPATH

Template:

```
PATH/A
```

Arguments:

PATH

pathname (format: 'path1;path2;...;pathn')

Description:

Defines the path where to search textures.

## 1.34 wintofront

WINTOFRONT

Template:

```
none
```

Arguments:

---

none  
Description:  
Brings RayStorm window in front

## 1.35 ARexx-commands for creating objects

### AREXX-COMMANDS FOR CREATING OBJECTS

LOADOBJ  
loads an Imagine TDDD-file

PLANE  
creates a plane (ground in Imagine)

SPHERE  
creates a sphere

TRIANGLE  
creates a triangle

## 1.36 loadobj

LOADOBJ

Template:

NAME/A, POS, ALIGN, SCALE, ACTOR

Arguments:

STRING NAME

filename

VECTOR POS

position

VECTOR ALIGN

alignment (in degrees)

VECTOR SCALE

scaling

STRING ACTOR

name of actor

Description:

Loads an Imagine TDDD-file object with attributes, brushes and textures.

Where to get Imagine object files?

Look on FTP-servers which support AMINET. For example try out

ftp.uni-paderborn.de

Path: ftp/aminet/pub/gfx/3dobj/

Default:

LOADOBJ ??? <0,0,0> <0,0,0> <1,1,1>

## 1.37 plane

## PLANE

### Template:

SURF/A, POS, NORM, ACTOR

### Arguments:

STRING SURF  
    surface name  
VECTOR POS  
    position  
VECTOR NORM  
    normal  
STRING ACTOR  
    name of actor

### Description:

Creates a infinite plane

### Default:

PLANE ??? <0,0,0> <0,1,0>

## 1.38 sphere

## SPHERE

### Template:

SURF/A, POS/A, RADIUS/A, ACTOR

### Arguments:

STRING SURF  
    surface name  
VECTOR POS  
    position  
FLOAT RADIUS  
    radius  
STRING ACTOR  
    name of actor

### Description:

Creates a sphere

## 1.39 triangle

## TRIANGLE

### Template:

SURF/A, P1/A, P2/A, P3/A, N1, N2, N3, ACTOR

### Arguments:

STRING SURF  
    surface name  
VECTOR P1  
    first point  
VECTOR P2  
    second point  
VECTOR P3  
    third point

```
VECTOR N1
  first normal
VECTOR N2
  second normal
VECTOR N3
  third normal
STRING ACTOR
  name of actor
```

Description:

Creates a triangle with corners at position P1, P2 and P3. If you specify the normals, a phong shaded triangle otherwise a flat triangle is created. Computing the normals by hand is a difficult task, and should be done by utility programs.

## 1.40 ARexx-commands for setting attributes

### AREXX-COMMANDS FOR SETTING ATTRIBUTES

Every object must have a surface definition. With the following commands you can set the attributes of a surface. First you have to define the current surface with 'NEWSURFACE <name>'. Raystorm will set the attributes of the new surface to default values. Every following command such as AMBIENT or DIFFTRANS refers to the current surface and will override the corresponding default values.

The following examples define two surfaces:

```
NEWSURFACE RED
AMBIENT [255,0,0]
DIFFUSE [255,0,0]
```

```
NEWSURFACE WATER
DIFFUSE [0,0,255]
REFRINDEX 1.333
```

List of surface commands:

```
NEWSURFACE
  creates a new surface

AMBIENT
  sets ambient color

BRUSH
  adds a brush

DIFFTRANS
  sets diffuse transmission color

DIFFUSE
  sets diffuse color

FOGLEN
  sets the foglength
```

IMTEXTURE  
adds a Imagine texture

REFEXP  
sets the specular reflection exponent

REFLECT  
sets the specular reflectivity

REFRINDEX  
sets the index of refraction

SPECTRANS  
sets the specular transmission

SPECULAR  
sets the specular color

TRANSEXP  
sets the specular transmission exponent

TRANSLUC  
sets the specular transmittance

TRANSPAR  
sets the diffuse transmittance

## 1.41 ambient

AMBIENT

Template:

COLOR/A

Arguments:

STRING COLOR

color

Description:

Sets the ambient color of surface

Default:

AMBIENT [255,255,255]

## 1.42 brush

BRUSH

Template:

NAME/A, TYPE/A, WRAP/A, POS/A, ALIGN/A, SIZE/A, REPEAT/S, MIRROR/S, SOFT/S, ACTOR

Arguments:

STRING NAME

name of brush file

KEYWORD TYPE [COLOR|REFLECT|FILTER|ALTITUDE|SPECULAR]

Brush type

---

KEYWORD WRAP [FLAT|WRAPX|WRAPY|WRAPXY]  
 Brush wrapping method

VECTOR POS  
 position

VECTOR ALIGN  
 alignment

VECTOR SIZE  
 size of brush

KEYWORD REPEAT  
 if set brush is repeated like a tile

KEYWORD MIRROR  
 if set brush is mirrored

KEYWORD SOFT  
 if set brush color is softly interpolated

STRING ACTOR  
 name of actor

Description:

Adds a brush to surface. A brush is a bitmap which is wrapped around an object. The specified file will be searched for in the current directory. If not found and a brushpath is given, the file will be searched there. If an error occurs the command returns an error string.

Constants for type:

COLOR

Replaces the surface color of the object with the image (sets  
     DIFFUSE  
     and  
     AMBIENT  
 ).

REFLECT

Map covers the surface and reflects environment (see  
     REFLECT  
 ).

FILTER

Uses the white color to pass colors and the black area to hold back color with a variance between two colors (see  
     TRANSPAR  
 ).

ALTITUDE

The red values of the brush are used to give the surface an appearance of bumpiness.

SPECULAR

The rgb values set the specular color of the surface (see  
     SPECULAR  
 ).

Constants for wrap :

FLAT

The brush is projected to X-Y plane, the axis is in the middle of the brush area, length is the distance from the middle to the border.

WRAPX

The brush is wrapped around the x-axis, like on a cylinder. The left edge of the brush begins at the positive X axis and wraps the brush around the cylinder from 'west' to 'east'.

WRAPY

Same as WRAPX, but wrapping is around the y-axis.



#### WRAPXY

Wrapping both: around X and Y axis. It is assumed, that the object is a sphere. The Y axis is the north/south pole of the spherical mapping. The left edge of the brush begins at the positive X axis and wraps the brush around the sphere from 'west' to 'east'. The brush covers the sphere exactly once.

### 1.43 difftrans

#### DIFFTRANS

Template:

COLOR/A

Arguments:

COLOR COLOR  
color

Description:

Sets the diffuse transmission color of surface. Same as diffuse reflection, but only used if the lightsource is on opposite side of surface. Only applied if tranlucency is not set to zero.

Default:

DIFFTRANS [0,0,0]

### 1.44 diffuse

#### DIFFUSE

Template:

COLOR/A

Arguments:

COLOR COLOR  
color

Description:

Sets the diffuse color of surface. The diffuse reflection falls off as the cosine of the angle between the normal and the ray to the light. Diffuse reflection determines the main color of the object (color in Imagine).

Default:

DIFFUSE [255,255,255]

### 1.45 foglen

#### FOGLEN

Template:

VALUE/A

Arguments:

FLOAT VALUE/A  
foglength

Description:

Sets the foglength of the surface. Fog color is set with  
TRANSPAR

Default:  
FOGLEN 0

## 1.46 imtexture

IMTEXTURE

Template:

NAME/A, POS, ALIGN, SIZE, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, ←  
ACTOR

Arguments:

STRING NAME

name of Imagine texture file

VECTOR POS

position

VECTOR ALIGN

alignment

VECTOR SIZE

size of texture axis

FLOAT P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16

texture parameters

STRING ACTOR

name of actor

Description:

Adds a Imagine texture to surface

Default:

defaults are taken from texture if not all paramters are given

## 1.47 newsurface

NEWSURFACE

Template:

NAME/A, BRIGHT/S

Arguments:

STRING NAME

name

KEYWORD BRIGHT

if set the brightness of the surface is everywhere the same

Description:

Creates a new surface with name 'NAME'

## 1.48 refexp

REFEXP

Template:

---

VALUE/A  
Arguments:  
FLOAT VALUE  
specular reflection exponent  
Description:  
Sets the specular reflection exponent of surface. Determines the size of the specularly reflected highlights, the higher the smaller the highlight (hardness in Imagine).  
Default:  
REFEXP 12.

## 1.49 reflect

REFLECT  
Template:  
COLOR/A  
Arguments:  
COLOR COLOR  
color  
Description:  
Sets the specular reflectivity of surface  
Default:  
REFLECT [0,0,0]

## 1.50 refrindex

REFRINDEX  
Template:  
VALUE/A  
Arguments:  
FLOAT VALUE  
index of refraction  
Description:  
Sets the index of refraction of surface. Determines how the ray is refracted through transparent objects, the higher the more (index of refraction in Imagine).  
Default:  
REFRINDEX 1.  
Examples:

MATERIAL	Index
Vacuum .....	1.00000 (exactly)
Air (STP).....	1.00029
Acetone .....	1.36
Alcohol .....	1.329
Amorphous Selenium .....	2.92
Calspar1 .....	1.66
Calspar2 .....	1.486
Carbon Disulfide .....	1.63

---

Chromium Oxide .....	2.705
Copper Oxide .....	2.705
Crown Glass .....	1.52
Crystal .....	2.00
Diamond .....	2.417
Emerald .....	1.57
Ethyl Alcohol .....	1.36
Flourite .....	1.434
Fused Quartz .....	1.46
Heaviest Flint Glass .....	1.89
Heavy Flint Glass .....	1.65
Glass .....	1.5
Ice .....	1.309
Iodine Crystal .....	3.34
Lapis Lazuli .....	1.61
Light Flint Glass .....	1.575
Liquid Carbon Dioxide .....	1.20
Polystyrene .....	1.55
Quartz 1 .....	1.644
Quartz 2 .....	1.553
Ruby .....	1.77
Sapphire .....	1.77
Sodium Chloride (Salt) 1 ....	1.544
Sodium Chloride (Salt) 2 ....	1.644
Sugar Solution (30%) .....	1.38
Sugar Solution (80%) .....	1.49
Topaz .....	1.61
Water (20 C) .....	1.333
Zinc Crown Glass .....	1.517

## 1.51 spectrans

SPECTRANS

Template:

COLOR/A

Arguments:

COLOR COLOR  
color

Description:

Sets the specular transmission color of surface. Same as specular reflection, but only used if the lightsource is on opposite side of surface. Only applied if tranlucency is not 0.

Default:

SETSPECTRANS [255,255,255]

## 1.52 specular

SPECULAR

Template:

COLOR/A

---

## Arguments:

```
COLOR COLOR
  color
```

## Description:

Sets the specular color of surface. Specularly reflected highlights fall off as the cosine of the angle between the reflected ray and the ray to the light source (specular in Imagine).

## Default:

```
SPECULAR [255,255,255]
```

## 1.53 transexp

```
TRANSEXP
```

## Template:

```
VALUE/A
```

## Arguments:

```
FLOAT VALUE
  specular transmission exponent
```

## Description:

Sets the specular transmission exponent of surface. Same as specular reflection exponent, but only used if the lightsource is on opposite side of surface.

## Default:

```
TRANSEXP 12.
```

## 1.54 transluc

```
TRANSLUC
```

## Template:

```
VALUE/A
```

## Arguments:

```
FLOAT VALUE
  specular transmittance
```

## Description:

Sets the specular transmittance of surface

## Default:

```
TRANSLUC 0
```

## 1.55 transpar

```
TRANSPAR
```

## Template:

```
COLOR/A
```

## Arguments:

```
COLOR COLOR
  color
```

## Description:

Sets the diffuse transmittance of surface  
Default:  
TRANSPAR [0,0,0]

## 1.56 ARexx-commands for animation control

### AREXX-COMMANDS FOR ANIMATION CONTROL

```
ALIGNMENT
    sets alignment

NEWACTOR
    creates a new actor

POSITION
    sets position

SIZE
    sets size
```

## 1.57 alignment

ALIGNMENT

Template:

```
FROM/A, TO/A, ALIGN/A, TYPE
```

Arguments:

```
FLOAT FROM, TO
```

time code

```
VECTOR ALIGN
```

alignment at time 'TO'

```
KEYWORD TYPE [LINEAR]
```

interpolation type (currently only linear)

Description:

Sets the alignment of the object. 'TYPE' can be one of the following identifiers:

LINEAR the interpolation is done in a straight way.

SPLINE the interpolation is done in a spline curve way. (NOT IMPLEMENTED YET)

## 1.58 newactor

NEWACTOR

Template:

```
NAME/A, POS, ALIGN, SIZE
```

Arguments:

```
STRING NAME
```

name of new actor

```

VECTOR POS
  axis position
VECTOR ALIGN
  axis alignment (in degrees)
VECTOR SIZE
  axis size
Description:
  Creates a new actor
Default:
  NEWACTOR ??? <0,0,0> <0,0,0> <1,1,1>

```

## 1.59 position

POSITION

Template:

```
FROM/A, TO/A, POS/A, TYPE
```

Arguments:

```
FLOAT FROM, TO
```

```
  time code
```

```
VECTOR POS
```

```
  position at time 'TO'
```

```
KEYWORD TYPE [LINEAR]
```

```
  interpolation type (currently only linear)
```

Description:

Sets the position of the object. 'TYPE' can be one of the following identifiers:

```
  LINEAR  the interpolation is done in a straight way.
```

```
  SPLINE  the interpolation is done in a spline curve way. (NOT
           IMPLEMENTED YET)
```

## 1.60 size

SIZE

Template:

```
FROM/A, TO/A, SIZE/A, TYPE
```

Arguments:

```
FLOAT FROM, TO
```

```
  time code
```

```
SIZE
```

```
  size at time 'TO'
```

```
KEYWORD TYPE [LINEAR]
```

```
  interpolation type (currently only linear)
```

Description:

Sets the size of the object. 'TYPE' can be one of the following identifiers:

```
  LINEAR  the interpolation is done in a straight way.
```

```
  SPLINE  the interpolation is done in a spline curve way. (NOT
           IMPLEMENTED YET)
```

## 1.61 ARexx-errors

### AREXX-ERRORS

These values are returned when something went wrong, you can get the error string with the command

```
GETERRORSTR
```

```
.
```

Application and parser errors

Here are the errors returned from the command parser and the application itself.

- 10 Wrong screen resolution  
Both components of the screen resolution have to be higher than one.
  - 11 Actor not defined  
The specified actor name does not exist.
  - 12 Surface not defined  
The specified surface name does not exist.
  - 13 Not enough memory  
Allocation of memory failed.
  - 14 Limitations of demo version reached  
The demo version is limited to a resolution of 160x128.
  - 15 Unknown brush mapping type  
You specified a unknown mapping method for the  
BRUSH  
command.
  - 16 Unknown brush wrapping method  
You specified a unknown wrapping method for the  
BRUSH  
command.
  - 17 Depth of octree too big (max. 6)  
The octree depth is limited to a depth of 6.
  - 18 Invalid time intervall  
One component of a time intervall was negative or the beginning time  
was later than the end.
  - 19 Antialiasing value too big (max. 8)  
The value of the  
ANTIALIAS  
command is limited to 8.
  - 20 Distribution value too big (max. 8)  
The value of the  
DISTRIB  
command is limited to 8.
  - 21 Unknown interpolation method  
You specified a unknown interpolation method for the  
POSITION  
,  
ALIGNMENT  
or  
SIZE  
command.
  - 22 No picture renderd  
There is no picture for
-



```

                SAVEPIC
                to save because you renderd none
or called
                CLEANUP
                before.
23 Can't open screen
The
                DISPLAY
                command was unable to open the screen
                (!!! THIS COMMAND ISN'T RELEASED IN THIS VERSION YET !!!).
24 Can't open iffparse.library
RayStorm failed to open iffparse.library (at least version 37 is
needed)
25 Can't open graphics.library
RayStorm failed to open graphics.library (at least version 33 is
needed)
26 Can't open intuition.library
RayStorm failed to open intuition.library (at least version 37 is
needed)
27 Can't open window
RayStorm failed to open the window.
28 Can't open muimaster.library
RayStorm failed to open muimaster.library (at least version 8 is
needed)
29 Invalid vector definition
The specified vector has the wrong format (must be '<x,y,z>').
30 Invalid color definition
The specified color has the wrong format (must be '[r,g,b]').
31 Invalid region definition
The specified region is out of range.

```

#### Internal errors

This are errors of the renderer.

```

101 Not enough memory
Allocation of memory failed.
102 Error in triangle definition
It's impossible to generate a triangle with the specified coordinates
(see
                TRIANGLE
                ).
103 The view and up directions are identical?
You specified a view-up-vector for the CAMERA command which is identical
to the view direction.
104 Not enough memory for screen buffer
The allocation of the screen buffer failed.
105 The backdrop picture has the wrong size
The backdrop picture must have the same resolution as the with
                SETSCREEN
                specified screen resolution.
106 Can't open Imagine texture file
RayStorm failed to open the specified Imagine texture file, check
filename and path.
107 Can't open brush file
RayStorm failed to open the specified brush file, check

```

filename and path.

108 Error initializing Imagine texture  
An error occurred as RayStorm tried to initialize a Imagine texture.

109 Can't open picture  
RayStorm failed to open the specified picture file, check filename and path.

110 Error reading picture  
An error occurred while RayStorm read a picture file.

111 Can't open picture type file ('modules/picture/types')  
RayStorm failed to open the typefile. The typefile is needed to identify the filetypes of the pictures. RayStorm was unable to open the file 'modules/picture/types'.

112 Error reading picture type file  
An error occurred while RayStorm read the picture type file, maybe the file is damaged.

113 Unknown picture format  
RayStorm was unable to recognize the format of the picture file.

114 An error occurred while invoking picture handler  
The used picture handler returned a error.

115 Can't open object  
RayStorm failed to open the specified object file, check filename and path.

116 Error reading object  
An error occurred while RayStorm read a object file.

117 Can't open object type file ('modules/object/types')  
RayStorm failed to open the typefile. The typefile is needed to identify the filetypes of the objects. RayStorm was unable to open the file 'modules/object/types'.

118 Error reading object type file  
An error occurred while RayStorm read the object type file, maybe the file is damaged.

119 Unknown object format  
RayStorm was unable to recognize the format of the object file.

120 An error occurred while invoking object handler  
The used object handler returned a error.

## 1.62 Examples

### EXAMPLES

We have included several demos in the directories 'rexx' and 'examples' to show how to use RayStorm.

In the 'arexx' directory are examples scripts which show the usage of RayStorm with ARexx. Start them simply by typing 'rx ??? .ray' in a shell (??? .ray is the name of the script).

Attrtest.ray  
Several examples for attributes.

Attrtest1.ray  
Several examples for attributes.

Backdrop.ray  
Demonstrates usage of backdrop picture.

---

Bounce.ray

Tutorial

.

Brush.ray

Demonstrates usage of brush mapping.

Bump.ray

Test of bump texture.

Checker.ray

Test of checker texture.

Chess.ray

Chess scene.

Coin.ray

Jumping coin with motion blur.

Dof.ray

Test of depth of field.

Eight.ray

Billard scene.

Fog.ray

Fog demonstration.

Fogl.ray

Fog demonstration.

Im\_texture.ray

Example for usage of Imagine textures.

Marble.ray

Test of marble texture.

Randomsphere.ray

Randomly colored sphere.

Simple.ray

Tutorial

.

Supersample.ray

Demonstrates adaptive supersampling.

Title.ray

Renders the RayStorm title.

Title1.ray

Renders the RayStorm title.

Wood.ray

---

Test of wood texture.

In the 'examples' directory are C-programs which show the usage of RayStorm directly with a program. They can only be run from a shell. These programs are producing a couple of pictures no animation, which must be glued together with a utility like MainActor.

Sphanim

Animation of several spheres which jump over a checker board. Camera follows them.

Worldanim

Rotating world.

## 1.63 Tutorials

TUTORIALS

Simple scene

Bouncing ball

## 1.64 Simple scene

Tutorial: Simple scene

Now we will create a very famous scene. A sphere over a checkerboard! This might be boring, but it's good for the absolute beginner to get an impression of building a scene.

Here we go:

1. In the drawer 'ARexx' of the RayStorm directory there is a file named 'default.ray'. This is a default form for RayStorm ARexx scripts. You can use this form to write your own scripts. We'll use this file as a default for our animation script. Copy this file to the file 'simple.ray'. After this load the file 'simple.ray' to your favorite text editor (e.g GoldEd or CygnusEd).
2. To view the scene, we need a camera. Insert after the line 'address RAYSTORM' the line:

```
'
    SETCAMERA
    <6,1.5,-1.5> <0,0,0> <0,1,0>'
```

This sets the camera to position <6,1.5,-1.5>. The camera points to <0,0,0> and the view-up vector is <0,1,0>. Note that you don't have to specify every single parameter. Every command has default values. Refer

to the description of a command to find out the default values.

3. Nothing can be seen without a lightsource.  
Go to the next line and type:

```
,  
    POINTLIGHT  
    <0,50,0> [255,255,255] SHADOW'
```

The sphere is illuminated from above with white light.

4. Before placing the objects in the scene, you have to define their surfaces.  
Insert this line:

```
,  
    NEWSURFACE  
    planesurf'
```

This creates a surface with name planesurf. The plane has a checkered surface, so insert:

```
,  
    IMTEXTURE  
    /textures/checker.itx <0.1,0.1,0.1> <0,0,0> <2,2,2>'
```

5. That was the plane texture. Let 's go over to sphere texture.  
Add:

```
,  
    NEWSURFACE  
    spheresurf'
```

The sphere has a mirrored surface. To simulate a perfect mirror, type

```
,  
    REFLECT  
    [255,255,255]'
```

6. Now we can add the objects to the scene:

```
,  
    SPHERE  
    spheresurf <0,0.5,0> 1'
```

This creates a sphere on position <0,0.5,0> and radius 1.  
Add the plane:

```
,  
    PLANE  
    planesurf'
```

The default values for the position and the normal vector fit to our scene, so we can take them over.

7. Let's make an end to the definitions and render the scene!  
Type:
-

```

,
    STARTRENDER
,

```

8. Finally we may not forget to save the picture, so add:

```

,
    SAVEPIC
    simple.iff'

```

which will save the rendered picture in the current directory as a IFF-ILBM file.

The last step is to free all the memory with the command 'CLEANUP'. Add:

```

    CLEANUP
,

```

9. Start the script from a shell-window with the sequence 'rx simple.ray'. RayStorm will now generate your picture. When RayStorm finished the work start your favourite viewer-program, load the file and have a look at it.

Looks very monochrome!!

To make the world colorful, we make a red checker and set the sky to blue. A blue sky can be done by setting the world's background color.

10. Before 'SETCAMERA' insert:

```

,
    SETWORLD
    [30,30,255]'

```

Add

```

,
    DIFFUSE
    [155,0,0]'

```

to the surface planesurf (this defines one checker color), the other one must be set in the 'IMTEXTURE' command, so change it to

```
'IMTEXTURE /textures/checker.itx <0.1,0.1,0.1> <0,0,0> <2,2,2> 255 0 0'
```

(Note that '255 0 0' describes a color, but is not embedded in < >, because the checker color belongs to the texture parameters which are all floats.)

11. Render the scene once again, and view it.

That's the end of the tutorial! Make some changes to the scene file and play around with the parameters to see their effects.

## 1.65 Bouncing ball

## Tutorial: Bouncing ball

The goal of this tutorial is to show you how to generate little animations. At the end of this tutorial you'll have a animation where the earth rotates and bounces on a rotating plane with a white checker texture on the top and a red checker on the bottom. If you have a fast computer you can also generate the animation with motion blur.

O.k. here we go:

1. In the drawer 'ARexx' of the RayStorm directory there is a file named 'default.ray'. This is a default form for RayStorm ARexx scripts. You can use this form to write your own scripts.  
We will use this file as a default for our animation script. Copy this file to the file 'bounce.ray'. After this load the file 'bounce.ray' to your favorite text editor (e.g GoldEd or CygnusEd).

2. First we define some values: the acceleration of the ball and the amount of frames to generate.

RayStorm has three commands to set the paths where it searches the files it needs. We use a brush for the surface of the ball and a texture for the surface of the ground.

To do this we have to insert after the command 'ADDRESS RAYSTORM' the lines:

```
g = .2
frames = 17
```

```
,
```

```
BRUSHPATH
  /brushes'
```

```
,
```

```
TEXTUREPATH
  /textures'
```

It's the same if you write the commands in upper case or lower case. But it's important to enclose all commans in quotes because ARexx tries to interpret the line before it sends it. It may happen that the line is changed and RayStorm don't do the things you want.

3. Next we set the screen resolution. For the first experiments we choose a low resolution of 160x128 pixels. Insert the line:

```
,
```

```
SETSCREEN
  160 128'
```

4. Now we set the camera parameters. The first three values determine the position of the camera. We want to place it so that we can see the ball all over the time. The next values set the viewpoint of the camera, this is the point the camera aims to. The next values determine the view up vector. And the last two values determine the field of view. To get a pixel aspect of 1:1 we have to set them to 25 and 20 degree.

```
,
```

```
SETCAMERA
  <0,10,40> <0,5,0> <0,1,0> 25 20'
```

5. We want to have a bright blue background for our animation. The background and the global ambient color is set with the 'SETWORLD' command. We want to set the ambient color to a dark gray, if this color is too bright the scene will look washed out and the objects appear flat. Insert the line:

```
,
    SETWORLD
    [10,30,200] [10,10,10]'
```

6. The illumination is an important part of a scene. We want to place a pointlight near the camera. Add the line:

```
,
    POINTLIGHT
    <5,10,50>'
```

7. Now we define the actor for the plane. We want to rotate it around the Z-axis. Insert the lines:

```
,
    NEWACTOR
    groundactor'
,
    ALIGNMENT
    0 ' frames+2 ' <0,0,360>'
```

7. Now we define the surface for the plane and the plane itself. We make it a little reflective and apply a checker texture. The surface 'groundtop' is for the top of the plane and the surface 'groundbottom' is for the bottom of the surface. The plane itself consists of four triangles. Two for the top and two for the bottom. Insert the lines:

```
,
    NEWSURFACE
    groundtop'
,
    DIFFUSE
    [255,255,255]'
,
    SPECULAR
    [0,0,0]'
,
    REFLECT
    [50,50,50]'
'IMTEXTURE checker.itx <0,-1,0> <0,0,0> <10,10,10> ACTOR groundactor'
,
    NEWSURFACE
    groundbottom'
,
    DIFFUSE
    [255,0,0]'
,
    SPECULAR
    [0,0,0]'
```



```

,
    REFLECT
    [50,50,50]'
,
    IMTEXTURE
    checker.itx <0,-1,0> <0,0,0> <1.5,1.5,1.5> ACTOR groundactor'
,
    TRIANGLE
    groundtop <-2,0,-2> <2,0,-2> <2,0,2> ACTOR groundactor'
,
    TRIANGLE
    groundtop <-2,0,-2> <-2,0,2> <2,0,2> ACTOR groundactor'
,
    TRIANGLE
    groundbottom <-2,-.01,-2> <2,-.01,-2> <2,-.01,2> ACTOR ↔
    groundactor'
,
    TRIANGLE
    groundbottom <-2,-.01,-2> <-2,-.01,2> <2,-.01,2> ACTOR ↔
    groundactor'

```

8. Next we define the motion of the ball. It starts at a height of 10 and accelerates until it bounces on the plane, changes it's direction and the motions ends as the ball is back at he start point. Additional the ball rotates around the Y-axis. Add the following sequence to your script:

```

speed = -g
pos = 10
,
    NEWACTOR
    ballactor <0,'pos',0>'
do i=0 to frames
,
    POSITION
    ' i i+1 ' <0,'pos',0>'
    pos = pos+speed
    if pos<=1 & speed<0 then
        speed = -speed
    else
        speed = speed-g
end
,
    ALIGNMENT
    0 ' frames+2 ' <0,360,0>'

```

9. Now we define the surface for the ball and the ball itself. The only thing we must do is to map a earth styled brush map to a sphere. To reach this goal the position of the brush must be set to the middle of the sphere and the size must be small enough to be completely inside the sphere. This are the lines to define the ball:

```

,
    NEWSURFACE
    ball'
,
    BRUSH

```

```
earth.iff COLOR WRAPXY <0,10,0> <0,0,0> <.1,.1,.1> ACTOR ←
ballactor'
```

```
,
SPHERE
ball <0,10,0> 1 ACTOR ballactor'
```

10. If your computer is fast enough you can insert the following lines:

```
,
ANTIALIAS
2'
,
DISTRIB
2'
```

'ANTIALIAS' improves the quality of the picture; 2 or 3 are normal values, higher values don't improve the quality significant.

A value higher than one for 'DISTRIB' switches {"motion blur" link Motion Blur} ← on.

11. At this the we have finished the definitions and now can render the single frames. If youn want the reflections of the ball on the plane you have to delete the keyword 'QUICK', because RayStorm renders no reflections in quick mode. The frame time is set with 'FROM' and 'TO'. We save the frames as IFF-ILBM pictures with the names 'bounce0001.iff' ... 'bounceXXXX.iff'. The last step is to free all the memory with the command 'CLEANUP'. Add these lines:

```
do i=0 to frames
,
STARTRENDER
QUICK FROM 'i' TO 'i+1
,
SAVEPIC
bounce' || RIGHT(i,4,0) || '.iff'
end
,
CLEANUP
,
```

12. Start the script from a shell-window with the sequence 'rx bounce.ray'. RayStorm will now generate your frames. When RayStorm finished the work you must glue the pictures together to get the animation.

That's all. Have fun!

## 1.66 Tips&Tricks

Tips&Tricks

- The commands '  
TEXTUREPATH  
, ,

```
OBJECTPATH  
' and '  
BRUSHPATH  
' are relative
```

to the directory RayStorm is started from.

- RayStorm renders faster if you don't use planes, because intersections with planes cannot be calculated with the octree (helpfull in scenes with motion blur or soft shadows).
- If RayStorm crashes with scenes with reflections and transparence, try to start RayStorm with a larger stack (e.g. 8192 Bytes).

## 1.67 Textures

```
TEXTURES
```

Textures are mathematical generated patterns which can be applied to the surface of a object.

There are several textures in the directory 'textures'.

```
Bump  
Checker  
Linear  
Marble  
Radial  
Stars  
Wood
```

## 1.68 Bump

```
BUMP
```

This texture applies bumps to the surface.  
Size of texture determines size of the bumps.

Parameters:

X bump size - Y bump size - Z bump size  
Sets the 'depth' of the bumps.

Example:

```
sphere with radius 1  
IMTEXTURE bump.itx <0,0,0> <0,0,0> <.3,.3,.3> 1 1 1
```

Picture

## 1.69 Checker

CHECKER

This texture applies a normal checks pattern to the surface.

Attention!

If you apply a checker texture to a plane, the plane may not be at the same position on which the checker changes its color. Otherwise you get a noisy texture due to rounding errors.

Parameters:

Color Red - Color Green - Color Blue

Color of the checks, other color is taken from object.

Reflect Red - Reflect Green - Reflect Blue

Reflect color of the checks.

Filter Red - Filter Green - Filter Blue

Filter color of the checks.

Example

## 1.70 Linear

LINEAR

This texture varies the color of the object in the y-direction of the texture.

Parameters:

Color Red - Color Green - Color Blue

Color to interpolate to.

Reflect Red - Reflect Green - Reflect Blue

Reflect to interpolate to.

Filter Red - Filter Green - Filter Blue

Filter to interpolate to.

Example

## 1.71 Wood

WOOD

This texture applies a wood like texture to the surface.

---

Size of texture determines size of wood.

Parameters:

Color Red - Color Green - Color Blue  
Color. Other color is taken from object.

Reflect Red - Reflect Green - Reflect Blue  
Reflect color.

Filter Red - Filter Green - Filter Blue  
Filter color.

Octave  
The higher the octave the noisier are the wood rings.

Frequency  
The higher the frequency the smaller the wood rings.

Example:  
cube with size 2  
IMTEXTURE wood.itx <0,0,0> <0,0,0> <1,1,1> 255 255 50 0 0 0 0 0 0 2 4

Picture

## 1.72 Marble

MARBLE

This texture applies a marble like texture to the surface.  
Size of texture determines size of bumps.

Parameters:

Color Red - Color Green - Color Blue  
Color. Other color is taken from object.

Reflect Red - Reflect Green - Reflect Blue  
Reflect color.

Filter Red - Filter Green - Filter Blue  
Filter color.

Octave  
The higher the octave the noisier is the texture.

Example:  
cube with size 2  
IMTEXTURE marble.itx <0,0,0> <0,0,0> <1,1,1> 150 50 50 0 0 0 0 0 0 0 7

Picture

---

## 1.73 Radial

RADIAL

This texture varies the color of the object radial around the texture axis.

Parameters:

Start radius

Interpolation start radius.

End radius

Interpolation end radius.

Color Red - Color Green - Color Blue

Color to interpolate to.

Reflect Red - Reflect Green - Reflect Blue

Reflect to interpolate to.

Filter Red - Filter Green - Filter Blue

Filter to interpolate to.

Example

## 1.74 Stars

STARS

This texture applies randomly stars to the surface.

Parameters:

Color Red - Color Green - Color Blue

Color of the stars.

Density

Star density. The higher the more stars (0. - 1.).

Example

## 1.75 Known Bugs

KNOWN BUGS

- RayStorm crashes with some Imagine textures (DinoSkin, FrogSkin, ...) if you use MUI 3.2 (maybe also 3.0 and 3.1). With MUI 2.3 it all works fine. Maybe there'll be a newer version of MUI soon.

## 1.76 Legal Stuff

### DISCLAIMER

THERE IS NO WARRANTY FOR THIS PROGRAM TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHERE OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDER AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

### COPYRIGHT

RayStorm 1.1 and RayStorm Demo 1.1 are Copyright 1995 by Andreas Heumann and Mike Hesser.

All Rights Reserved. It is released under the concept of 'Shareware'.

The archive of the RayStorm Demo may only be distributed in unmodified form. No files may be added, changed or removed. You may not charge for this archive, other than the cost of the media and duplication fees. Distribution is allowed in all forms, such as BB systems, floppy or compact disks, and ftp sites.

## 1.77 Credits

### CREDITS

We want to thank the following persons:

- Stephan Dorenkamp & Marcus Ritter - for testing

## 1.78 Register

### REGISTER

If you like RayStorm use the registration programm to register. Fill out the registration form and press the Print button. If the printer is installed correctly, the registration is printed out. You can get information about the current agreements by pressing the Info button.

---

## 1.79 Author

### AUTHORS

For bug reports, comments, suggestions ... you can contact us at the following addresses.

Andreas Heumann

E-mail: heumann@hugo.rz.fh-ulm.de  
S-mail: Heilmeyersteige 105  
89075 Ulm  
Germany

Mike Hesser

E-mail: calvin@sol.wohnheim.uni-ulm.de  
S-mail: Heilmeyersteige 105  
89075 Ulm  
Germany

## 1.80 History of Changes

### HISTORY

version 1.0 (09-July-95)  
- first release.

version 1.01 (15-August-95)  
- added soft shadows  
- added random jitter  
- added brush repeat and mirror

version 1.02 (16-August-95)  
- bugfix: altitude mapping -> black object: fixed  
- bugfix: loading of TDDD-objects with brushes crashed: fixed  
- added backdrop picture  
- added BRIGHT-flags for surface  
- added fog  
- deleted TRANSATTU

version 1.03 (17-August-95)  
- bugfix: sphere intersection test: fixed

version 1.04 (21-August-95)  
- added global fog

version 1.05 (28-August-95)  
- added animation commands

version 1.06 (01-September-95)  
- added motion blur

---



version 1.07 (10-September-95)

- added specular brush mapping

version 1.08 (11-September-95)

- added rendering box

version 1.081 (08-October-95)

- added listview for history
- added global reflection map
- changed error messages

version 1.082 (11-October-95)

- improved memory management for Imagine objects

version 1.083 (12-October-95)

- changed spotlight direction to lookpoint and added actor for lookpoint
- new form for vectors '<x,y,z>'
- new form for colors '[r,g,b]'

version 1.1 (18-October-95)

- next official release

version 1.11 (19-October-95)

- bugfix: Imagine fog objects are now loaded properly
- added parameter check for field rendering

version 1.12 (21-October-95)

- speedup of motion blur

version 1.13 (01-November-95)

- now more than one path with PATH-commands possible
- bugfix: spotlight look point changed camera view point
- added soft interpolation of colors for brushmapping
- bugfix: objects behind light sources casted shadows

version 1.14 (03-November-95)

- changed default gaussian filter width from 1.8 to 1.3
- bugfix: problem with global fog
- plane can now be animated
- changed axis position in flat brush mapping
- added 'Time spend' and 'Time left'

version 1.15 (28-November-95)

- added PNG- and ILBM-modules
- added radial texture

version 1.16 (09-January-96)

- bugfix in PNG-module: had problems with palette pictures
  - bugfix motion blur: had a problem with voxel calculations
  - bugfix motion blur: had no motion blur in scenes with planes
  - added TGA-module
  - added 'SOFTSHADOW' to 'DISTRIB'
  - optimized octree (up to 10% faster)
  - added 'FALLOFF' for 'POINTLIGHT' and 'SPOTLIGHT'
  - bugfix can now load plane
  - bugfix can now load multiple planes or perfect spheres
-

version 1.17 (18-February-96)

- object loading is now done in modules
- added 3DS module
- bugfix in TDDD-module: hardness is now used to set REFEXP
- brushes and textures are no applied to surfaces which lie between the light source and the illuminated surface
- bugfix antialiasing: there where some bright pixel trash in the picture if antialiasing with fog was used
- rewrote bump, wood and marble textures
- global reflections map is now applied in 'QUICK' mode too

version 1.2 (29-February-96)

- bugfix 'ANTIALIAS' command: parameter CONTRIB produced error 30
- bugfix: last line of picture has been always black
- bugfix: had problems with Imagine 4.0 TDDD objects

## 1.81 PC-version

PC-VERSION

The PC version is available on the Internet.

The most import differences between the PC-Version and the Amiga-Version are:

- the Amiga-Version is able to load Imagine texture-files
- the PC-Version uses its own script language, whereas the Amiga-Version uses ARExx

## 1.82 Homepage

HOMEPAGE

Come and visit our RayStorm-Homepage! There you can always get the latest version of RayStorm and can see some example pictures.

The address:

<http://sol.wohnheim.uni-ulm.de/~calvin/raystorm.html>

## 1.83 Future

FUTURE ADDITIONS

- modeler
  - more objects (torus, cylinder, ...)
  - JPEG-saver
  - use Imagine staging files
  - animation language (ALAN)
  - shadow caching
  - more textures
-

- don't allocate whole picture buffer at once
  - diffuse reflectivity
  - diffuse transparency
  - better light FX
  - log file
  - apply post-2D-FX
  - spline interpolation
  - load Lightwave format
  - CSG (Constructive Solid Geometry)
-