BOGL

Jarno van Linden

COLLABORATORS			
	TITLE : BOGL		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Jarno van Linden	August 7, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

BOGL

1.1 BOGL: The Blanker for OpenGL programs

```
                BOGL v1.0
Copyright © 1998/1999 by Jarno van der Linden
jarno@kcbbs.gen.nz

A Madhouse screen blanker module that runs many AmigaMesaRTL
compiled OpenGL programs


              Introduction
              - This thing is cool!

              Requirements
              - Stuff you really should have

              Installation
              - Get up and go!

              Programme specification
              - Setting up a programme file

              How it works
              - How the magic is done

              Nasty stuff
              - Danger, Will Robinson! Problems, bugs
```

1.2 BOGL - Introduction

```
INTRODUCTION

BOGL allows you to run many OpenGL demos compiled with AmigaMesaRTL
as a screen blanker using the Madhouse modular screen blanker system.

For example, one of the popular blankers on other computer systems
```

shows morphing 3D polyhedra. An OpenGL version of this exists, and is distributed with Mesa as Morph3D.c. Using BOGL, the Amiga now too has a morphing polyhedra screen blanker, without making any changes to the Morph3D source code! Similarly, virtually any Mesa demo can now be run as a screen blanker without modification or recompiling.

Some demos require user input to do anything interesting, so BOGL gives you the ability to automatically control the OpenGL program.

1.3 BOGL - Requirements

REQUIREMENTS

There are a few things that you absolutely should have to make BOGL work:

- Madhouse v2.6
- AmigaMesaRTL libraries (at least v2.0)
- Some OpenGL programs compiled with AmigaMesaRTL
- Lots of memory

1.4 BOGL - How it works

HOW IT WORKS

BOGL consists of three parts:

- The BOGL Madhouse screen blanker module
- The Blank output handler
- A programme data file describing which demos to run and how

OpenGL programs compiled with AmigaMesaRTL typically ask an output handler to open a window. This is particularly true for programs using AmigaMesaRTL's GLUT implementation.

When the BOGL blanker module is activated, it opens a screen and window, passes the address of the window to the Blank output handler, sets the default output handler for windows to the Blank output handler, and runs a program randomly chosen from the programme data file.

The demo will now use the Blank output handler for rendering. When the demo requests a window, the output handler will give it the window created by the BOGL blanker module.

Hey presto! Any OpenGL program which nicely uses a window output handler will run as a screen blanker.

To unblank, the BOGL module sends a Control-C (^C) signal to the OpenGL program.

1.5 BOGL - Nasty stuff

NASTY STUFF

There are some issues and bugs which you should be aware of:

- Only those OpenGL programs which request the output handler set by the AmigaMesaRTL/Window environment variable for a window will work with BOGL. A program which ignores the environment variable or opens its own window will not work.

- A program run by BOGL better respond to a ^C signal by terminating, or the BOGL module will wait forever, preventing any further use of the screen blanker.

- Faking user interaction with the OpenGL program is done by creating fake IntuiMessages and sending them to the window's UserPort. However, only Intuition really knows how to properly construct IntuiMessages, and only Intuition should be sending messages to a window's UserPort. Hence this is dodgy at best, and a thoroughly illegal hack which will blow up your computer at worst.

- The OpenGL program terminating unexpectedly is a Bad Thing. It may or may not cause strange things to happen.

- The OpenGL program failing to launch is similarly a Bad Thing which may or may not be handled gracefully. So make sure the programme data file is correct.

- It may be some seconds before an OpenGL program checks its signals to see if there is a ^C signal. As you probably want the blanker to stop as soon as there is keyboard or mouse activity, the blanking screen is pushed to the back as soon as Madhouse wants the blanker to quit. Madhouse is then told that the blanker has finished. This isn't entirely true. The blanker is waiting for the OpenGL program to terminate.

- The AmigaMesaRTL/Window environment variable MUST be set to a valid window output handler (I recommend DL1Plus). Do NOT set it to the Blank output handler. Here be dragons!

1.6 BOGL - Installation

INSTALLATION

- Copy the BOGL directory to your Madhouse blankers directory
- Copy the Blank output handler to your output handlers directory
- Make sure that you have a valid window output handler specified in the AmigaMesaRTL/Window environment variable
- Modify the BOGL.programme data file to match your system and tastes
- Use Madhouse to set the preferences for the BOGL blanker module

1.7 BOGL - Programme specification

```
                PROGRAMME SPECIFICATION

BOGL uses a data file to determine what OpenGL programs to run, and
how to run them. The data file contains a number of entries, one
entry for every program. Which program is actually run is determined
randomly from those present in the data file.

An entry is composed of one or more lines. Each line is composed of
a four letter tag followed by a space, and a string of data. A line
starting with a semi-colon (;) is a comment line and is ignored.

The tags recognised are:

  PROG <program with full path>
  ARGS <argument string passed to program>
  CDIR <current directory to use by program>
  SCRN <monitor ID> <screen depth>
  VKEY <start> <step> <end> <ASCII code | 'char'>
  INEV <start> <step> <end> <input event string>
  MENU <start> <step> <end> <menu #> <item #> <sub #>

A programme entry starts with PROG and ends with whitespace or the
end of the file. The other tags can appear in any order any number
of times in the entry.

Start, step, and end values are in seconds, unless step size is
negative in which case the values are interpreted as frame counts.
For indefinite repetition, use an end value of -1.

See the example BOGL.programme for inspiration.

PROG <program with full path>
You must give the full path to the program, otherwise the program
probably won't be found.

Example:
; The Morph3D demo, from my Mesa:demos directory
PROG Mesa:demos/Morph3D

ARGS <argument string passed to program>
You may like to pass some arguments to a program. It is probably a
bad idea to pass in output handler information recognized by
AmigaMesaRTL's GLUT implementation (e.g. -outputhandler). It would
interfere with the
                workings
                 of the blanker

Example:
; Run program using nearest sampling and colour index mode
ARGS -nearest -ci

CDIR <current directory to use by program>
Some OpenGL programs read in files, such as textures, from their
current directory. This tag allows you to set the current directory
```

```
for the OpenGL program. If you don't specify this tag, it will
default to BOGL's current directory. If an OpenGL program tries to
read, for example, a texture file relative from the current
directory, and the current directory is not set correctly, the
program will probably terminate prematurely. This can be
                bad
                .

Example:
; Make sure the current dir is the Mesa:demos directory for textures
CDIR Mesa:demos

SCRN <monitor ID> <screen depth>
Allows you to set the monitor ID and screen depth to use for this
program. This overrides the preferences set in Madhouse. Very useful
for speeding up programs which require, for example, only a few
colours (e.g. the Bounce demo).

Example:
; Override BOGL preference settings, use 3 bitplane DBLPal
SCRN 659456 3

VKEY <start> <step> <end> <ASCII code | 'char'>
Specifies the sending of a Vanilla key event. These are typically
ASCII characters. In GLUT, these are the events that cause the
keyboard function (set by glutKeyboardFunc()) to be called.

Example:
; Send z keypress every 2 frames, from frame 10 until forever
VKEY 10 -2 -1 'z'
; Send space keypress once at the beginning
VKEY 0 0 0 32
; Send x keypress every second between 9 and 16 seconds
VKEY 9 1 16 'x'

INEV <start> <step> <end> <input event string>
Only intended to send rawkey codes, this tag may be able to send any
input event. Typically, these events end up calling the
glutSpecialFunc() function. The input event string follows the
format used by Commodity's InvertString(). Basically it is the
format familiar from specifying hotkeys for commodities enclosed in
angle brackets.

Example:
; Send a cursor left every 2 frames from frame 10 until forever
INEV 10 -2 -1 <left>
; After 20 seconds, send function key 1 and a cursor up and right
INEV 20 0 20 <f1><up><right>

MENU <start> <step> <end> <menu #> <item #> <sub #>
Controls menu selection. Menu numbers are counted from 0, use -1 if
the number is not applicable (e.g. there is no sub-menu). Note that
in GLUT, the most commonly used menu is the right mousebutton menu,
which in AmigaMesaRTL's GLUT is menu number 2

Example:
; Reset time in pointblast demo (menu 2, item 0, no submenu)
```

```
; every 60 seconds
MENU 60 60 -1 2 0 -1
```