NAME Manual

COLLABORATORS			
	TITLE : NAME Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

NAME Manual

1.1 Contents

Puppeteer Manual V1.2

An Articulated Hierarchical Model Viewer
For CyberGL

1.2 Introduction

Introduction

Puppeteer is a port of my graphics project for cs488 at the university of Waterloo (where I have finally graduated after far to long). The project was written in OpenGL with the user interface written in Tcl/Tk and ran on a rather antiquated SGI Iris (they upgraded to Octanes the next term). I invested a lot of time in this project (or rather I invested a lot of sleepless hours during a brief 2-3 week period) and wanted to clean up the code and some of the bugs, as well as perform some experiments with the system I developed.

Another reason for doing the port was to experiment with CyberGL, to see how full an implementation of OpenGL it was, and how well it worked. The documentation in the developer distribution for CyberGL was rather sparse and I have not seen any programs that use CyberGL other than the demos supplied with the archive (and they were pretty straightforward).

It seems that every software product that Phase5 is associated with is shrouded in controversy. This demo probably wouldn't be enhanced by a version of CyberGL for the Cybervision64/3D (since it uses no texture mapping and this seems to be the main function of the VIRGE chipset) but it should take advantage of a PowerPC version of CyberGL and a version of CyberGL optimized for the CybervisionPPC card (or any other 3D graphics card for the Amiga that was supported by CyberGL). I hoped that by writing this I would be able to have something that could run fast on my PPC card when I purchased it. (And that other people would have a demo that showed how much faster CyberGL was on their PPC).

It took me about 2 weeks to make this
port
but most of the time was spent
cleaning up the SGI code, and translating the
Tcl
code to
CPGui
code. This
is a really quick and dirty port that I threw together. It has little error checking and virtually no error reporting. It requires at least on 030 to run, but performance isn't really acceptable on an 030 25Mhz. Unfortunately, I don't know what the minimum reasonable spec machine to run the demo on is, since I have a A3000 030/25 with a Picasso II video card, which means I can't really run it myself. Thus, the animations might not look right, because I haven't really seen them running smoothly on the Amiga.

1.3 Description

Description

Puppeteer is a model viewer that displays articulated hierarchical models and allows the user to manipulate them using an inverse kinematics system. A hierarchical model is a model that consists of a set of primitives positioned relative to each other. Forward kinematics is the manipulation of such a system by specifying the angle of each joint. This makes it difficult to specify simple configurations of the system intuitively. (Forward kinematics was the assignment that the project was based on). Inverse kinematics is the manipulation of a hierarchical system by specifying goal configurations for primitives and solving for the joint angles that yield the goal.

Puppeteer uses a Jacobian matrix to represent the system, and uses a damped least squares approach to solve the Jacobian in order to account for the inherent instability of the Jacobian (and the problem). If you want a more comprehensive description of the numerics I could give you a copy of my paper, but it's a bit much to go into here (and I'm lazy). This method produces an approximate solution to the system. An exact solution could be obtained by iterating towards the goal, but because the system is being used with direct user feedback the lack of an exact solution gives a realistic "feel" to the model. Ropes kink and need to be "tugged" straight, and arms will twist into orientations that they must be "yanked" out of.

Puppeteer uses the inverse kinematics to implement gravity and wind based animations. You can manipulate the model and watch the effects of wind and gravity on it. I have supplied several sample models, and more can be

created
using a text editor (and a lot of patience).

1.4 Requirements

Requirements

Puppeteer requires an 030 and an FPU. It also requires OS 2.04 and CyberGL library V39.11 (Plus all requirements for running CyberGL). However, since it has very sparse error codes, and little testing was performed I'm listing the version numbers of all libraries that it opens here to make debugging easier:

```
   intuition.library    V37
   graphics.library     V36
   utility.library      V37
   gadtools.library     V37
   asl.library          V37
   icon.library         V33
   cybergl.library      V39
   cybergrapics.library V40 (Not required)
```

I'm not sure what screenmodes Puppeteer will work on, because the CyberGL documentation doesn't specify the screenmode restrictions. The web page says ECS is supported, but I wasn't able to open a window on an ECS screen. Perhaps RGBA mode cannot be used on ECS but I am not sure, since it is not documented. So it may or may not work on 8 bit AGA screens. Personally, I always run at 16 bit, because this gives the best tradeoff between performance and appearance on my Picasso II.

1.5 Tcl

Tcl/Tk

Tcl is an interpreted scripting language. Tk is an extension to Tcl that provides windows and widgets for creating user interface elements. It is relatively easy to add C functions to Tcl so a C program can be implemented with the user interface written in Tcl. As on aside, the same thing could be done on the Amiga using Arexx. It would make for very customizable applications. I'm not sure if anyone has done this, or if any Arexx libraries exist with enough functionality to do this (i.e.: Tk for Arexx), but it would be interesting to try (I'm not going to do it though). There is a Tcl interpreter for the Amiga, but to my knowledge, no Tk implementation for Tcl.

1.6 CPGui

CPGui

CPGui, which stands for Cross Platform Graphical User Interface is the name of a class library I wrote to simplify porting code between various operating systems. It isolates all of the OS specific stuff in one set of classes. By porting these classes the entire program is ported. Since CPGui is my own library the idea is to implement methods as I need them. To increase flexability the rendering functions can be implemented in machine specific classes, or using generic classes like the OGLWidget, or the double buffered raster widget. Thus simple code that needs to be portable can use the emulated widgets and works on all platforms instantly, but code that requires more efficiency can implement all of it's low level rendering in specific classes for the target architecture. Porting then requires rewriting these classes, which is usually a simple task.

The base window classes have been ported to MacOs, Windows, X11 and Amiga, but the Amiga is the only platform to implement the OGLWidget. Porting the widget to other platforms would be simple as it consists of about 200 lines of code. Currently CPGui has no error reporting classes, which is why Puppeteer has no error reporting.

CPGui is far from a full user interface library, but it is great for writing simple apps, games and prototype software. By compiling for different systems, I can catch a lot of bugs that would have gone unnoticed. It's like using multiple compilers, with the added advantage that not only the code, but the execution environment changes.

1.7 Conditions of Use

Conditions of Use

Well, I'm not going to tell you not to send me money, but I'm also not going to require you to send me any if you use Puppeteer, since it doesn't really do anything. If you really like Puppeteer you could give me something I would prefer to money - a
job
. Or you could send me a CyberstormPPC, and a CyberVisionPPC (when they come out) so that I can actually run my own program.:) (With that kind of hardware I could develop some really cool CyberGL demos).

If you get a kick out of Puppeteer, or just find it interesting I would appreciate a
letter
, postcard, what have you, with any feedback, benchmarks for your machine, or the brand of toothpaste your dog uses. (Note: I am notoriously slow at responding to email, but I do get around to it.)

1.8 Using Puppeteer

Using Puppeteer

Puppeteer currently outputs all error messages to the cli, so if you have trouble using Puppeteer try running it from a shell to see what messages it returns. The messages aren't very descriptive, and the most likely cause for failure is that it couldn't find the cybergl.library in your libs: drawer. The default model is searched for in the directory "models", which should be in the same directory as the program (if you want the default file to load). There are currently no arguments, no tooltypes, and no preferences.

When first started Puppeteer displays a window with two string display gadgets at the top, and a black rectangle containing a white line. Believe it or not, that white line is what you paid for. The first string display contains the frames per second for the last ten frames. Since the program doesn't always render frames, the frame rate will drop when the model is first moved until ten frames have been rendered. The second string display is for messages, and currently indicates the mouse mode.

Mouse Buttons

The primary control method is the mouse, and there are two mouse modes. The mouse modes determine what kind of object the action will be applied to. In both modes the mouse buttons are assigned the following functions:

```
Select: Dragging with the select button moves the object left and
        right and up and down.  Clicking with the select button
        selects the joint under the mouse.  Clicking on the joint
        again deselects it.
Middle: (You mean you don't have a three button mouse? The middle
        button is simulated by holding down the alt key and using
        the select button).  Dragging with the middle button down
        moves the object towards and away from the camera.  Moving
        up pushes it away, down brings it closer.  Left and right
        do nothing.
Menu:   Dragging with the menu button down implements a virtual
        trackball.  If the "Rotate" item is checked in the
        "Animate" menu the model will continue to rotate when the
        mouse is released if it was moving at the time the button
        came up.  A virtual trackball simulates a large trackball
        being embedded in the screen.  Clicking and dragging on
        the trackball has the effect of rotating the model in the
        direction the ball would rotate if you had pushed it in
        that direction.  Rotating the ball by the height of the
        window should rotate the model by approximately 180 degrees.
        Rotating the mouse around the outside of the trackball
        should rotate the model around the camera in the indicated
        direction.
```

Note that there is not a 1 to 1 corespondence between the mouse movement and the movement of the object on the screen. Moving the mouse left pushes in the indicated direction, but the object may not move by as many pixels as the mouse moves (in "Position Model" mode it does move as many units as indicated, but since it is foreshortened by perspective it will appear to move more or less depending on its distance fro the camera). If the mouse mode is "Position Model" then the effect of the buttons is applied to the entire model. If the mouse mode is "Position Joints" then the effect of the buttons is applied to all selected joints. Because it is applied to all joints, only one joint on each path to the root may be selected at once. If multiple buttons are held down the effect of all buttons will be applied at once. So it is possible to translate the model along all three axis and rotate it at the same time.

Menus

Because the menu button must be trapped even when the mouse is outside the window (to catch button releases) the menus are not activated until the mouse is over the menu bar. On a slow machine it could take a second or two before the menus are activated if rendering is in progress. Furthermore, depending on the mode, each unsuccessful press can cause the program to render again, causing even more slowdown.

The Project Menu
- Open: Opens a file requestor to select a new model. If a valid model is selected the current model is deleted and the new one is displayed.
- Reset Position: Moves the model to a visible position in front of the camera.
- Reset Orientation: Resets the orientation of the model so that it is not rotated.
- Reset Joints: Resets all joints to be in the middle of their constraint range.
- Reset All: The same as selecting the above three entries.
- Quit: Exits the program.

The Edit Menu
- Select None: Deselects all selected joints.

The Mode Menu
- Position Model: Sets the mouse mode to "Position Model".
- Position Joints: Sets the mouse mode to "Position Joints".

The Display Menu
- Joints: If checked all joints will be represented by a red sphere.
- Primitives: If checked the models primitives (cubes) will be rendered in their default colour.
- Wind Direction: If checked a nearly invisible line from the origin will indicate wind speed and direction.

The Effects Menu
- Wireframe: If checked the model will be drawn in wireframe.
- Lighting: If checked OpenGL lighting will be used to shade the model. If not checked the model will be solid white.
- Fog: If checked the model will get dimmer as it gets further from the camera.
- Depth Buffer: If checked the depth buffer will be used for hidden surface removal. If not checked Picasso will be consulted for hidden surface removal.
- Smooth Shading: If checked all polygons will be Gouraud shaded. If not checked all polygons will be a uniform colour (flat shaded).

The Animate Menu
- Gravity: If checked a pseudo gravity will be applied to all joints. While not very accurate, it does draw all

```
              parts of the model downward.
   · Wind: If checked a random wind force will be applied to all
           joints.  The wind changes direction and gusts
           occasionally.
   · Rotate: If checked the virtual trackball works like in
             Open Inventor.  When the mouse button is released
             the model continues to move with its current
             velocity and direction.  Currently the velocity
             is not precisely matched, so there may be a
             change is speed when the button is released.
   · Fast Move: If checked the model is rendered as an unshaded
                wireframe when the mouse button is held down.  When
                the button is released, the default settings are
                restored.

                            Error Handling

   Currently very few errors are reported by the program, and these are only
reported to the console.  If you received the error "Error during setup.
Aborting."  then either there was not enough memory or the correct version of
the cybergl.library could not be found.  If the error is "Machine doesn't meet
specifications" then the version of one of the other libraries is not current
enough. See
                requirements.
                .
```

1.9 Model File Format

```
                                                File Format

   The model description language is based loosely on the
                Tcl
                 scripts
used in assignment 3 for cs488.  Since I didn't want to write a full
                Tcl
                interpreter only enough commands to describe models are included.  ↩
                   Furthermore,
since I don't really expect anyone to make their own models this description
will be brief.  Defining hierarchical models can be an art form, particularly
when articulated, and an understanding of how matrix transformations compose
together is particularly useful.  Two peculiarities of the implementation are
that scale transformations do not apply to nodes further down the tree, and
the set origin primitive allows you to set the origin of a node.  This allows
nodes to be eliminated from the tree that would be necessary to isolate
descendants from the effects of these transformations.

   The nodes of the tree are specified by name.  The name of each node is the
colon separated list of all nodes in the tree leading up to the node (like a
pathname).  Commands that create a node take a parent node as a parameter, and
the name of the new node.  The two will be concatenated to form the full name
of the new node.  Commands that set a node attribute take the name of the node
and the arguments.  Arguments to commands are all single entries except for
vectors, points and colours which are enclosed in '{}' braces and contain 3
numeric values separated by spaces.
```

```
   For convenience, variables may be defined with the set command and
dereferenced by preceding the label with the '$' character.  Variables may
consist of any string not containing a space (unless enclosed in '{}' braces).

The commands are:
   Node Creation:
      gr_cube ParentName NodeName
      gr_startMesh ParentName NodeName
      gr_addMesh ParentName NodeName Points
      gr_endMesh ParentName NodeName
      gr_mesh ParentName NodeName Points
      gr_transform ParentName NodeName
      gr_rotJoint ParentName NodeName Axis
      gr_transJoint ParentName NodeName Axis
      gr_translateAnim ParentName NodeName StartPos EndPos Frames
      gr_rotateAnim ParentName NodeName RotationAxis Speed
      gr_material MaterialName Diffuse Specular Shininess
      gr_texture MaterialName [filename]
   Node Initialization:
      gr_setOrigin NodeName Origin
      gr_identity NodeName
      gr_scale NodeName ScaleVector
      gr_translate NodeName TranslateVector
      gr_rotate NodeName Axis Amount
      gr_surfaceproperty NodeName MaterialName
      gr_limitJoint NodeName MinVal MaxVal [InitialVal]
   Global Attributes Associated With the Model:
      gr_addWind WindDirection
      set Variable_Name Value
      positionHint Position
      orientationHint OrientationVector
   Misc:
      gr_objectDone

                         Supplied Models

   Seven models are supplied with the program.  They are:

antenna.gr:  This demonstrates the use of prismatic (translation)
             joints.  It consists of a chain of rectangles
             attached by prismatic joints at the ends.  It can be
             stretched out to a long chain, or collapsed to one
             link.  Unfortunately prismatic joints don't work
             well and can't be combined well with rotation
             joints.  (A problem in the original project, not
             just the Amiga version).
puppet.gr:   A version of my model for A3.  A robot rendered in
             a beautiful metallic tone with a specular highlight
             that catches the light wonderfully as the model is
             rotated.  The robot has  a Ceylon eye (which is most
             definitely not a red nose) which will bounce back and
             forth when animations are active.  (The eye makes
             use of a translateAnim node.)  The limbs are
             articulated and many contain multiple joints at one
             node.  The robot looks quite limp with wind and
             gravity turned on, and has a Raggedy Ann look if you
             start him tumbling in a gravity field.
```

```
chain.gr:     This is a series of rectangles tied together at
              the ends in a long chain.  Each link has 3
              joints, one in each of the x,y and z directions.
              The chain is constrained so that it can only
              bend by 30 degrees at each joint.  If you set
              a rotation goal for the last link of the chain
              the whole chain will curl up as each link hits
              its constraint until the chain is completely
              locked.
rope.gr:      This model is identical to the chain, but it
              has no constraints.  It is actually quite
              entertaining because it gets kinked and
              requires tugs to straighten it.  It will
              also rebound from the end of its reach
              realistically.  While experimenting with the
              damping factor it varied from a bungy cord
              to a piece of string in its springiness.
              In a future version I may add the damping
              factor as a parameter.
slug.gr:      This is a simple demonstration of hierarchical meshes.
              It doesn’t really look like anything, but it displays
              an interesting fabric quality (especially when twisted).
              It is just the rope model with a hierarchical mesh hung
              on it.
worm.gr:      This was the root of the dinosaur model.  It just
              looked kind of neat when bent, so I kept it.
dino.gr:      This is the model to demonstrate hierarchical meshes
              and texture mapping.  It is based on a gif of a
              dinosour skeleton, but I lost the name of the
              skeleton so I don’t know what kind of dinosaur it is.
              It took me three or four days to design this model,
              not counting the time my A3000 was down.  The main body
              is a hierarchical mesh, and the legs arms and head are
              normal meshes.  As with the puppet, it looks pretty
              pathetic when gravity is turned on.  This model is
              really not finished, since the constraints aren’t
              quite right, and the arms don’t have proper claws.
              The hips are also designed improperly, so dragging
              the head down doesn’t push the tail up.  This would
              be a simple fix, but it would be tedious.  Next time
              maybe.
```

1.10 Benchmarks

```
                                                  Benchmarks

   Since the program doesn’t run at an acceptable speed on my machine I’d like
to collect a series of benchmarks from users with more powerful machines. In
order to make the benchmarks as useful as possible I’d like to include as much
information about each test.  Since different models take longer to render I
performed most tests with the puppet.gr model because it is the most complex.
For a demonstration of texture mapping I did a few tests with the dino.gr
model.  Since gravity and wind both function by aplying the IK system to
every jointin the model, they both take as long to compute.  Thus, all tests
with animation enabled were performed with only gravity enabled.
```

Other tests were performed with wireframe enabled, to show the effects of less area being rendered, and then with all effects turned off to see how the system performed when no calculations were required for individual pixels (ideally my graphics card could do the rendering at this stage). Since the program does not display the window size the only two sizes used were the default (300x300) and the smallest size (100x100).

On slow machines moving the mouse can actually affect the frame rate, and on some video cards and screenmodes (without a hardware sprite) having the mouse over the window can slow down rendering as well. In modes that are 16 bit and lower, dithering is enabled which slows down the system. On my machine the dithering time almost accounts for the extra data to be manipulated in 16 bit modes.

In order to keep the model moving as fast as possible I enabled rotate animations and gave the model a spin with the menu button in "Position Model" mode. After at least 10 frames have been rendered, I record the frame rate. If you collect some benchmarks you can
mail
them to me along with the
settings you used and I will include them in this document.

I would also appreciate comparisons of the 040 and 060 versions with the generic versions on the appropriate processors, so I can see if they really make a difference.

The results so far:

Model	Machine	Processor	Window	Effects	Graphics Mode	FPS
puppet.gr	A3000	030/25	100x100	A-LFDS	PicassoII 24Bit	0.5
puppet.gr	A3000	030/25	100x100	--L-DS	PicassoII 24Bit	0.7
puppet.gr	A3000	030/25	100x100	-WL-DS	PicassoII 24Bit	0.8
puppet.gr	A3000	030/25	100x100	-W----	PicassoII 24Bit	1.2
puppet.gr	A3000	030/25	300x300	A-LFDS	PicassoII 24Bit	0.3
puppet.gr	A3000	030/25	300x300	--L-DS	PicassoII 24Bit	0.4
puppet.gr	A3000	030/25	300x300	-WL-DS	PicassoII 24Bit	0.5
puppet.gr	A3000	030/25	300x300	-W----	PicassoII 24Bit	0.7
puppet.gr	A3000	030/25	100x100	A-LFDS	PicassoII 16Bit	0.5
puppet.gr	A3000	030/25	100x100	--L-DS	PicassoII 16Bit	0.8
puppet.gr	A3000	030/25	100x100	-WL-DS	PicassoII 16Bit	0.8
puppet.gr	A3000	030/25	100x100	-W----	PicassoII 16Bit	1.2
puppet.gr	A3000	030/25	300x300	A-LFDS	PicassoII 16Bit	0.3
puppet.gr	A3000	030/25	300x300	--L-DS	PicassoII 16Bit	0.4
puppet.gr	A3000	030/25	300x300	-WL-DS	PicassoII 16Bit	0.6
puppet.gr	A3000	030/25	300x300	-W----	PicassoII 16Bit	0.8
puppet.gr	A3000	030/25	100x100	A-LFDS	PicassoII 8Bit	0.5
puppet.gr	A3000	030/25	100x100	--L-DS	PicassoII 8Bit	0.7
puppet.gr	A3000	030/25	100x100	-WL-DS	PicassoII 8Bit	0.8
puppet.gr	A3000	030/25	100x100	-W----	PicassoII 8Bit	1.2
puppet.gr	A3000	030/25	300x300	A-LFDS	PicassoII 8Bit	0.4

```
puppet.gr   A3000    030/25    300x300 --L-DS  PicassoII 8Bit       0.5
puppet.gr   A3000    030/25    300x300 -WL-DS  PicassoII 8Bit       0.7
puppet.gr   A3000    030/25    300x300 -W----  PicassoII 8Bit       0.9

dino.gr     A2000    030/25    300x300 A-LFDS  PicassoII 16Bit      0.1
dino.gr     A2000    030/25    300x300 --L-DS  PicassoII 16Bit      0.2
------------------------------------------------------------------------
puppet.gr   A2000    040/40    100x100 A-LFDS  PicassoII+ 16Bit     3.9
puppet.gr   A2000    040/40    100x100 --L-DS  PicassoII+ 16Bit     5.2
puppet.gr   A2000    040/40    100x100 -WL-DS  PicassoII+ 16Bit     7.6
puppet.gr   A2000    040/40    100x100 -W----  PicassoII+ 16Bit     8.2

puppet.gr   A2000    040/40    300x300 A-LFDS  PicassoII+ 16Bit     2.0
puppet.gr   A2000    040/40    300x300 --L-DS  PicassoII+ 16Bit     2.6
puppet.gr   A2000    040/40    300x300 -WL-DS  PicassoII+ 16Bit     3.1
puppet.gr   A2000    040/40    300x300 -W----  PicassoII+ 16Bit     4.1
========================================================================
Effects: A = Animation (gravity/wind)   W = Wireframe Mode
         L = Lighting   F = Fog   D = Depth Buffer   S= Smooth Shading
========================================================================

Thanks to Haflinger for A2000 040/40 results.
The A2000 030/25 results are courtesy myself, since the SCSI in my
 A3000 died.
```

1.11 Porting To CyberGL

Porting To CyberGL

Since one of the reasons I undertook this project was to evaluate CyberGL I thought I'd list my comments on developing for the library here. I would like to state that this is not a formal review, as my project only touched on a few features of the library, and I have not kept up to date with all developments with the library, and have submitted very few bug reports since my net access is currently expensive. Thus if I mention anything that has been fixed or changed in CyberGL, my apologies to the authors.

I downloaded cybergl39_9dev.lha which contained version 39.8 of the CyberGL library, despite the fact that the file Changes.txt refers to version 39.9 of the library. The library on my machine is 39.12. (I think it comes from a user archive). The headers included in the archive did not define the tag GLWA_buffered which was documented as belonging to version 39.9 in Changes.txt, so I assume the headers were also from version 39.8. I didn't realize the library would do it's own buffering so I wrote my own buffering system before discovering the tag in Changes.txt. Since no value was given I defined it to be 1 more than the other tags that were defined in the header. With this modification to the header double buffering worked perfectly, but I wasted a bit of time because of the disorganization of the distribution.

The headers were not organized into a directory structure like normal Amiga includes (and the CybergraphX includes), so I installed them myself. The demo code included the headers from the local directory (with "" instead of <>) which indicates they had not yet been added to the standard include path. Another easy fix, but the overall impression was that the developer package was rushed together (kind of an alpha release). The lack of documentation was frustrating since I couldn't easily discover exactly how the Amiga specific calls worked, and on what systems/screenmodes the system would run. The demo code also consistently failed to open the cybergl.library, in many cases hanging the machine. (This only happened when I compiled the demos. They may have been compiled with different link libraries, but I used the supplied smakefile). In the future a real development archive is strongly advised.

Once these problems were ironed out things got much easier, since I was using code initially compiled and tested on OpenGL. A few routines in the glu library had changed names and entered the gl library (this was rather fortunate since there is no glu library with CyberGL, just cybergl.lib which only has a few functions which are not documented). There were a few functions that did not have their parameters defined as const, but I'm not sure if this would be correct or not. Otherwise my code compiled perfectly and I was able to render a model after just a few hours, most of which was spent translating Tcl code into C.

The biggest part of porting the project was then cleaning up my code which had grown in a convoluted way as I worked constantly against a deadline without sleep. I had written A3 in C, because I never fully trusted g++, and started writing the project in C with a C++ structure. This became a mess, and I switched to C++, which resulted in code that was a mix of object oriented and complete chaos. Furthermore everything except the Jacobian and interface stubs was in one massive (1750 lines) file. I completely re-organized the code for the Amiga version, so that it could be more useable. I also had to finish porting

CPgui

to the Amiga (the original version was for the Macintosh and the Amgia version had few features implemented). During this phase of the project I had very few problems with CyberGL, and made almost no modifications to the original OGL code (except to clean it up and improve efficiency).

Features

Because of the lack of the glu library I had to replace the code which rendered my spheres. The glLineWidth function was not implemented which makes my wind vector a little anemic. The most surprising omission however was the lack of display lists. Display lists give the most improvement in a networked environment, but it seems they would make a difference when running the library on a PPC card and an application on a 68K processor. (Shared memory notwithstanding). Furthermore display lists make rendering primitives simpler because calculations can be eliminated, and could be essential when porting code from other environments. I was able to work around the omission easily, but it seems like an obvious feature to implement.

Bugs

I noticed a few refresh bugs when rendering into a GL window obscured by another window, but none were serious. I got an enforcer hit within the CyberGL library when I tried to open a window on an ECS screen, but 8 bit seems like the minimum config for CyberGL (and it doesn't look good unless you're in 16 bit mode). The only serious problem was in the picking routines. The select buffer returned when in GL_SELECT mode didn't have depth information for the nodes that were hit. This meant that if multiple hits occurred I couldn't differentiate between them and determine which object was closest. All records in the pick list had a min and max z value of -1. Aside from these problems the library performed perfectly and rendered images that were superficially identical to those rendered on the Iris.

Speed

Puppeteer is certainly not the most efficiently coded OGL application, but the Amiga version is much more efficient than the SGI version which did run at an acceptable frame rate, so the code should be able to run well on a fast enough machine. I may port the Amiga version back to the SGI so that meaningful speed comparisons could be made with the new code. I may also write a version using Amiga Mesa so that I can do some comparisons (though I believe Mesa for the Amiga was dog slow). To improve the value of test results I added an Effects menu to disable most OGL rendering modes so their impact on performance could be evaluated.

When rendering Gouraud shaded, depth buffered polygons I expected the library to be slower than other 3D applications, since each pixel would have to be examined. When rendering flat shaded wireframe graphics I found the library was still slow. It seems that the speed could be improved for trivial cases, but the authors probably feel that their time is better spent on writing drivers for cards or the PowerPC (I hope).

Support

I have currently only submitted one bug report based on Puppeteer, but following this release I will be submitting several more. Response to previous bug reports and the release of Puppeteer has been very prompt, and the authors of CyberGL have been helpful, and communicative. While the first bugs have not yet been fixed, I have been informed that they are being looked at.

Summary

For my purposes CyberGL provided a reasonably accurate implementation of OpenGL with most necessary features. The rendering quality was excellent, and produced results that were comparable to those on high end workstations. For porting software from other platforms it might be desirable to expand the feature set. The speed however, was a bit slow. While I can appreciate that more time should be devoted to developing drivers for video cards, if any software is to be developed that uses the library the software emulation will need to be as fast as possible due to the small number of video cards out there. In my opinion the library should be able to perform well on at least an 040 if some of the more time consuming options are disabled.

1.12 Bugs glitches and other snafus

Bugs Glitches and Other Snafus

Puppeteer currently has a few bugs:

1. If two primitives lie one on top of the other, picking the top primitive may select the lower one. This is due to a bug (shortcoming?) in CyberGL.
2. Once texture mapping has been enabled it cannot be disabled. This means selection looks a little weird, and loading a non texture mapped

```
      model after a texture mapped model has been displayed doesn't look
      right.  This appears to be due to a bug in CyberGL.
   3. The normals for mesh primitives are not calculated correctly.  This
      makes long meshes appear shadowed when they should not be.
   4. I'm not sure if the trackball is too fast or not.  I can't really
      see it in real-time on my machine.  The same goes for animations.
      It all worked smoothly on the SGI, but I've added a lot of
      features since then that have not been appraised visually.
   5. Mouse input and animation are done using intuiticks.  This
      effectively limits the top speed of the animation to around 10 fps.
      For me this is not a problem, but I would hope it is to slow for
      somebody.  If you feel you could get me more fps,
                contact
                      me or make your window larger :)
   6. I have had one spurious crash I never identified.  I never managed
      to reproduce it, so I don't know if it is still there.
   7. I haven't tested the program on different machines or configurations
      so I don't know if all the library versions are lenient enough,
      the text offsets are calculated correctly, etc... I also haven't
      tested with different configurations, processors, system versions,
      etc...
   8. Currently there is no way to control the tiling of texture maps.  This
      makes them distorted in certain parts of the model.

   If you discover a bug you can characterize, drop me a
                line
                .  I
can't guarantee I'll fix it, because like I said, this program does nothing
useful.  I will try to incorporate the fix into future releases if possible.
```

1.13 Things to do

```
                            Things To Do
                      (Numbered But Not Ordered.)

1. Implement variable damping factors to be able to set the springiness
   of nodes.
2. Fix prismatic joints.
3. Clean up the old code a bit.
4. Create a version that uses hardware (screen) double buffering.  On my
   machine this would make a huge difference.  On a machine that can
   actually run this it probably not be too significant.
4. Implement animations faster than intuiticks.
5. Create a walking object.
6. Create an object editor.
7. Create a link to the ray tracer I wrote in CS488.  This should be easy
   since I've allready ported enough of the TCL interpreter to handle the
   scene files used by the ray tracer.
8. World peace and resurrection of the Amiga.
```

1.14 Revision History

```
                              Revision History

Version 1.2 (Second Public Release) Feb 16 1998
   · Implemented Hierachical meshes.  These meshes bend with the joints
     contained within them.  They are a bit of a hack, however.
   · Implemented mesh primitives.  These are solid triangle meshes defined
     via contours.
   · Made the material class into a proper object with methods and private
     members.  Added texture mapped materials.
   · Expanded the command parser to be able to handle nested parentheses and
     commands with variable numbers of arguments.
   · Added a default value for a joint when setting it’s constraints.
   · Created the dino object to demonstrate the new features.  Also created
     the slug and worm objects.
   · Added fast move option to animate menu to enable wireframe while the
     model is being manipulated.
   · Reduced stack usage.
   · Compiled versions of Puppeteer for 040 and 060 processors.
Version 1.1 (First Public Release) Jan 31 1988
   · Moved all code to C++.
   · Rewrote TCL/Tk code as C++, and replaced TCL interface with CPGui code.
   · Created object loader class to take place of TCL for parsing models.
   · Removed spider model because it depended on TCL functions.
   · Re-implemented orientation of model with virtual trackball.
   · Moved selection to first mouse button, and allowed selection in both
     mouse modes.  Now the interface is the same in both mouse modes.
   · Added effects menu to control the depth buffer, wireframe mode, gouraud
     shading and lighting.
   · Added wind vector to display menu.
   · Implemented open inventor style animated rotations.
   · Added frame rate counter to window.
Versopm 1.0 (CS488 Final Project) April 1 1997
   · Implemented generic loader that could load arbitrary models.
   · Created rope, chain and spider models.
   · Implemented joint nodes that can be manipulated using inverse
     kinematics.
   · Moved most of primitive functions to C++.
   · Replaced pseudo gravity with downward force applied to all joints using
     the inverse kinematics system.
   · Added wind animations.
   · Modified selection code to disallow multiple nodes being selected in the
     path to the root.
   · Added display menu to enable/disable rendering of joints and primitives.
   · Removed orientation of model from user interface.
   · Combined x and y translations into the second mouse button.
   · Moved selection to the third mouse button.
   · Changed mouse mode 2 to translate and rotate joints.
Version  0.9 (CS488 Assignment 3) March 3 1997
   · Implemented hierarchical model display of Puppet.gr model. Supported
     primitives are:
        cube
        transform
     All primitives can be rotated, scaled or translated to define a model.
   · Implemented selection and manipulation of joints using forward
     kinematics.  Button 1 selects joints, while dragging button 2 changes
     the value of selected joints.
```

- Dragging button 3 rotates the head.
- Implemented manipulation of models position using mouse. Dragging with button 1 translates along X axis, button 2 along Y axis and button 3 along Z axis.
- Implemented manipulation of models orientation using mouse and virtual trackball.
- Implemented Cylon eye Bounce animation.
- Implemented pseudo gravity animation, based on projection of normal vectors onto axis planes. While fast, this aproach never worked properly, and "down" would always be the closest axis.

1.15 Not So Legal Junk

Disclaimer

This bit just says that if Puppeteer corrupts your hard disk, your files, or your life I’m not to blame. In other words, you use Puppeteer at your own risk.

1.16 Not So Legal Junk

Disclaimer

This bit just says that if Puppeteer corrupts your hard disk, your files, or your life I’m not to blame. In other words, you use Puppeteer at your own risk.

1.17 Easter Egg

EASTER EGG

I was too lazy to put any Easter egg in the program so here’s one in the Docs. If you stripped the ASCII out of the amigaguide you’ll find this quite easily. Otherwise you either got lost or had too much time on your hands.

Feel free to skip, disparage, discard or otherwise ignore this. Since only the bored, curious, or otherwise distracted will read this, here is where I can put all of the inappropriate doc things, such as gratuitous ASCII art:

```
       ,___.
     gW@@8c~+s
   ,W@@W8/~, ’N.
   W@@4@G_t-  ’W
  d@WA@Wbg!-,  !b
 i@A@@@4Mt(` -  Yi
 ]@@@W@WA4\/,   ][
 @@W@W@D*Z+Nm_.  @
]P8b i@W [-@@-!@KY[ <- Easter Egg
]b8Kmm@WzWmW+==*Ld[
 M@@8@K@A[-f -’ ,A
 ]@@@WA8Z,D’ .- ][
                            '@@@W@@D]+!    W`
                             !@@A@WAN_' ' d!
                              Y@@@MGG/ ' iP
                               V@@@Wb.! gf
                                'VM@W_Df`
                                ~*@@@Af`

NOTE: This Easter Egg has already appeared in the documentation for ChunKit.
```

1.18 About the Author

```
                                        Contact Information

   I am a recent graduate of the University of Waterloo with a BMath in
computer science.  I live in Stouffville Ontario Canada, near Toronto. If you
want to give me a job writing this kind of stuff (or other stuff) here is my

                resume
                .

   I can be reached by email at

               progers@undergrad.math.uwaterloo.ca

for at least six months, but I'll be getting a new email address in the
next few weeks.  (Undergrad will be forwarded to me).

   If you insist on using snail mail my current address is:

               Patrick Rogers
               49 Albert St.
               Stouffville Ontario, Canada
               L4A 1B5
```

1.19 Resume

```
                                Patrick Rogers

                  49 Albert St., Stouffville Ont., N2L 3W6
                              (905)640-8652
                    progers@undergrad.math.uwaterloo.ca

Summary of Qualifications:

   · Experienced in independent software development.
   · Working knowledge of C++, C, 680X0, X86 and 6502 assembler,  ADA,
     Modula-3, PASCAL, Matlab, Turing, BASIC and GL/OGL.
   · Extensive knowledge of programming and usage of graphical user interfaces
     including X-Windows,  Macintosh and Amiga.
   · French Canadian citizen, fluent in both spoken and written French and
     English.
```

```
Education:

   University of Waterloo, Ontario..................................Fall 1997
      Bachelor of Mathematics Degree in Honors Computer Science.
      Relevant Courses:
        Computer Graphics, Real-Time Programming, Artificial Intelligence,
        Computer Algorithms, Computer Theory, Software Abstraction, Sequential
        Programming, Concurrent Programming, Operating Systems, Data
        Abstraction and Algorithms, Digital Design, Scientific Computation
   Northern Summer School for excellence in science...............Summer 1990
   Dale Carnegie course in human relations........................Winter 1989

Work Experience:

   Gemsoft Inc. Toronto, Ontario. Employer: Mark Vange (416) 368-1552
      Software Developer.......................September 1995 - December 1996
      ·  Designed and programmed Macintosh version of H.R. Giger screen saver.
      ·  Created installer and localization package for H.R. Giger screen
         saver.
      ·  Worked on Macintosh version of 3D flight simulator.
      ·  Began port of network game client from Windows95 to Macintosh.

   Apple Canada, Markham, Ontario. Supervisor: Denis Shelston (905) 513-5910
      Computer Technician...........................May 1993 - November 1993
      ·  Handled  technical support calls from end users across Canada.
      ·  Answered queries from end users on Ask Apple bulletin board.

Awards:

   ·  Inverse kinematics project selected one of top six graphics projects
      for the winter term, 1997, University of Waterloo.
   ·  Ontario Scholar.
   ·  Scored 47 on Descartes Mathematics contest (Top 25% of participants).

Interests And Activities:

   ·  Have written and released public domain software, released on Aminet 6.
   ·  Operating System Research.
   ·  Artificial Intelligence
   ·  Computer Programming.
   ·  Electronics/Robotics.
   ·  Reading and Creative Writing.

References:

   ·  Dr. Michael D. McCool, Professor
      Computer Science Department, University of Waterloo
      (519) 885-1211 x4422        mmccool@cgl.uwaterloo.ca
```