CPK v2.2

A program to render space-filling molecules on the Amiga
Edition 1.0, for CPK Version Two
December 1993

by Eric G. Suchanek, Ph.D.
E-mail: esuchanek@bix.com

Legal/Disclaimer

You may use this program freely for *non–commercial use only*. You may also distribute this program and its associated support files so long as it is not “For Profit”, and as long as all files in the distribution remain intact. Fred Fish is expressly given permission to distribute this code in his disk collection.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. Use at your own risk!

CPK displays molecules. It does not crash my machine, nor do I get `Enforcer` or `Mungwall` hits with it (see Section A.1 [Known Limitations], page 22, for an exception). It also does not "leak" memory and should gracefully exit and clean up in the event of a fatal error. Alas, this program will not solve the world’s problems nor will it balance the federal budget.

I consider this program to be `CharityWare`. If you like it, please consider sending a donation to the National SIDS foundation. You can read more about SIDS in the About.SIDS document included in this distribution.

If you have any 3D solid modeling algorithms with shading I’d love to see them, (especially some 3D cylinder shading routines).

I am unable to distribute the CPK source code except to licensed Amiga developers, since the IFF routines are developer beta and can not be generally distributed - sorry :-(

Dedication

I dedicate this program to my late son George Kelly Suchanek. His life was short, but while he was with us he showed me what is really important…

Requiescat in Pace.

1 What is CPK?

CPK is a program that renders space-filling representations of atoms in molecules. This is the type of representation one would find in the plastic 'CPK' (Corey, Pauling, Koltun) models often used in organic chemistry. The program is AmigaDos V 3.x specific, and has no hard-coded constraints in the number of atoms it can process. Unlike many programs of a similar nature, CPK correctly handles intersecting 3-dimensional spheres by using the famous Bresenham circle algorithm in 3D. In order to keep the program simple and reasonably fast I do not supersample the spheres, so their resolution is essentially equal to the display screen you're using at the time.

That's all well and good, but what does it really mean? Well, simply put it means that the sphere's surface gets increasingly inaccurate as your resolution goes down. Don't be surprised to see the image quality degrade as the screen resolution drops. See Chapter 3 [Tips], page 21, for a workaround.

The molecular format used is called `Protein Data Bank` (.pdb) format. This is one of the most prevalent formats used in modern chemistry. The entire Protein Data Bank consists of a few hundred protein and DNA molecular structures and can be obtained from the Brookhaven laboratories for a nominal fee (or from me if you know what you want).

CPK is also capable of parsing various `annotation` records in the source file. This information typically describes the structure itself, the author, resolution, crystal parameters and secondary structure of the molecule. This information, if present, is accessible through the Annotation Window menu item.

Finally, through my new `pdb2pov` program it is possible to generate a `POV-Ray` version 2.0 compatible scene file and actually invoke `POV-Ray` from CPK. See Section 2.6.2 [POV-Ray], page 14, for more information.

1.1 Distribution kit contents

The distribution should contain:

- '`CPK.000`' - Generic version which should run on all Amigas.
- '`CPK.030`' - 68030 specific version.
- '`CPK.040`' - 68040 specific version.
- '`pdb/cube`'.pdb - A small file to test scaling, aspect ratios, etc.

- '`pdb/nati2.pdb`' - An actual protein molecule. This protein is used in some modern detergents as an enzyme which digests proteinaceous stains.
- '`pdb/hello.pdb`' - A whimsical "molecule".
- '`pdb/heme.pdb`' - The heme group from myoglobin.
- '`pdb/zna2b.pdb`' - A Z-Dna molecule.
- '`pdb/lysozyme.pdb`' - The enzyme lysozyme.
- '`pdb/znf2.pdb`' - A modeled zinc-finger domain from the AIDS virus.
- '`pdb/mbn4.pdb`' - Myoglobin
- '`pdb/*.pdb - More`' atomic files as I acquire them.
- '`prefs/*.prefs`' - Various preference files of settings.
- '`docs/cpk.guide`' - CPK documentation in AmigaGuide format.
- '`docs/cpk.texinfo`' - Texinfo formatted documentation.
- '`docs/cpk.dvi`' - Device–independent TEX formatted documentation.
- '`docs/cpk.ps`' - Postscript version of the CPK documentation.
- '`docs/pdb2pov.doc`' - Describes the `pdb2pov` converter program for the `POV-Ray` interface
- '`docs/About.SIDS`' - Information about SIDS.
- '`prefs/*.*`' - Various preference files and palettes.
- '`util/*.*`' - Various utility programs for the interface to POV-Ray.
- '`include/*.inc`' - Various `POV-Ray` include files needed for the CPK to `POV-Ray` interface.
- '`Scenes/*.pov`' - Some example scene files for `POV-Ray` v2.0 created by the `pdb2pov` converter.
- '`Arexx/test.cpk`' - `ARexx` script to test the CPK - `ARexx` interface.

1.2 What you need to run CPK

This program was written for Version 3.X of the Amiga Operating System! It will no longer run under Version 2.X!

CPK should run on any Amiga with at least 1MB memory. It tries to be reasonably tolerant of low memory conditions, and should gracefully degrade if it can't get memory. The program has become something of a memory hog in the version 2.x series, since I now cache some of the surface displacement arrays. As a result, I recommend having at least 2MB of RAM to run this program comfortably.

This program also requires Nico Francois' `reqtools.library`, which is available on many public BBS. I currently use `reqtools.library` V38.1042.

Finally, if you wish to use the new `POV-Ray` interface you will need a copy of `POV-Ray` version 2.0 or higher. This is an excellent public domain ray-tracer which may be found on a number of BBS and internet sites.

I have customized `POV-Ray` to be slightly more compatible with this version of CPK. My version also has direct support for the Picasso II RTG 24 bit graphics board. This version is available on aminet and BIX. The name will be '`povami_picasso_xx.lha`', where xx represents my internal version.

1.3 Installation

Drag the CPK drawer to some appropriate location on your system. Keep the directory hierarchy intact, though.

POV-Ray users take note:

If you wish to utilize the interface I've created to `POV-Ray` v2.0, you must copy all files from the include directory to your normal `POV-Ray` include area.

2 Using CPK

2.1 Running the Program

To launch the program, double click on the icon appropriate for your machine:

- 68000 - CPK.000, (IEEE math emulation).
- 68030 - CPK.030, (with math coprocessor).
- 68040 - CPK.040, (with math coprocessor).

The program should start and immediately begin rendering the structure defined in the *SETTINGS* tooltype. See Section 2.5 [ToolTypes], page 12, for more information on this tooltype. The preference file stores various program settings like window positions, the name of the atomic structure file, and the screen size and display mode. If the program can't find the particular screen mode present in the preference file, you will be prompted to select another display mode. This is likely to happen (unless you have a Picasso II RTG board), since most of the preference files were saved at a resolution of 1152x900x256. Once you choose the new display mode the program should open the appropriate screen and render the molecule. At this point you can manipulate the structure and windows. In order to make your changes permanent, select the `Save` menu item.

2.2 A Quick Example

You've now launched the program and rendered a structure. How do you view a new structure? This is easily accomplished in a few simple steps:

1. Click on the `Load...` gadget in the Section 2.3.2 [Molecular Orientation Window], page 7.
2. Then, select a PDB file from the ASL requester.

The program will read the selected molecule and immediately render it. You are now free to adjust window positions and sizes, orientation and scale until you get the desired view. When you're satisfied select the `Save As...` menu item under the Section 2.4.1 [Project Menu], page 9. This will save the current program state (screen mode, window positions, sizes and structure filename) to the specified preference file. Whenever you wish to restore this view, simply reload the preference file by using the `Open...` menu item in the Section 2.4.1 [Project Menu], page 9.

2.3 The CPK user interface

The CPK user interface consists of several windows:

2.3.1 Molecule Window

This window contains the currently defined molecule. Be aware that whenever you *resize* the window the program will perform the following operations:

- Clear the window
- Re-center the molecule based on the current window dimensions
- Re-render the molecule

The title of the current preference file is also shown in this window.

2.3.2 Molecular Orientation Window

This window contains the gadgets that let you manipulate the orientation and the scale of the currently defined molecule. These are described below:

X, *Y*, `and` *Z* `sliders`
: These control the rotation angles about the *X*, *Y*, and *Z* axes, respectively.

Scale `slider`
: This gadget governs the overall size of the molecule.

File `string gadget`
: This gadget allows the user to specify the complete path to an atomic structure file.

Load... `button gadget`
: This gadget, when pushed, brings up the ASL file requester to load an atomic structure file.

Reset `button gadget`
: When pushed this gadget resets the current molecule's scale and rotation angles.

Auto Render `checkbox gadget`
: When *Auto Render* is checked, any manipulation of the rotation or scale gadgets will force a re-render. This can be annoying when you want to set a number of angles and/or scale. By turning *Auto Render off* rendering *only* takes place when the molecule window is resized or the *Render* button is pressed.

Render `button gadget`
: When pressed forces a molecular re-render based on the current orientation and scale.

2.3.3 Light Source Window

This window controls the positioning of the point light source used to illuminate the molecule.

X`,` *Y*`, and` *Z* `sliders`
: These three sliders for X, Y and Z represent the direction vectors for the light source. X and Y can vary from -100 to 100, Z from 0 to 100. The Z direction is out of the plane of the screen in this program. These are the direction vectors times 100 (or percentages).

OK `button gadget`
: This forces the program to re-read the slider positions, create a new, normalized light source, and re-render the molecule.

Done `button gadget`
: Closes the window

For example, if X were 0, Y were 0 and Z 100, the light source would be pointing along the Z axis. Simply set the values you want and hit the OK button. The program will normalize the values for you and re-render the molecule. This works best when the QuickRender flag is turned ON. Select the `Quick Render` menu option (Section 2.4.4 [Misc Menu], page 10), and try moving the light source around.

2.3.4 Animation Control Window

This window allows the user to specify a frame sequence of IFF files for use with post-processing programs. I use DPAINT IV to load the individual frames into an animation, and manipulate them from there. By using the images as an AnimBrush it's possible to create some nice animations. The gadgets are described below:

`Frame Prefix:`
: Specifies the complete path prefix for the anim frames. It should end in a period. For example: ram:mol_anim. The program will add a numerical suffix like 001 to this prefix in creating the file name.

`Start Frame:`
: Frame number the program should use when starting the animation. (Zero based).

`End Frame:`
: Frame number to use for the ending frame. (Exclusive).

`Total Rotation:`
: Total rotation in degrees for the animation.

`X, Y, Z boxes:`
: If checked, will rotate about this axis.

  NOTE: This is set up for Loop style animations. That is, if you have selected, start 0, end 10, 360 degrees rotation set, the program will compute frames 000 - 009 but you'll only rotate 360 - 360/10 degrees. This means that cyclic loop animations will not hesitate on the first frame. In essence, you're doing one less frame, since the frame counter is 0 based.

2.4 The CPK menus

2.4.1 Project Menu

The Project Menu controls preference file I/O as well as providing some informational messages about CPK and the `pdb2pov` conversion program.

`Open...` Open a preference file. Try to Open one after launching the program. Hello.prefs is interesting.

`Save` Save the current X,Y,Z rotations, scaling, window sizes and positions to the currently named 'preferences' file.

`SaveAs...` Save the current X,Y,Z rotations, scaling, window sizes to a named 'preferences' file. (Right Amiga A).

`Print` Print the molecule window to the printer. Uses default WB prefs. No abort gadget yet. Print if you're serious about it! (Right Amiga P).

`Save Window as IFF...`
: Save the Molecule Window to an IFF ILBM format file using a user-specified name. (Right Amiga I).

`About CPK...`
: Informational window about the current release of CPK. (Right Amiga ?).

`About pdb2pov...`
: Informational window about the current release of the `pdb2pov` conversion program

`Quit...` Quit CPK. (Right Amiga Q).

2.4.2 Screen Menu

The screen menu provides tools to set the screen mode, number of colors and color/grayscale rendering options.

`Screen Format...`
: Brings up the ASL screen mode requester to select a screen mode. You must select a display with at least 4 bitplanes of color. I recommend a 1024x768x256 display if possible. This mode is saved in the settings file as well as window positions and sizes.

`Color Mode`
: Check either color or grayscale rendering mode.

`Palette...`
: Opens the palette editor window. You currently cannot save color palettes. (Right Amiga T).

`Photo Mode`
: Converts the molecule image window into a *borderless* window which fills the entire screen. Now, when saving IFF images you'll no longer see the window borders, and the image should be a more 'standard' size.

2.4.3 Animation Menu

The animation menu allows the user to set up multiple frame renderings to IFF files. These images may be combined by using a variety of post-processing tools to create animations.

`Anim Setup...`
: Opens the animation setup window, which is described in Section 2.3.4 [Animation Control Window], page 8. (Right Amiga M).

2.4.4 Misc Menu

The Misc menu consists of menu options to display the annotations window, set the rendering mode, atomic radius scaling factor and the saving of program icons.

`Annotations Window`
: Brings up a window to show the various 'annotations' which may be present in the PDB file. Things like the compound name, structure author, journal references and crystallographic parameters are shown. Just click on the cycle gadget to browse the

annotations. Most of the files included don't have any annotations present. (Right Amiga N).

`Light Window`
: Brings up the light source positioning window. See Section 2.3.3 [Light Source Window], page 8, for a description of this window and its gadgets. (Right Amiga L).

`CPK Scaling`
: This menu option brings up a window which lets you specify the atomic radii scaling factor. A factor of 1.0 gives the normal CPK style rendering, while factors < 1.0 make smaller spheres. This can be useful when rendering structures like DNA which have a regular internal structure. Factors > 1.0 make more of a *van der Waals* style rendering. (Right Amiga C).

`Quick Render:`
: When set this menu option disables some of the specular highlight calculations (which use the transcendental power function). On unaccelerated machines this can have a noticeable effect on rendering times. Unfortunately in Quick Render mode the program suffers from pretty severe color banding.

`Save Icons:`
: When set, this menu option saves a nifty icon with IFF files.

2.4.5 `POV-Ray` Menu

This menu controls the interfaces to the `pdb2pov` conversion program and the `POV-Ray` ray-tracer. See Section 2.6.2 [POV-Ray], page 14, for more information about these topics.

`Make Checkerboard:`
: Enables the creation of the (in)famous black & white checkerboard ground.

`Make Sky:`
: Enables the creation of a nice blue cloudy sky.

`Use Arealights:`
: Turns on the use of area lights for soft shadows rather than a single point light. These scenes take much longer to render!

`Use CPK radii:`
: Selects CPK radii for atom scaling in the ray-traced scene.

`Use VDW radii:`
: Selects van der Waals radii for atom scaling in the ray-traced scene.

`Use Covalent:`
: Selects covalent radii for atom scaling in the ray-traced scene.

`Ball and Stick mode:`
: Renders the molecule in ball and stick mode. This takes a lot longer to render but looks really good.

`Ball & Stick & Glass mode:`
: Renders the molecule in ball and stick mode, with glass VDW atoms overlaying them. This takes much longer to render than ball and stick mode.

`Bond distance cutoff:`
: Allows the user to select the distance above which a bond will not be drawn. Only relevant in ball and stick mode. The distance is specified in Angstrom units.

`Run Pdb2pov:`
: This runs the `pdb2pov` converter program to generate a scene file based on your current selections. It does not run the ray-tracer.

`POV-Ray environment variable`
: This allows the user to set/change the normal '`POVRAYOPT`' environment variable. See your `POV-Ray` docs for more info. I recommend turning off the verbose option with `-v` to minimize textual output from `POV-Ray`.

`POV-Ray program path`
: This allows you to set the path to your version of the POV-Ray program. You must do this at least once in order to run the ray-tracer from CPK. The program actually sets the environment variable '`POVRAYPATH`' through this option.

`Run POV-Ray`
: Takes all of the options listed above and runs `POV-Ray`. Automatically runs the converter `pdb2pov` first. This is run synchronously, so all input to CPK is blocked while the raytracer is being run.

2.5 The CPK ToolTypes

CPK supports the following tooltypes:

- TOOLPRI - The multi–tasking priority of the program. This allows the user to run CPK without impacting other running programs. I usually run at -1. This does `not` affect the priority of `POV-Ray` when it runs.
- SETTINGS - The preference file to open upon startup. The default is '`prefs/cpk.prefs`'. By changing this the user may customize the starting preference file to render. The program looks in the `prefs` directory for this file.
- SAVEICONS - If set to `1`, the program will automatically save icons for preference files and IFF files. If set to `0` it won't. You may still elect to save icons by selecting the `Save Icons` menu item See Section 2.4.4 [Misc Menu], page 10, for more information.

- PUBSCREEN - This tooltype will eventually let the user select a named public screen for CPK. `Currently ignored`.

2.6 The `POV-Ray` interface

The `POV-Ray` interface is the newest and one of the most exciting additions to CPK. It is now possible to render a molecule using the `POV-Ray` ray tracing program directly from CPK. This is accomplished by the new utility program `pdb2pov` found in the '`cpk/util`' directory. Versions for generic, 68030 and 68040 based Amigas are provided.

2.6.1 Creating a Ray-Traced Molecule from CPK

Assuming you have the `POV-Ray` version 2.0 distribution installed (and the CPK include files described in Section 1.3 [Installation], page 5), follow these steps to render a molecule with `POV-Ray`[1]:

1. Set the path to your specific version of POV-Ray. Do this by choosing the `POV-Ray Program Path`... menu option. You only need do this once.
2. Set your POV-Ray program environment variable *POVRAYOPT* by choosing the *POV-Ray Environment Variable...* menu option. This variable is described in detail in the `POV-Ray` user's manual. One suggestion: I keep the *verbose* option turned off with *-v*. This keeps the textual output of `POV-Ray` to a minimum. Also remember to set the *+D* flag if you want to see the image as it is being rendered. I `highly` recommend setting this flag, since `POV-Ray` will bring up an 'abort' requester allowing you to abort the rendering.
3. Set the rendering options desired for the scene. These include such features as a cloudy sky, checkerboard ground, and several molecular rendering options like ball and stick mode, CPK or van der Waals style rendering.
4. Select `Run POV-Ray` to launch the ray-tracer. CPK will automatically run the `pdb2pov` converter and launch `POV-Ray`. *Since* `POV-Ray` *is run synchronously, all input to CPK is blocked for the duration of the rendering!*

The files created by the above process are named as follows: If the preference file name is '`cpk.prefs`', the `POV-Ray` scene file will be named '`cpk/scenes/cpk.pov`'. The display file will be named '`cpk/scenes/cpk.dis`'. Remember, the files always end up in the directory '`cpk/scenes`', and are named for the preference file. The `POV-Ray` objects created will also be named for the preference file.

[1] All of these options are contained under Section 2.4.5 [POV-Ray Menu], page 11

2.6.2 Creating POV-Ray scene files from the CLI

Since this conversion process goes through the pdb2pov program it is possible to create POV-Ray scene files by running pdb2pov from the command line. The pdb2pov programs are located in the 'cpk/util' directory with extensions indicating the processor for which the program was compiled (just like the CPK program). To get the program usage message simply type the program name file with no arguments. For example:

```
1> util/pdb2pov.040

Program: pdb2pov.xxx 1.15 1993/12/20 19:59:43


USAGE: pdb2pov.040 InputFile OutputFile
        [-o object_only]
        [-t (atm file format)]
        [-s (writes cloudy sky)]
        [-g (writes plain ground)]
        [-h (writes checkered ground)]
        [-a (create area light)]
        [-v (do van der Waals radii)]
        [-c (do covalent radii)]
        [-b (do ball_and_stick)]
        [-d x.x (bond cutoff threshold)]
        [-q (ball and stick + glass atoms)]
        [-x X-Axis rotation]
        [-y Y-Axis rotation]
        [-z Z-Axis rotation]


        Example: util/pdb2pov.040 crambin crambin -s -h -b -d 1.5 -x 90
        Converts crambin.pdb to crambin.pov, writes checkered ground, sky,
        rotates the protein 90 degrees in X and renders ball and stick mode
        with a distance cutoff of 1.5 angstroms.

        Copyright © 1993, Eric G. Suchanek, Ph.D.
        All rights reserved.
```

So, to render the file 'pdb/hello.pdb' to 'scenes/hello.pov' with a nice rotation, sky and checkerboard and area lights you'd type (from the CPK top level directory):

```
1> util/pdb2pov.040 pdb/hello scenes/hello -x 35 -y 35 -z 35 -s -h -a
```

⇒
```
    Scanning  atom file <pdb/hello.pdb>... Got <39> atoms.
    Re-orienting and positioning protein.
    Writing output file <scenes/hello.pov>
```

You'd then run `POV-Ray` as usual.

2.7 The `ARexx` interface

CPK now supports `ARexx`! The port name is *CPK1*. Errors are returned in the *CPK.LASTERROR* variable. The return code will be 0 for normal returns. In general the load and save functions require a complete pathname. See Section 2.7.1 [ARexx example], page 16, for a test of all the `ARexx` functions.

The command set is listed below:

`Command Arguments`

`PORT <NULL>`
: Returns the port name in the *CPK.LASTERROR* variable.

`LOAD PREFS 'filename'`
: Loads the specified preference file.

`LOAD PDB 'filename'`
: Loads the specified molecule file.

`SAVE PREFS 'filename'`
: Saves the specified preference file.

`SAVE IFF 'filename'`
: Saves the molecule window to an iff file.

`XROT` *angle*
: Set the X axis rotation to *angle* degrees. Does NOT render.

`YROT` *angle*
: Set the Y axis rotation to *angle* degrees. Does NOT render.

`ZROT` *angle*
: Set the Z axis rotation to *angle* degrees. Does NOT render.

`ROTATE` *angleX angleY angleZ*
: Set the X, Y and Z axis rotations to *angleX angleY angleZ* degrees. Does NOT render.

`SCALE` *amount*
: Sets the overall scale to *amount*. This can range from 1 to 10. Does NOT render.

`RESET <NULL>`
: Resets the rotations and scaling to default values. Does NOT render.

`PHOTOMODE` <*ON*|*OFF*>
: Turns full borderless molecule window *ON* or *OFF*.

`CHECKERBOARD` <*ON*|*OFF*>
: Sets the checkerboard rendering for `POV-Ray` *ON* or *OFF*.

`SKY` <*ON*|*OFF*>
: Sets the cloudy sky rendering for `POV-Ray` *ON* or *OFF*.

`RADIUS` <*CPK*|*COVALENT*|*VDW*|*BALLSTICK*|*BALLSTICKGLASS*>
: Sets the atomic radii to either *CPK*, *COVALENT*, *van der WAALS*, *COVALENT*, *ball and stick* or *ball and stick* with glass radii.

`PDB2POV <NULL>`
: Runs the `pdb2pov` program using the current settings. Does not run the ray-tracer

`POVRAY <NULL>`
: Runs the `POV-Ray` raytracer using the current settings. Does not re–render in the CPK window.

`RENDER <NULL>`
: Renders the molecule in CPK using the current settings. Does not run the ray-tracer.

`QUIT <NULL>`
: Quits the program.

2.7.1 An example `ARexx` script

Shown below is an example `ARexx` script which exercises the CPK functions. This script may be found included in your distribution as `ARexx/test.cpk`.

```
/*
 * CPK ARexx communications test...
 *
 * Author: Eric G. Suchanek, Ph.D.
 * Copyright ©1993 Eric G. Suchanek, Ph.D. All Rights Reserved
 *
 * You need to run the CPK program first...
 *
 */

Options FailAt 100

Options Results

address cpk1
```

```
/*
 * Try to read the window title bar
 */

/*
   NOTE: The Port command actually sets the errorlevel to 5,
   so you have to check for an 'error' to get the port name...
*/

'Port'

if rc > 0 then say 'Port is: 'CPK.LASTERROR
else say 'Port is: 'Result

/*
 * bogus command
 */

'Junk'

if rc > 0 then say 'Error was (this should be an UNKNOWN COMMAND error) 'CPK.LASTERROR
else say 'The command worked!'


/*
"Load Prefs prefs/b_dna.prefs"
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The command worked!'
*/


"Load PDB pdb/hello.pdb"
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Load PDB command worked!'

"XRot 45.0"
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The XRot command worked!'

"YRot 35.0"
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The YRot command worked!'

"ZRot 25.0"
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The ZRot command worked!'

'Render'

/* or, more compactly, use the ROTATE command */
```

```
"Rotate 20.0 30.0 40.0"
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Rotate command worked!'

'PhotoMode ON'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The PhotoMode ON command worked!'

'Render'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Render command worked!'

'Reset'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Reset command worked!'

'Render'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Render command worked!'

'Scale 3'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Scale command worked!'
'Render'

/* valid range is 1 -> 10 */

'Scale 11'
if rc > 0 then say 'Error was (should be an Argument Error) 'CPK.LASTERROR
else say 'The Scale command worked!'

'Save IFF ram:test.iff'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Save IFF command worked!'

'Save PREFS ram:test.prefs'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Save Prefs command worked!'

/* Play with the POV-Ray menu items */

'Checkerboard Off'
if rc > 0 then say 'Error was  'CPK.LASTERROR
else say 'The Checkerboard OFF command worked!'

'Sky Off'
if rc > 0 then say 'Error was  'CPK.LASTERROR
else say 'The Checkerboard OFF command worked!'

'AreaLights ON'
```

```
if rc > 0 then say 'Error was  'CPK.LASTERROR
else say 'The AreaLights ON command worked!'

'Radius bogus'
if rc > 0 then say 'Error was (should be bad argument) 'CPK.LASTERROR
else say 'The Radius bogus command worked!'

'Radius CPK'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Radius VDW command worked!'

'Radius COVALENT'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Radius COVALENT command worked!'

'Radius BALLSTICK'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Radius BALLSTICK command worked!'

'Radius BALLSTICKGLASS'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Radius BALLSTICKGLASS command worked!'

'AreaLights OFF'
if rc > 0 then say 'Error was  'CPK.LASTERROR
else say 'The AreaLights OFF command worked!'

'Checkerboard ON'
if rc > 0 then say 'Error was  'CPK.LASTERROR
else say 'The Checkerboard ON command worked!'

'Sky ON'
if rc > 0 then say 'Error was  'CPK.LASTERROR
else say 'The Sky ON command worked!'

'Radius VDW'
if rc > 0 then say 'Error was  'CPK.LASTERROR
else say 'Radius VDW command worked!'

'PDB2POV'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Pdb2POV command worked!'

/* OK, set up a real scene to 'render' */
"rotate 45.0 45.0 45.0"
if rc > 0 then say 'Error was 'CPK.LASTERROR

/* this 'render' is not really necessary, but synchronizes the
   CPK display to our 'scene'
*/
```

```
'render'

/* run POv-Ray */

'POVRAY'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The POV-Ray command worked!'


/*
 * Quit the program...
 */


'Quit'
if rc > 0 then say 'Error was 'CPK.LASTERROR
else say 'The Quit command worked!'
```

3 Tips and hints for using CPK effectively

- Be patient. The sphere intersection code is all integer math, but it still takes a while to traverse the atom lists during the course of rendering, especially on a 68000 machine.
- Try to run the program with a fairly large scale. The output looks better with larger spheres.
- Run at the highest possible resolution. A Picasso II board really helps!
- A narrow rendering window speeds up the calculations, since the program builds its clipping rectangle based on the window inner width and only traverses that width on rendering.
- When saving IFF images, make sure the molecule image window is in front of all other windows. If you don't the IFF image might have other window overlapping regions saved as well. I've tried to fix this by calling `WindowToFront()`, but this doesn't always work since the function is asynchronous.
- When creating animations I recommend using the `PhotoMode` menu option under the Section 2.4.2 [Screen Menu], page 10, since this will remove the molecule window borders and turn it into a full–screen image. This will take longer to render, since at present the program doesn't automatically calculate the molecules image bounding box.
- In order to get the best performance and interaction with CPK, I highly recommend my version of the `POV-Ray` ray tracer v 2.0. This should be available on aminet and BIX. I've made a few small changes to the program to better integrate it with CPK. I've also used LaMonte Koop's 68040 math library, which leads to about a 25% speed improvement over the normal libraries. This version also supports the Picasso II RTG board, so you can display preview renderings in 24 bit color.
- If you wish to use the *ball and stick* mode, please be aware that the default bond cutoff is 2.0 Angstroms. It's very likely that you'll get spurious bonds unless your structure is 'perfect' with respect to ideal bond lengths. As a result, you might need to play with the *bond cutoff distance* under the Section 2.4.4 [Misc Menu], page 10.

Appendix A Miscellaneous

A.1 Known program limitations

1. No abort from printing. Do it if you really mean it.
2. No stereo viewing.
3. No wireframe viewing.
4. No launch from Tool icon.
5. No multi-project management.
6. Due to a bug in the `Picasso II` emulator there are some `Mungwall` hits that occur during rendering if you're drawing to a `Picasso` mode screen. This does not occur when using normal (NTSC, DBLNTSC etc.) screens.
7. It is not possible to pause the `POV-Ray` program after it has finished a rendering since there is no normal input stream. As a result, any displayed image will immediately disappear upon rendering completion.

A.2 Future development plans

I have a number of items in the queue for future developments of CPK. I'll try to list these in order of priority:

1. Add an interactive wire–frame rendering mode.
2. Add stereo support to CPK and the `pdb2pov` interface
3. Add support for additional atom file types
4. Add ball and stick or stick rendering
5. Add direct Picasso rendering capabilities
6. Add a browser tool capable of reading the Brookhaven CD-ROM and displaying the structure names, resolutions etc.

A.3 The Brookhaven data bank

The Brookhaven data bank is one of the principal repositories for protein and nucleic acid crystallographic structures. The lab maintains an anonymous FTP site at: `pdb.pdb.bnl.gov`, (130.199.144.1). This site has the current structure database as well as a number of useful programs. Brookhaven also distributes a quarterly CD-ROM with the latest database. Send E-Mail to `pdb@chm.bnl.gov` for ordering information.

A.4 Program History

- V1.0 - First public release, January 1993
- V1.1 - (generally an internal 'release')

  March 18, 1993

  Modified the code slightly to utilize Math040.lib ©1992 LaMonte Koop for the 68040 specific version. This has led to about a 25% (subjective) increase in performance for CPK.040.

  June 20, 1993

  Made the reading of .pdb files more robust. NOTE: Make sure .pdb files end with an END record!

  ```
     e.g.

  ATOM ....
  ATOM ....
  END
  ```

  Also, added asynchronous file reading using 2x32K buffers. This has led to a VERY large performance improvement in reading PDB files

  July 1, 1993

  Corrected a number of user-interface 'buglets' to reset gadgets and scales when reading PDB files, etc. Many small changes.
- V2.0 - October 9, 1993
  - Most notably now REQUIRES 3.X and is now AGA compatible!
  - Uses the ASL screenmode requester to choose a screen mode
  - Added complete parsing for PDB annotations and a new 'Annotations' window
  - Completely re-wrote the rendering code for use with 8bit color palette.
  - Added the Light positioning window.
  - Added the quickrender/slowrender menu option.
  - Added `ARexx`! See the file test.cpk for example command usage.
  - Added the color palette window.
- V2.1 - October 31, 1993
  - Fixed two mungwall hits (thanks to Roger Uzin).
  - Added an elapsed time string in the rendering window.
  - Many small internal changes leading to slight code size reduction.
  - Now limits maximum scale to 10 rather than 20.
- V2.2 - December 20, 1993
  - Fixed potential enforcer hit in the ARexx handling code.

- Included the `pdb2pov.*` programs to interface to `POV-Ray` v2.0.
- Added interface to `pdb2pov` programs in the util subdirectory.
- Included my `POV-Ray` include files in the include directory.
- Reorganized directories slightly; put the ARexx script test.cpk into its own directory.
- Added a bunch of options to control the pdb2pov program, like area lights, ball and stick mode, etc.
- Modified `POV-Ray` v2.0 slightly to make it more compatible with CPK. Now the program will find the CPK screen and pop the 'Abort' requester up on this screen. My version is called '`povami_picasso.lha`'. This also can display the preview images directly to a 24 bit Picasso screen. I highly recommend it if you have a Picasso II board.
- Reformatted and expanded the original documentation considerably. This document is now in Texinfo format, which enables me to build the AmigaGuide file '`cpk.guide`' quite readily. Added the new documentation files to the distribution.
- Added the new `Photo Mode` menu option under the Section 2.4.2 [Screen Menu], page 10. This now allows the user to save a full screen IFF.
- Fixed the problem with window to front/back when doing IFF saves.

A.5 Technical Info

CPK was developed over the last seven years on machines ranging from an Amiga 1000, 2000, 2500, 3000T and finally an A4000/040. My current development configuration is:

- Hardware
  - Amiga 4000/040, 14MB RAM
  - 1GB disk (various SCSI and IDE drives)
  - Picasso II RTG board
  - Syquest 44MB removable drive
  - Wangtek 150MB tape drive
  - Apple PowerCD CD-ROM drive
  - NEC Multisync 4FG monitor
- Development Software
  - GadtoolsBox 2.0
  - SAS/C 6.50
  - GNU Emacs 18.59
  - PasTeX V 3.14
  - RCS (latest version)

  - Enforcer 37.55
  - Mungwall 37.71
  - Kickstart 40.62
  - Workbench 40.35

The program has grown from a fairly simple utility to a large application with over 15,000 lines of source code. CPK features fully dynamic memory allocation and deallocation routines, so there are no hard coded constraints in the size of the molecules you can manipulate. The ultimate limitation is the amount of available RAM. The program consumes about 64 bytes per atom, so it would take roughly 64KB to read a 1000 atom file. The largest structure I have rendered was a 10,000 atom protein from the *Hemaglutinin Influenza* virus.

A.6 Acknowledgements

At this time I'd like to acknowledge several people who helped (directly or indirectly) in the development of CPK:

- LaMonte Koop for his 68040 math library
- Roger Uzin for spotting a truly ancient bug (that's not a bug, it's a feature!)
- Nico Francois for his marvelous `reqtools.library`
- The good folks at SAS for such a great compiler
- Commodore for the greatest PC ever
- My wife Maureen for putting up with the very long hours I've spent on this program. I think I'll keep her!

Concept Index

This section contains a list of general topics (or *concepts* in Texinfo parlance) derived from this document.

Function Index

This list represents the `functions` (`ARexx` or menu selections) available from within CPK.

Variable Index

This list shows the *variables* present in CPK.

Keystroke Index

This list shows the keyboard shortcuts present in CPK.

Table of Contents