Help file for the Picture project.
Ralph Gonzalez, PO Box 54, Newark, DE 19711, USA.

## PICTURE LIBRARY

Picture is a free class library for displaying and animating three-dimensional color wire-frame drawings, although it may also be used for displaying two-dimensional drawings. It supports (1) segmentation of graphical entities, (2) multiple viewpoints and multiple projection windows, and (3) animation with an independent frame of reference for each (nested) segment. It was written with the object-oriented Think C 5.0 compiler by Symantec Corp., using Macintosh computers. However, complete portability of Picture-based applications is possible to those environments for which a Screen class has been written. Presently Screen classes for Macintosh/Think C, PC-compatible (EGA/VGA)/Borland Turbo C++, and Unix/X Window/gnu C++ combinations exist. Few changes should be needed to use these classes with different C++ compilers.

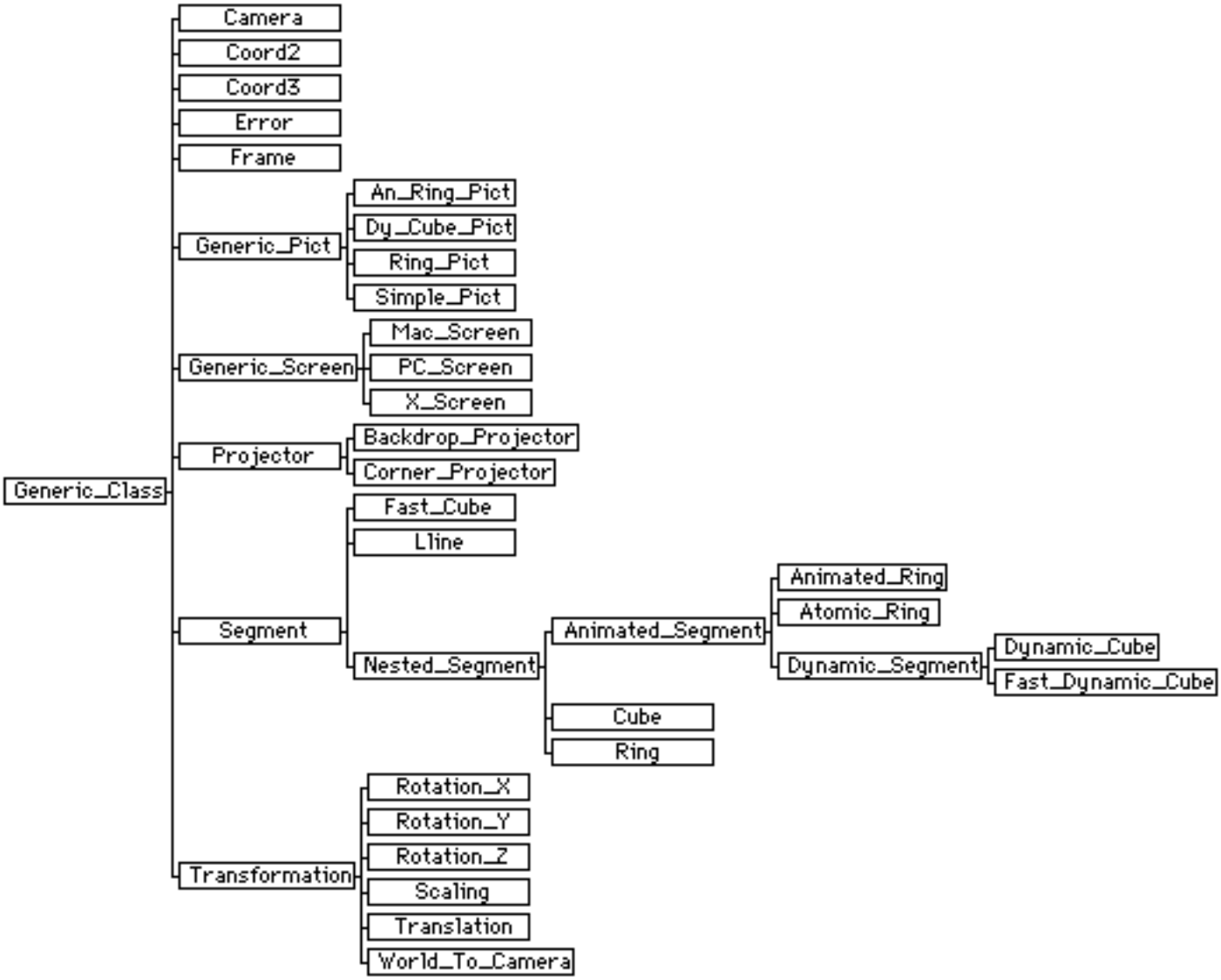
## DISTRIBUTION

Picture may be distributed freely as long as this Help file is included. It is intended for educational use only - permission is required for commercial use. Users are encouraged to add functionality to the existing Picture library, or to write new Screen classes for other computers. Please send any such additions to me, including documentation, so I can include them for distribution. I would also be interested in seeing well-documented copies of any Picture applications you come up with.

Note that an ealier version of the Picture library was released. The earlier version lacked C++ style constructors and destructors, and X Window portability.

## DESCRIPTION

Picture consists of the following major classes: Camera, Frame, Projector, Generic_Screen, Segment, Transformation, and Generic_Pict. The following illustrates the class hierarchy, showing the inheritance relationships:




**Picture class hierarchy (via Think C Browser)**

Three-dimensional figures are defined by creating new classes which are descendants of the Segment class. An existing descendant of Segment is Nested_Segment. Figures which are defined as descendants of Nested_Segments can easily be composed of combinations of existing figures. (See the description of the file segment.h, below.) An existing descendant of Nested_Segment is Animated_Segment. Figures which derive from this class can define animate() methods which indicate how the nested segments are to be animated. Dynamic_Segment's are Animated_Segment's which maintain velocity and acceleration vectors for physics simulations.

Segments can be moved and animated using Transformations, including scaling, translation, and rotation about the coordinate axes.

Camera objects are positioned and oriented in three dimensions to serve as viewpoints. (See the description of the file camera.h, below.) A rectangular cropping region (or Frame) on the Camera's projection plane declares the size of the "photo" for use with a Projector, as mentioned below.

Each application should contain a single "screen" object (whose class is a descendant of Generic_Screen) which is appropriate for the environment (although it may be possible to allow multiple screen objects, for situations where multiple monitors are available). Presently there exist Mac_Screen, PC_Screen, and X_Screen classes, and it is hoped that more will become available in the future.

The application may contain several Projector objects, each of which is associated with a rectangular window (or Frame) on the Screen. (It is also possible to have a console window for text I/O, as mentioned in the description of the file pict.c, below.) The application may also contain several Camera objects and Segments. Segments are drawn by specifying which Camera and which Projector to use. Segments may be drawn repeatedly in a loop to obtain animation.

A class called Generic_Pict serves as a generic Picture application, which defines Screen, Error, and Backdrop_Projector (black backdrop) objects. The easiest way to create a new Picture application is to define a new class which inherits these properties from Generic_Pict and overrides the run() method (and defines a new constructor and destructor) to draw specialized figures. These specialized figures should be defined elsewhere as subclasses of Segment. Four sample descendants of Generic_Pict are included, and mentioned below.

Note that the Error object automatically reports errors to the file error.fil. Examine this file after running the application to aid debugging.

## FILES

| | |
|---|---|
| picthelp.txt | this file. |
| Pict Help (MacWrite) | same as this file, but with text formatting and graphics. |
| picture.pi | sample Think C project file. |
| Makefile | sample Unix Makefile. You may need to change the ".c" extension to ".cc" for all the source files. |

(The following files comprise the Picture library. Please read the comments in the source and header files of any classes you will override, especially Generic_Pict.)

| | |
|---|---|
| class.h | defines Generic_Class, from which all classes derive. |
| class.c | methods for Generic_Class, enables reporting when a constructor fails, though this isn't used in the Picture code. |

| | |
|---|---|
| error.h | defines Error class, for reporting certain errors to error.fil. |
| error.c | Error methods. |
| screen.h | defines Generic_Screen class encapsulating low-level graphics instructions. |
| screen.c | methods for Generic_Screen. |
| macscrn.h | Mac_Screen for Macintosh/Think C environment. |
| macscrn.c | methods for Mac_Screen. |
| pcscrn.h | PC_Screen for PC/Borland Turbo C++. |
| pcscrn.c | methods for PC_Screen. |
| xscrn.h | X_Screen for Unix/X Window/gnu C++. |
| xscrn.c | methods for X_Screen. |
| color.h | defines mapping of color values, for color or B&W displays. |
| camera.h | defines Camera class representing the viewpoint for 3D perspective projection. Describes coordinate system used. |
| camera.c | Camera methods. |
| project.h | defines Projector class representing the mapping from the Camera's projection plane to a screen window |
| project.c | Projector methods |
| backdrop.h | defines Backdrop_Projector, which simply fills the entire screen with a single color to hide the operating system desktop. |
| backdrop.c | Backdrop_Projector methods. |
| coord.h | defines Coord2 and Coord3 classes for 2D and 3D coordinates. |
| coord.c | defines operations on Coord2 and Coord3 objects. |
| trans.h | defines Transformation class and descendants: Translation, Scaling, Rotation_X, Rotation_Y, and Rotation_Z. Also a composite transformation for 3D perspective transformation. |
| trans.c | methods for these transformation classes. |
| frame.h | defines the Frame class, for 2D mappings. |
| frame.c | methods for Frame. |
| segment.h | defines Segment and Nested_Segment for defining figures. Each Nested_Segment maintains an instance variable representing the cascaded transformations which have been applied to it. Thus the segments which are contained within it may be transformed with reference to a local coordinate system, without regard to transformations which are applied to the entire Nested_Segment. |
| segment.c | Segment methods. |
| line.h | defines Line class, a simple Segment descendant. |
| line.c | Line methods. |
| cube.h | defines Cube, a descendant of Nested_Segment consisting of several Lines. Also defines Fast_Cube, a direct descendant of Segment which draws faster. |

| | |
|---|---|
| cube.c | Cube methods. |
| ring.h | defines Ring, a Nested_Segment descendant for ring-shaped collections of Cubes. |
| ring.c | Ring methods. |
| animate.h | defines Animated_Segment class, a Nested_Segment descendant for defining animations of nested segments. |
| animate.c | Animated_Segment methods. |
| atring.h | defines Atomic_Ring, an Animated_Segment descendant very similar to Ring. |
| atring.c | Atomic_Ring methods. |
| anring.h | defines Animated_Ring, an Animated_Segment whose nested segments may also be animated. |
| anring.c | Animated_Ring methods. |
| dynamic.h | defines Dynamic_Segment class, an Animated_Segment descendant supporting dynamics. |
| dynamic.c | Dynamic_Segment methods. |
| dycube.h | defines Dynamic_Cube and Fast_Dynamic_Cube classes. |
| dycube.c | Dynamic_Cube and Fast_Dynamic_Cube methods. |
| pict.h | defines Generic_Pict, a generic class from which the main application class should be derived. |
| pict.c | Generic_Pict methods. Also contains comments on how to use a window for stdio-type text input and output. |
| main.c | main() function allocating a descendant of Generic_Pict. |

(The following files are used to demonstrate how to use the Picture library. Simple_Pict, Ring_Pict, An_Ring_Pict, and Dy_Cube_Pict are sample pictures derived from Generic_Pict. Comments in main.c indicate how to compile the applications.)

| | |
|---|---|
| simpict.h | defines Simple_Pict, a sample 2D picture. |
| simpict.c | Simple_Pict methods. |
| ringpict.h | defines Ring_Pict, a sample 3D picture. |
| ringpict.c | Ring_Pict methods. |
| anringpi.h | defines An_Ring_Pict, a sample animated 3D picture. |
| anringpi.c | An_Ring_Pict methods. |
| dycubpic.h | defines Dy_Cube_Pict, a sample dynamic animated 3D picture. |
| dycubpic.c | Dy_Cube_Pict methods. |

## REFERENCES

Hearn & Baker, "Computer Graphics", Prentice-Hall, 1986