Graphics.Gloss.Data.Picture

Safe Haskell	None
Language	Haskell98

Documentation

Contents

```
data Picture :: *
```

A 2D picture

Constructors

`Blank`	A blank picture, with nothing in it.
`Polygon Path`	A convex polygon filled with a solid color.
`Line Path`	A line along an arbitrary path.
`Circle Float`	A circle with the given radius.
`ThickCircle Float Float`	A circle with the given thickness and radius. If the thickness is 0 then this is equivalent to `Circle`.
`Arc Float Float Float`	A circular arc drawn counter-clockwise between two angles (in degrees) at the given radius.
`ThickArc Float Float Float Float`	A circular arc drawn counter-clockwise between two angles (in degrees), with the given radius and thickness. If the thickness is 0 then this is equivalent to `Arc`.
`Text String`	Some text to draw with a vector font.
`Bitmap Int Int BitmapData Bool`	A bitmap image with a width, height and some 32-bit RGBA bitmap data. The boolean flag controls whether Gloss should cache the data between frames for speed. If you are programatically generating the image for each frame then use `False`. If you have loaded it from a file then use `True`.
`Color Color Picture`	A picture drawn with this color.
`Translate Float Float Picture`	A picture translated by the given x and y coordinates.
`Rotate Float Picture`	A picture rotated clockwise by the given angle (in degrees).
`Scale Float Float Picture`	A picture scaled by the given x and y factors.
`Pictures [Picture]`	A picture consisting of several others.

Instances

```
Eq Picture
Data Picture
Show Picture
Monoid Picture
```

```
type Point = (Float, Float)
```

A point on the x-y plane.

```
type Vector = Point
```

A vector can be treated as a point, and vis-versa.

```
type Path = [Point]
```

A path through the x-y plane.

Aliases for Picture constructors

`blank :: Picture` Source

A blank picture, with nothing in it.

`polygon :: Path -> Picture` Source

A convex polygon filled with a solid color.

`line :: Path -> Picture` Source

A line along an arbitrary path.

`circle :: Float -> Picture` Source

A circle with the given radius.

`thickCircle :: Float -> Float -> Picture` Source

A circle with the given thickness and radius. If the thickness is 0 then this is equivalent to `Circle`.

`arc :: Float -> Float -> Float -> Picture` Source

A circular arc drawn counter-clockwise between two angles (in degrees) at the given radius.

`thickArc :: Float -> Float -> Float -> Float -> Picture` Source

A circular arc drawn counter-clockwise between two angles (in degrees), with the given radius and thickness. If the thickness is 0 then this is equivalent to `Arc`.

`text :: String -> Picture` Source

Some text to draw with a vector font.

`bitmap :: Int -> Int -> BitmapData -> Bool -> Picture` Source

A bitmap image with a width, height and a Vector holding the 32-bit RGBA bitmap data.

The boolean flag controls whether Gloss should cache the data between frames for speed. If you are programatically generating the image for each frame then use `False`. If you have loaded it from a file then use `True`.

`color :: Color -> Picture -> Picture` Source

A picture drawn with this color.

`translate :: Float -> Float -> Picture -> Picture` Source

A picture translated by the given x and y coordinates.

`rotate :: Float -> Picture -> Picture` Source

A picture rotated clockwise by the given angle (in degrees).

`scale :: Float -> Float -> Picture -> Picture` Source

A picture scaled by the given x and y factors.

`pictures :: [Picture] -> Picture` Source

A picture consisting of several others.

Compound shapes

`lineLoop :: Path -> Picture` Source

A closed loop along a path.

`circleSolid :: Float -> Picture` Source

A solid circle with the given radius.

`arcSolid :: Float -> Float -> Float -> Picture` Source

A solid arc, drawn counter-clockwise between two angles at the given radius.

`sectorWire :: Float -> Float -> Float -> Picture` Source

A wireframe sector of a circle. An arc is draw counter-clockwise from the first to the second angle at the given radius. Lines are drawn from the origin to the ends of the arc.

`rectanglePath` Source

`:: Float`	width of rectangle
`-> Float`	height of rectangle
`-> Path`	

A path representing a rectangle centered about the origin

`rectangleWire :: Float -> Float -> Picture` Source

A wireframe rectangle centered about the origin.

`rectangleSolid :: Float -> Float -> Picture` Source

A solid rectangle centered about the origin.

`rectangleUpperPath :: Float -> Float -> Path` Source

A path representing a rectangle in the y > 0 half of the x-y plane.

`rectangleUpperWire :: Float -> Float -> Picture` Source

A wireframe rectangle in the y > 0 half of the x-y plane.

`rectangleUpperSolid :: Float -> Float -> Picture` Source

A solid rectangle in the y > 0 half of the x-y plane.

Produced by Haddock version 2.16.1