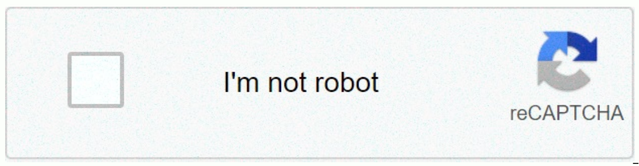

Continue

Android canvas clear background

The Canvas class keeps the draw calls. To draw something, you need 4 basic components: A bitmap to hold the pixels, a Canvas to host the pull calls (write in the bitmap), a token primitive (e.g. Rect, Path, text, Bitmap), and a paint (to describe the colors and styles for the drawing). This enum was decimated in API level 30. quickReject no longer uses it. enum Canvas.VertexMode int ALL_SAVE_FLAG restore everything when restore() is called (standard storage flags). Canvas() Build a blank raster cloth. Canvas (Bitmap bitmap) Builds a canvas with the specified bitmap to move in. boolean clipOutPath (Road path) Sets the clip to the difference from the current clip and the specified path. boolean clipOutRect(int left, int top, int right, int bottom) Set the clip to the difference of the current clip and the specified rectangle, which is expressed in local coordinates. boolean clipOutRect(Stretch correction) Sets the clip to the difference of the current clip and the specified rectangle, which is printed out in local coordinates. boolean clipOutRect (float left, float top, float right, float bottom) Set the clip to the difference of the current clip and the specified rectangle, which is expressed in local coordinates. boolean clipOutRect(RectF rect) Sets the clip to the difference of the current clip and the specified rectangle, which is expressed in local coordinates. Boolean clipPath(Road, Region.Op) This method was depreated in API level 26. Region.On values other than Region.Op #INTERSECT and Region.On#DIFFERENCE has the ability to expand the clip. The canvas clipping APOs are intended to expand only the clip due to a recovery operation. This enables a view parent to clip a napkin to clearly define the maximal drawing area of his children. The recommended alternative calls are clipPath (android.graphics.Path) and clipOutPath (android.graphics.Path); From which API level API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P only Region.On #interSECT and Region.On#DIFFERENCE is valid Region.On parameters. boolean clipPath (Road path) Crosses the current clip with the specified path. boolean clipRect (float left, float top, float right, float bottom, Region.Op) This method is depreated in API level 26. Region.On values other than Region.Op #INTERSECT and Region.On#DIFFERENCE has the ability to expand the clip. The canvas clipping APOs are intended to expand only the clip due to a recovery operation. This enables a view parent to clip a napkin to clearly define the maximal drawing area of his children. The recommended alternative calls are clipRect (float, float, float, float) and clipOutRect (float, float, float); From which API level API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P only Region.On #interSECT and Region.On #DIFFERENCE is valid Region.Op 2007 Honda CRF 2000 2007 Honda TR500 rect) Cross the current clip with the specified rectangle, which is expressed in local coordinates. boolean clipRect (Rect rect) Crosses the current clip with the specified rectangle, which is expressed in local coordinates. boolean clipRect(Stretch rect, Region.Op) This method is depreced in API level 26. Region.On values other than Region.Op #INTERSECT and Region.On#DIFFERENCE has the ability to expand the clip. The canvas clipping APOs are intended to expand only the clip due to a recovery operation. This enables a view parent to clip a napkin to clearly define the maximal drawing area of his children. The recommended alternative calls are clipRect (android.graphics.Rect) and clipOutRect(android.graphics.Rect); From which API level API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P only Region.On #interSECT and Region.On#DIFFERENCE is valid Region.On parameters. boolean clipRect(int left, int top, int right, int bottom) Cross the current clip with the specified rectangle, which is expressed in local coordinates. boolean clipRect (float left, float top, float right, float bottom) Cross the current clip with the specified rectangle, which is expressed in local coordinates. boolean clipRect(RectF rect, Region.Op) This method is depreated in API level 26. Region.On values other than Region.Op #INTERSECT and Region.On#DIFFERENCE has the ability to expand the clip. The canvas clipping APOs are intended to expand only the clip due to a recovery operation. This enables a view parent to clip a napkin to clearly define the maximal drawing area of his children. The recommended alternative calls are clipRect (android.graphics.RectF) and clipOutRect(android.graphics.RectF); From which API level API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P only Region.On #interSECT and Region.On#DIFFERENCE is valid Region.On parameters. void concat (Matrix matrix) Preconcat the current matrix with the specified matrix. void disablesZ() Disable Z support, which prevents any RenderNodes drawn to this point from being visually reordered or has delivered shadows. void trekARGB(int a, int r, int g, int b) Fill off the entire canvas's bitmap (limited to the current clip) with the specified ARGB color, using circulation porterduff. void pullArc (float left, float top, float right, float bottom, float startAngle, float veeAngle, boolean useCenter, Paint paint) Draw the specified arc, which will be scaled to fit into the specified oval. void pullArc(RectF oval, float startAngle, float wipe Angle, boolean usageCenter, Paint paint) Draw the specified arc, which will be scaled to fit into the specified oval. void pullBitmap(Bitmap bitmap, Matrix matrix, Paint paint) Draw the bitmap using the specified void pullBitmap(int[] colors, int offset, int stride, float x, float y, int width, int height, boolean hasAlpha, Paint Paint This method was depreated in API level 21. Use with a hardware accelerated canvas requires an internal copy of color buffer content each time this method is called. Using a Bitmap avoids this copy, and allows the application to more explicitly control the lifetime and copies of pixel data. void pullBitmap(int[] colors, int offset, int stride, int x, int y, int width, int height, boolean hasAlpha, Paint paint) This method is depreated in API level 21. Use with a hardware accelerated canvas requires an internal copy of color buffer content each time this method is called. Using a Bitmap avoids this copy, and allows the application to more explicitly control the lifetime and copies of pixel data. void pullBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint) Draw the specified bitmap, scale/translation automatic to fill the destination rectangle. void pullBitmap(Bitmap bitmap, Rect src, RectF dst, Paint paint) Draw the specified bitmap, scale/translation automatic to fill the destination rectangle. void pullBitmap(Bitmap bitmap, float left, float top, Paint paint) Draw the specified bitmap, with its upper/left corner at (x,y), using the specified paint, converted by the current matrix. void pullBitmapMesh(Bitmap bitmap, int meshWidth, int meshHeight, float[] vertical, int vertOffset, int[] colors, int colorOffset, Paint paint) Draw the bitmap through the mesh, where mesh verticals are evenly distributed across the bitmap. void pullCircle(float cx, float cy, float radius, Paint paint) Draws the specified circle using the specified paint. void pullColor(long color, BlendMode mode) Fill the entire canvas's bitmap (limited to the current clip) with the specified color and mixing mode. void pullColor(int color) Fill off the entire canvas's bitmap (limited to the current clip) with the specified color, using circling porter duff. void pullColor(int color, BlendMode mode) Fill the entire canvas's bitmap (limited to the current clip) with the specified color and mixing mode. void pullColor(long color) Fills the entire canvas's bitmap (limited to the current clip) with the specified color, using srcover porterduff mode. void pullColor(int color, PorterDuff.Mode) Fill the entire canvas's bitmap (limited to the current clip) with the specified color and porter-duff xfermode. void pullDoubleRoundRect(RectF outer, float[] outerRadi, RectF inner, float[] innerRadii, Paint paint) Pull a double rounded rectangle using the specified paint. void pullDoubleRoundRect(RectF outer, float outerRx, float outer, RectF inner, float innerRx, float inner, Paint paint) Pull a double rounded rectangle using the specified paint. void pullLine(float startX, startY, float stopX, float stopY, Paint paint) Draw a line segment with the specified start and stop x,y coordinates, using the specified paint. void pullLines(float[] pts, int offset, offset, Score, Paint Paint) Draw a series of lines. void pullLines(float[] pts, Paint paint) void pull (float left, float top, float right, float right, float bottom, Paint paint) Draw the specified oval using the specified paint. void pullOval(RectF oval, Paint paint) Draws the specified oval using the specified paint. void pullPaint(Paint paint) Fill the entire canvas's bitmap (limited to the current clip) with the specified paint. void pullPath(Path Path, Paint Paint) Draws the specified path using the specified paint. void pullPicture (Photo photo, RectF dst) Draw the photo, stretched to fit into the i.e. rectangle. void TokenPicture (Picture pictured) Save the canvas state, draw the picture, and restore the canvas state. void pullPicture (Photo photo, Rect dst) Draw the photo, stretched to fit into the i.e. rectangle. void drawingPoint(float x, float y, Paint paint) Helper for drawing points() for drawing a single point. void pullPoints(float[] pts, Paint paint) Helper for DrawingPoints() that assumes you have the entire array void pullPoints(float[] pts, int offset, int count, Paint paint) Draw a series of points. void pullPosText(String text, float[] mail, Paint paint) This method is depreated in API level 16. This method does not support glyph composition and decomposition and therefore should not be used to render complex scripts. Nor does it deal with complementary characters (e.g., emoji). void pullPosText(char[] text, int index, int count, float[] mail, Paint paint) This method is depreated in API level 16. This method does not support glyph composition and decomposition and therefore should not be used to render complex scripts. Nor does it deal with complementary characters (e.g., emoji). void trekRGB(int r, int g, int b) Fill the entire canvas' bitmap (limited to the current clip) with the specified RGB color, using srcover porterduff mode. void pullAchew(float left, float top, float right, float bottom, Paint paint) Draw the specified Stretch using the specified paint. void PullAcken(Rect r, Paint Paint) Draws the specified Rack using the specified Paint. void TokenAcity(RectF rect, Paint paint) Draws the specified Rack using the specified paint. void pullRenderNode(RenderNode renderNode) draws the given RenderNode. void pullRoundRect(RectF rect, float rx, float row, Paint paint) Draws the specified round-rack using the specified paint. void pullRoundRect(float left, float top, float right, float bottom, float rx, float row, Paint paint) Draw the specified round-rack using the specified paint. void tokenText(CharSequence text, int start, int end, float x, float y, Paint paint) Draw the specified range text, specified by start/end, with its origin at (x,y), in the Paint. void tokenText(String text, float x, float y, Paint paint) Draw the text, with origin at (x,y), using the specified paint. void tokenText(char[] text, int index, int count, drive drives float y, Paint paint) Draw the text, originating (x,y), using the specified paint. void tokenText(String text, int start, int end, float x, float y, Paint paint) Draw the text, with origin at (x,y), using the specified paint. void pullTextOnPath(String text, Path path, float hOffset, float vOffset, Paint paint) Draw the text, with origin at (x,y), using the specified paint, along the specified path. void pullTextOnPath(char[] text, int index, int count, Path path, float hOffset, float vOffset, Paint paint) Draw the text, with origin at (x,y), using the specified paint, along the specified path. void tokenTextRun(MeasuredText text, int start, int end, int contextStart, int contextEnd, float x, float y, boolean isRtl, Paint paint) Draw a run of text, all in a single direction, with optional context for complex text formation. void tokenTextRun(char[] text, int index, int count, int contextIndex, int contextCount, float x, float y, boolean isRtl, Paint paint) Draw a run of text, all in a single direction, with optional context for complex text formation. void tokenTextRun(CharSequence text, int start, int end, int contextStart, int contextEnd, float x, float y, boolean isRtl, Paint paint) Draw a run of text, all in a single direction, with optional context for complex text formation. void pullVertical(Canvas.VertexMode mode, int vertexCount, float[] vertical, int vertOffset, float[] texs, int texOffset, int[] colors, int colorOffset, short[] indexes, int indexOffset, int indexCount, Paint paint) Draw the array of vertical, interpreted as triangles (based on void EnableZ() Enable Z support which defaults are disabled to disabled. boolean getClipBounds(Rect boundaries) Returns the boundaries of the current clip (in local coordinates). , and return true if it is not empty. final Stretch getClipBounds() Retrieves the boundaries of the current clip (in local coordinates). int getDensity() Returns the target density of the canvas. DrawFilter getDrawFilter() int getHeight() Returns the height of the current token layer void getMatrix(Matrix ctm) This method is depreated in API level 16. Hardware accelerated diaper can have any matrix when passed to a view or tokenable, since it is implementation that is defined where in the hierarchy such dians are created. It is recommended in such cases to either pull content regardless of the current matrix, or to detect relevant transform state outside the canvas. final Matrix getMatrix() This method was depreded in API level 16. Hardware accelerated diaper can have any matrix when passed to a view or tokenable, since it is implementation that is defined where in the hierarchy such dians are created. It is recommended in such cases to pull content regardless of the current matrix, or to detect relevant transform state outside the canvas. int getMaximumBitmapHeight() Returns the maximum allowed height for bitmaps drawn this canvas. int getMaximumBitmapWidth() Returns the maximum allowed width for bitmaps drawn with this canvas. int getSaveCount() Returns the number of matrix/clip states on the Canvas' private stack. int getWidth() Returns the width of the current drawing layer of boolean isHardwareAccelerated() Indicates whether this Canvas uses hardware acceleration. boolean isOpaque() Return true if the device that pulls the current layer in is opaque (i.e. boolean quickReject (float left, float top, float right, float under, Canvas.EdgeType type) This method is depreated in API level 30. The EdgeType is ignored. Use quickReject (float, float, float, float) instead. Boolean quickReject(Path path) Return true if the specified path, after being transformed by the current matrix, would lie completely outside the current clip. boolean quickReject (RectF rect) Return true if the specified rectangle, after being it transformed by the current matrix, would lie completely outside the current clip. boolean quickReject(RectF rect, Canvas.EdgeType type) This method is depreated in API level 30. The EdgeType is ignored. Use quickReject instead (android.graphics.RectF). boolean quickReject(Road path, Canvas.EdgeType type) This method is depreced in API level 30. The EdgeType is ignored. Instead, use quickReject(android.graphics.Path) boolean quickReject (float left, float top, float right, float bottom) Return true if the specified rectangle, having been converted by the current matrix, would lie completely outside the current clip. void restore() This call balances a previous call to store (), and is used to remove all changes to the matrix/clip state since the last storage call. void restoreToCount(int saveCount) Efficient way to pop any calls to save() that happened after the storage count reached SaveCount. void Rotate (float grades) Preconsect the current matrix with the specified rotation. final void rotate (float grades, float px, float py) Preconcat the current matrix with the specified rotation. int save() Save the current matrix and clip on a private stack. Int saveLayer(RectF boundaries, Paint paint, int saveFlags) This method is depreated in API level 26. Use SaveLayer instead (android.graphics.RectF, android.graphics.Paint). Int saveLayer(RectF boundaries, Paint paint) It behaves the same as save(), but in addition it assigns and redirects drawing to an offscreen rendering target. int saveLayer (float left, float top, float right, float bottom, Paint paint) Convenience for saveLayer(android.graphics.RectF, android.graphics.Paint) that takes the four float coordinates of the borders rectangle. int saveLayer (float left, float top, float right, float bottom, Paint paint, int saveFlags) This method is depreated in API level 26. Use saveLayer (float, float, float, float, android.graphics.Paint) instead. Int (float left, float top, float right, right, bottom, int alpha, int saveFlags) This method is depreated in API level 26. Use saveLayerAlpha (float, float, float, float, int) instead. int saveLayerAlpha(RectF boundaries, int alpha) Convenience for saveLayer(android.graphics.RectF, android.graphics.Paint) but instead of taking an entire Paint object it only takes the alpha parameter. int saveLayerAlpha(RectF boundaries, int alpha, int saveFlags) This method is depreated in API level 26. Use SaveLayerAlpha instead (android.graphics.RectF, int). int saveLayerAlpha (float left, float top, float right, float bottom, int alpha) Convenience for saveLayerAlpha (android.graphics.RectF, int) that takes the four float coordinates of the borders rectangle. final void scale (float sx, float silk, float px, float py) Preconcat the current matrix with the specified scale. void scale (float sx, float its) Preconcat the current matrix with the specified scale. void setBitmap(Bitmap bitmap) Specifies a bitmap for the canvas to revok. void setDensity(int density) Specifies the density for this Canvas' backing bitmap. void setDrawFilter(DrawFilter filter) void setMatrix(Matrix matrix) Completely replaces the current matrix with the specified matrix. void skew (float sx, float its) Preconcat the current matrix with the specified skew. void translate(float dx, float dy) Preconcat the current matrix with the specified translation of class java.lang.Object Object clone() Create and returns a copy of this object. boolean equal (Object obj) Indicates whether another object is equal to this one. void finalizes() Called by the garbage collector on an object when garbage collection determines that there are no more references to the object. final Class<?> getClass() Returns the runtime class of this Object. int hashCode() Returns a hash code value for the object. final void set() Wake up to a single thread awaiting this object's monitor. String toString() Returns a string representation of the object. final void await(long timeout, int nanos) Causes the current thread to wait until another thread invokes the notification() method or the notifyingAll() method for this object, or another thread interrupts the current thread, or a certain amount of real time has elapsed. final void await(long timeout) Causes the current thread to wait until either another thread invokes the notify() method or the notifyingAll() method for this object, or a specified amount of time has elapsed. final void await() Causes the current thread to wait until another thread invokes the notify() method or the notifyIngAll() method for this object. public static final int ALL_SAVE_FLAG Restore all when restore() is called (standard save flags). Note: for performance reasons, it is recommended to pass it - the full set of flags - to save any call toLayer() save() saveLayerAlpha() variants. Note: all methods that accept this flag have flagless versions that equate to passing this flag. Constant Value: 31 (0x0000001f) public Canvas() Build an empty raster cloth. Use setBitmap() to specify a bitmap to move in. The initial target density is Bitmap #DENSITY_NONE; it will typically be replaced when a target bitmap is set for the canvas. public Canvas (Bitmap bitmap) Builds a canvas with the specified bitmap to move in. The bitmap must be stabistabel. The initial target density of the canvas is the same as the given bitmap's density. Parameters bitmap Bitmap: Specifies a stable bitmap for the canvas to pull in. This value cannot be null. public boolean clipOutPath (Padpad) Sets the clip to the difference from the current clip and the specified path. Parameters path path: The path used in the difference operation This value cannot be null. Returns boolean true if the resulting clip shows non-empty public boolean clipOutRect(int left, int top, int right, int bottom) Sets the clip to the difference from the current clip and the specified rectangle, which is expressed in local coordinates. Parameters left int: The left side of the rectangle used in the difference operation top int: The top of the rectangle used in the difference operation right int: The right side of the rectangle used in the difference operation bottom int: The bottom of the re rectangle used is in the difference operation Returns boolean true if the resulting clip is non-empty public boolean clipOutRect(Rect rect) Sets the clip to the difference from the current clip and the specified rectangle, which is expressed in local coordinates. Parameters rect Rect: The rectangle to perform a difference on with the current clip. This value cannot be null. Returns boolean true if the resulting clip is non-empty public boolean clipOutRect (float left, floats top, floats right, floats below) Sets the clip to the difference of the current clip and the specified rectangle, which is expressed in local coordinates. Parameters left float: The left side of the rectangle used in the difference operation top float: The top of the rectangle used in the difference operation right float: The right side of the rectangle used in the difference operation bottom drives: The bottom of the rectangle used in the difference operation Returns boolean true if the resulting clip is non-empty public boolean clipOutRect(RectF rect) Sets the clip to the difference of the current clip and the specified rectangle, which is expressed in local coordinates. Parameters rect RectF: The rectangle to perform a difference on with the current clip. This value cannot be null. Give boolean true if the clip non-empty public boolean clipPath(Path path) Intersect is the current clip with the specified path. Parameters path path: The path to crossing with the current clip This value value be. Returns boolean true if the resulting clip is non-empty Added in API level 1 Decimated in API level 26 public boolean clipRect (float left, float top, float right, float bottom, Region.Op) This method is depreated in API level 26. Region.On values other than Region.Op #INTERSECT and Region.On#DIFFERENCE has the ability to expand the clip. The canvas clipping APOs are intended to expand only the clip due to a recovery operation. This enables a view parent to clip a napkin to clearly define the maximal drawing area of his children. The recommended alternative calls are clipRect (float, float, float, float) and clipOutRect (float, float, float); From which API level API

android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P only Region.On #interSECT and Region.On#DIFFERENCE is valid Region.On parameters. Changes the current clip with the specified rectangle, which is expressed in local coordinates. Parameters left drives: The left side of the rectangle to cross with the current clip top drives: The top of the rectangle to cross with the current clip right drives: The right side of the rectangle to cross with the current clip bottom drives: The bottom of the rectangle to cross with the current clip on Region.On: How the clip is changed This value cannot be changed Return boolean true if the resulting clip is non-empty public boolean clipRect(RectF rect) Intersect the current clip with the specified rectangle, which is expressed in local coordinates. Parameters rect RectF: The rectangle to cross with the current clip. This value cannot be null. Returns boolean true if the resulting clip is non-empty public boolean clipRect (Rect reg) Crosses the current clip with the specified rectangle, which is expressed in local coordinates. Parameters rect Rect: The rectangle to cross with the current cut. This value cannot be null. Returns boolean true if the resulting clip is non-empty public boolean clipRect(int left, int top, int right, int bottom) Crosses the current clip with the specified rectangle, which is expressed in local coordinates. Parameters left int: The left side of the rectangle to cross with the current clip top int: The top of the rectangle to cross with the current clip right int: The right side of the rectangle to cross crossing with the current clip bottom int: The bottom of the rectangle to cross with the current clip Returns boolean where if the resulting clip is non-empty public boolean clipRect (float left float top, float right, float bottom) Cross the current clip with the specified rectangle, which is expressed in local coordinates. Parameters left drives: Cross the left side of the rectangle with the current clip top drives: The top of the rectangle to cross with the current clip right drives: The right side of the rectangle to cross with the current clip float: The bottom of the rectangle to cross with the current clip Returns boolean true if the resulting clip is non-empty public void concat (Matrix matrix) Preconcat the current matrix with the specified matrix. If the specified matrix is not, this method does nothing. Parameters matrix Matrix: The matrix to precaution with the current matrix This value can be neither. public void disablesZ() Disable Z support, which prevents any RenderNodes drawn to this point from being visually reordered or has delivered shadows. Note: This is not affected by any storage() or recovery() calls if it is not considered to be part of the current matrix or clip. See enableZ() public void pullARGB(int a, int r, int g, int b) Fill off the entire canvas's bitmap (limited to the current clip) with the specified ARGB color, using srcover porterduff mode. Parameters an int: alpha component (0.255) of the color to draw on the canvas r int: red component (0.255) of color to draw on the canvas g int: green Component (0..255) of the color to draw on the canvas b int: blue component (0.255) of the color to draw on the canvas public void trekArc (float left, float top, float right , float bottom, float startAngle, float whipAngle, boolean usageCenter, Paint paint) Draw the specified arc, which will be scaled to fit into the specified oval. If the starting angle is negative or >= 360, the starting angle is treated as starting angle modulo 360. If the livestock angle is for >= 360, the oval is completely drawn. Note that this is slightly different from SkPath::arcTo, which treats the cattle angle modulo 360. If the cattle angle is negative, the cattle angle is treated as livestock angle modulo 360 The bow is drawn clockwise. An angle of 0 degrees corresponds to the geometric angle of 0 degrees (3 hours on a clock.) Parameters left float upper drives upper drives bottom drives startAngle float: Start angle (in degrees) where the arc begins to float sweepIng Angle: Swipe angle (in degrees) measured clockwise useCenter boolean: If true, include the center of the oval in the arc, and close it if stroked. This will draw a wedge paint paint: The paint used to draw the bow This value cannot be null. public void pullArc(RectF oval, float startAngle, float wipe Angle, boolean usageCenter, Paint paint) Draw the specified arc, which will be scaled to fit into the specified oval. If the starting angle is negative or >= 360, the starting angle is treated as starting angle modulo 360. If the livestock angle is for >= 360, the oval is completely drawn. Note that this is slightly different from SkPath::arcTo, which treats the cattle angle modulo 360. If the cattle angle is negative, the cattle angle is treated as livestock angle modulo 360 The bow is drawn clockwise. An angle of 0 degrees corresponds to the geometric angle 0 degrees (3 hours on a watch.) Parameters oval RectF: The limits of ovals used to and size of the arc This value cannot be zero. StartAngle float: Start angle (in degrees) where the arc begins to float sweeping Angle: Swipe angle (in degrees) measured clockwise useCenter boolean: If true, include the center of the oval in the arc, and close it as it is stroked. This will draw a wedge paint paint: The paint used to draw the bow This value cannot be null. public void pullBitmap(Bitmap bitmap, Matrix matrix, Paint paint) Draw the bitmap using the specified matrix. Parameters bitmap Bitmap: The bitmap to draw this value cannot be null. Matrix Matrix: The matrix used to convert the bitmap when drawn This value cannot be zero. paint: May be insidious. The paint used to draw the bitmap This value can be instigated. Added in API level 3 Decimated in API level 21 public void pullBitmap(int[] colors, int offset, int stride, float x, float y, int width, int height, boolean hasAlpha, Paint paint) This method is depreated in API level 21. Use with a hardware accelerated canvas requires an internal copy of color buffer content each time this method is called. Using a Bitmap avoids this copy, and allows the application to more explicitly control the lifetime and copies of pixel data. Treats the specified array of colors as a bitmap, and draws them. It returns the same result as first creating a bitmap from the array, and then drawing it, but this method explicitly avoids creating a bitmap object that can be more efficient if the colors often change. Parameters colors int: Variety of colors that represent the pixels of the bitmap This value cannot be null. offset int: Offset in the variety of colors for the first pixel stride int: The number of colors in the array between rows (must > = width or x drives: The X coordinates for where to pull the bitmap y float: The Y coordinate for where to pull the bitmap width int: The width of the bitmap height int: The height of the bitmap hasAlpha boolean : True if the alpha channel of the colors contains valid values. If false, the alpha bytes are ignored (assumed to be 0xFF for each pixel). paint: May be insidious. The paint used to draw the bitmap This value can be instigated. Added in API level 1 Decimated in API level 21 public void pullBitmap(int[] colors, int offset, int stride, int x, int y, int width, int height, boolean hasAlpha, Paint paint) This method is depreated in API level 21. Use with a hardware accelerated canvas requires an internal copy of color buffer content each time this method is called. Using a Bitmap avoids this copy, and allows the application to explicitly control the lifetime and copies of pixel data. Legacy version of pullBitmap(int[] colors, ...) that have taken ints for x,y Parameters colors int: This value cannot be null. offset int stride int x int y int width int height int hasAlpha boolean paint Paint: This value can be sneering. public void pullBitmap(Bitmap(Bitmap Rect src, Rect dst, Paint paint) Draws the specified bitmap, scale/translation automatically to fill the destination rectangle. If the source rectangle is not null, it specifies the subset of the bitmap to draw. Note: if the paint contains a mask filter that generates a mask that extends beyond the bitmap's original width/height (e.g. BlurMaskFilter), then the bitmap will be drawn as if it were in a Shader with KLEM mode. Thus, the color outside the original width/height will repeat the edge color. This feature ignores the density associated with the bitmap. This is because the source and destination rectangle coordinating spaces are in their respective densities, so the appropriate scale factor must be applied. Parameters bitmap Bitmap: The bitmap to be drawn This value cannot be null. src Rect: May be negligent. The subset of the bitmap to be drawn This value can be nunite. dst Rect: The rectangle that the bitmap will be scaled/translated to fit into this value cannot be null. paint: May be insidious. The paint used to draw the bitmap This value can be instigated. public void pullBitmap(Bitmap bitmap, Rect src, RectF dst, Paint paint) Draw the specified bitmap, scale/translation automatic to fill the destination rectangle. If the source rectangle is not null, it specifies the subset of the bitmap to draw. Note: if the paint contains a mask filter that generates a mask that extends beyond the bitmap's original width/height (e.g. BlurMaskFilter), then the bitmap will be drawn as if it were in a Shader with KLEM mode. Thus, the color outside the original width/height will repeat the edge color. This feature ignores the density associated with the bitmap. This is because the source and destination rectangle coordinating spaces are in their respective densities, so the appropriate scale factor must be applied. Parameters bitmap Bitmap: The bitmap to be drawn This value cannot be null. src Rect: May be negligent. The subset of the bitmap to be drawn This value can be nunite. dst RectF: The rectangle that the bitmap will be scaled/translated to fit into this value cannot be null. paint: May be insidious. The paint used to draw the bitmap This value can be instigated. public void pullBitmap(Bitmap bitmap, float left, float top, Paint paint) Draw the specified bitmap, with its upper/left corner at (x,y), using the specified paint, converted by the current matrix. Note: if the paint contains a mask filter that generates a mask that extends beyond the bitmap's original width/height (e.g. BlurMaskFilter), then the bitmap will be pulled wash it in a Shader with KLEM mode. Thus, the color outside the original width/height will repeat the edge color. If the bitmap and canvas have different densities, this feature will ensure that the bitmap automatically scales to draw on the same density as the canvas. Parameters Parameters Bitmap: The bitmap to be drawn This value cannot be null. Left drives: The position of the left side of the bitmap drawn top float: The position of the upper side of the bitmap drawn paint: The paint used to pull the bitmap (can be zero) This value can be null. public void pullBitmapMesh(Bitmap bitmap, int meshWidth, int meshHeight, float[] verts, int vertOffset, int[] colors, int colorOffset, Paint paint) Draw the bitmap through the mesh, where mesh verticals are evenly distributed across the bitmap. There is meshWidth + 1 vertical ear, and meshHeight + 1 vertical mode. The verts array is obtained in row-large order, allowing the first meshWidth + 1 vertical spread across the top of the bitmap from left to right. A more common version of this method is drawVertices(). Before API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P vertOffset and colorOffset were ignored and effectively treated them as zeros. In API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P and above these parameters will be respected. Parameters bitmap Bitmap: The bitmap to draw using the mesh This value cannot be null. meshWidth int: The number of columns in the mesh. Nothing is drawn if it ints 0 meshHeight: The number of rows in the mesh. Nothing is drawn if it drives 0 vertical: Variety x,y pairs, specifying far where the mesh should be drawn. There must be at least (meshWidth+1) * (meshHeight +1) * 2+ vertOffset values in the array Be This value cannot be null. vertOffset int: Number of verts elements to skip before drawing colors int: May be null. Specifies a color at each vertex, which is multiplied across the cell, and whose values are multiplied by the corresponding bitmap colors. If not null, there must be at least (meshWidth +1) * (meshHeight + 1) + colorOffset values in the array. This value can be nite. colorOffset int: Number of color elements to skip before drawing paint Paint: Can be null. The paint used to draw the bitmap This value can be instigated. public void pullCircle(float cx, float cy, float radius, Paint paint) Draws the specified circle using the specified paint. If radius is Parameters cx float: The x-coordination of the center of the circle drawn cy float: The y-coordination of the center of the circle radius float is drawn: The radius of the circle drawn paint Paint: The paint used to pull the circle This value cannot be null. public void pullColor(long color, BlendMode mode) Fill the entire canvas's bitmap (limited to the current clip) with the specified color and mixing mode. Parameters color long: the ColorLong to draw on the canvas. See the Color class for details on ColorLongs. Mode the mixing mode to apply for the color This value cannot be null. public void PullColor(int color) Fill the entire canvas's bitmap (restricted to the current clip) with the specified color, using from porterduff mode. Parameters color int: the color to draw on the canvas public void pullColor(int color, BlendMode mode) Fill the entire canvas's bitmap (limited to the current clip) with the specified color and mixing mode. Parameters color int: the color to draw on the canvas mode BlendMode: the blendmode to apply to the color This value cannot be null. public void pullColor(long color) Fills the entire canvas's bitmap (limited to the current clip) with the specified color, using circling porter duff mode. Parameters color long: the ColorLong to draw on the canvas. See the Color class for details on ColorLongs. public void pullColor(int color, PorterDuff.Mode) Fill the entire canvas' bitmap (limited to the current clip) with the specified color and porter-duff xfermode. Parameters color int: the color to draw on the canvas mode PorterDuff.Mode: the porter-duff mode to apply to the color This value cannot be null. public void pullDoubleRoundRect(RectF outer, float[] outerRadii, RectF inner, float[] innerRadii, Paint paint) Pull a double rounded rectangle using the specified paint. The resulting round rack will be filled in the area defined between the outer and inner rectangular boundaries as the Paint is configured with Paint.Style #FILL. Otherwise if Paint.Style is used #STROKE, 2 rounded stretch riots will be drawn at the outer and inner rounded rectangles Parameters outer RectF: The outer rectangular boundaries of the round Rack to be drawn This value cannot be zero. outerRadii float: Array of 8 drives that are the x, y angle radii for top right, bottom right, bottom left corners respectively on the outer rounded rectangle This value cannot be zero. Inner RectF: The inner rectangular confines of the round Rack to be drawn This value cannot be null. innerRadii float: Variety of 8 drives that the x, y angle radii for top right, bottom right, bottom left corners respectively on the outer rounded rectangle This value cannot be zero. Paint Paint: The paint used to draw the double round Stretch This value cannot be null. public void pullDoubleRoundRect(RectF outer, float outerRx, float outer, RectF inner, float innerRx, float inner, Paint paint) Pull a double rounded rectangle using the specified paint. The resulting round rack will be filled in the area defined between the outer and inner rectangular boundaries as the Paint is configured with Paint.Style #FILL. Otherwise if Paint.Style #STROKE is used, 2 rounded stretch riots will be drawn at the outer and inner rounded rectangles Parameters outer RectF: The outer rectangular confines of the round Rack must be drawn This value cannot be null. outerRx drives: The x-radius of the oval used to round out the corners on the outer rectangle outer drives: The y radius of the oval used to round the corners on the outer rectangle inner RectF: The inner rectangular boundaries of the to be drawn This value cannot be null. InnerRx drives: The x-radius of the oval used to round the corners on the inner rectangle inner drives: The y-radius of the oval used to round the corners on the outer rectangle paint Paint: The paint used to draw the double round Stretch This value cannot be null. public void pullLine(float startX, float startY, float stopX, float stopY, Paint paint) Draw a line segment with the specified start and stop x,y coordinates, using the specified paint. Note that since a line is always framed, the style in the paint is ignored. Unearthed lines (length is 0) will not be drawn. Parameters startX drives: The x-coordination of the starting point of the line startY drives: The y coordination of the starting point of the line stopX drives stopY drives paint Paint: The paint used to draw the line This value cannot be null. public void pullLines(float[] pts, int offset, int count, Paint paint) Draw a series of lines. Each line is taken from 4 consecutive values in the pts array. So to draw 1 line, the array must contain at least 4 values. It is logically the same as drawing the array as follows: drawing line(pts[0], pts[1], pts[2], pts[3]) followed by drawLine(pts[4], pts[5], pts[6], pts[7]) and so on. Parameters pts float: Variety points to draw [x0 y0 x1 y1 x2 y2...] This value cannot be null. offset int: Number of values in the array to skip before drawing. count int: The number of values in the array to process, after skipping offsets from them. Since each line uses 4 values, the number of lines drawn is really (>> 2). Paint Paint: The paint used to draw the points This value cannot be null. public void pullLines(float[] pts, Paint paint) Parameters pts float: This value cannot be null. Paint Paint: This value cannot be null. public void pullOval (float left, float top, float right, float bottom, Paint paint) Draw the specified oval using the specified paint. The oval will be filled or framed based on the style in the paint. Parameters left float upper drives right float bottom drives paint Paint: This value cannot be null. public void pullOval(RectF oval, Paint Paint) Draws the specified oval using the specified paint. The oval will be filled or framed based on the style in the paint. Parameters Oval RectF: The rectangle boundaries of the oval to be drawn This value cannot be null. Paint Paint: This value cannot be null. public void pullPaint(Paint paint) Fill the entire canvas's bitmap (limited to the current clip) with the specified paint. It is equivalent (but faster) to an infinitely large rectangle with the paint. Parameters paint Paint: The paint used to draw on the canvas This value cannot be null. public void PullPath(Path Path, Paint Paint) Draws the specified path using the specified paint. The path will be filled or framed based on the style in the paint. Parameters path path: The The to be drawn This value cannot be null. Paint Paint: The paint used to draw the path This value cannot be null. public void pullPicture(Photo, RectF dst) Draw the photo, stretched to fit into the i.e. rectangle. Parameters picture Picture: This value cannot be null. dst RectF: This value cannot be null. public void tokenPicture (Photo) Save the canvas state, draw the picture, and restore the canvas state. This is different from picture.draw(canvas), which performs no storage/restore. Note: This forces the image to call #endRecording Picture to prepare for playback. Parameters picture Picture: The image to be drawn This value cannot be null. public void pullPicture (Photo, Rect dst) Draw the photo, stretched to fit into the i.e. rectangle. Parameters picture Picture: This value cannot be null. dst Rect: This value cannot be null. public void drawing point (float x, float y, Paint paint) Helper for drawing points() for drawing a single point. Parameters x float y float paint Paint: This value cannot be null. public void pullPoints(float[] pts, Paint paint) Helper for drawingPoints() that assumes you float the entire array Parameters pts: This value cannot be null. Paint Paint: This value cannot be null. public void PullPoints(float[] pts, int offset, int count, Paint paint) Draw a series of points. Each point is centered on the coordination specified by pts[], and its diameter is specified by the paint's stroke width (as converted by the canvas' CTM), with special treatment for stroke width of 0, which always draws exactly 1 pixel (or at most 4 if antialiasing is activated). The shape of the tip is controlled by the paint's Cap type. The shape is a square unless the cap type is Round, in which case the shape is a circle. Parameters pts float: Variety points to draw [x0 y0 x1 y1 x2 y2...] offset int: Number of values to skip before you start drawing. count int: The number of values to process, after they offset from them. Since one point uses two values, the number of points drawn is really (>> 1). Paint Paint: The paint used to draw the points This value cannot be null. Added in API level 1 Decimated in API level 16 public void pullPosText(String text, float[] mail, Paint paint) This method is depreated in API level 16. This method does not support glyph composition and decomposition and therefore should not be used to render complex scripts. Nor does it deal with complementary characters (e.g., emoji). Draws the text into the array, with each character's origin specified by the mail array. Parameters text string: The text to be drawn This value cannot be null. post drives: Variety of [x,y] used to position each character This value cannot be null. Paint Paint: The paint used for the text (e.g. color, size, style) This value cannot be null. Added in API level 1 1 in API level 16 public void pullPosText(char[] text, int index, int count, float[] post, Paint paint) This method is depreated in API level 16. This method does not support glyph composition and decomposition and therefore should not be used to render complex scripts. Nor does it deal with complementary characters (e.g., emoji). Draws the text into the array, with each character's origin specified by the mail array. Parameters text char: The text to be drawn This value cannot be null. index int: The index of the first character to draw count int: The number of characters to draw, from index. post drives: Variety of [x,y] positions, used to position each character This value cannot be null. Paint Paint: The paint used for the text (e.g. color, size, style) This value cannot be null. public void pull GB(int r, int g, int b) Fill the entire canvas' bitmap (limited to the current clip) with the specified RGB color, using srcover porterduff mode. Parameters r int: red component (0.255) of the color to draw on the canvas g int: green component (0.255) of color to draw on the canvas b int: blue component (0.255) of the color to draw on the canvas public void pullRec (float left, float top, float right, float bottom, Paint paint) Draws the specified Rect using the specified paint. The rectangle will be filled or framed based on the style in the paint. Parameters left float: The left side of the rectangle is pulled upper float: The upper side of the rectangle is pulled right drift: The right side of the rectangle is pulled bottom drift: The bottom of the rectangle is drawn paint: The paint used to draw the rack This value cannot be zero. public void PullAcacity(Rect r, Paint Paint) Draws the specified Rack using the specified Paint. The rectangle will be filled or framed based on the style in the paint. Parameters r Rect: The rectangle to be drawn. This value cannot be null. Paint paint: The paint used to draw the rectangle This value cannot be null. public void PullAccess(RectF rect, Paint Paint) Draws the specified Stretch using the specified paint. The rectangle will be filled or framed based on the style in the paint. Parameters rect RectF: The elastic to be drawn This value cannot be null. Paint Paint: The paint used to draw the rack This value cannot be null. public void pullRenderNode(RenderNode renderNode) draws the given RenderNode. It is supported only in hardware rendering, which can be verified by claiming that isHardwareAccelerated() is true. If isHardwareAccelerated() is false, it throws an exception. See RenderNode for more information about what a and how to use it. Parameters renderNode RenderNode: The RenderNode to sign must be non-zero. This value cannot be null. public void pullRoundRect(RectF rect, float rx, float row, Paint paint) Draws the specified round-rack using the specified paint. The will be filled or framed based on the style in the paint. Parameters rect RectF: The rectangular boundaries of the roundArredRec to be drawn This value cannot be null. rx float: The x-radius of the oval used to drive the corners drives around: The y radius of the oval used to roundride the corners paint Paint: The paint used to pull the roundRect This value may not be null. public void pullRoundRect(float left, float top, float right, float bottom, float rx, float row, Paint paint) Draw the specified round-rack using the specified paint. The roundrect will be filled or framed based on the style in the paint. Parameters left float upper drives right float rx drives: The x-radius of the oval used to drive the corners drives around: The y-radius of the oval used to round the corners paint Paint: The paint used to draw the roundRec This value cannot be null. public void tokenText(CharSequence text, int begins, int end, float x, float y, Paint paint) Draws the specified range of text, specified by start/end, with its origin at (x,y), in the specified Paint. The origin is interpreted based on the Align setting in the Paint. Parameters text CharSequence: The text to be drawn This value cannot be null. start int: The index of the first character in text to draw end intent: (end - 1) is the index of the last character in text to draw x drives: The x-coordination of origin for where to draw the text y float: The y coordinates of origin for true to draw the text paint paint: The paint used for the text (e.g. color, size, style) This value cannot be public void tokenText(String text, float x, float y, Paint paint) Draw the text, with origin at (x,y), using the specified paint. The origin is interpreted based on the Align setting in the paint. Parameters text string: The text to be drawn This value cannot be null. x float: The x-coordination of the origin of the text drawn floats y: The y coordination of the baseline of the text that paint is drawn: The paint used for the text (e.g., color, size, style) This value cannot be null. public void tokenText(char[] text, int index, int count, float x, float y, Paint paint) Draw the text, with origin at (x,y), using the specified paint. The origin is interpreted based on the Align setting in the paint. Parameters text char: The text to be drawn This value cannot be null. index int count int x float: The x-coordination of the origin of the text that is drawn y float: The y coordination of the baseline of the text that drew paint Paint: The paint used for text (e.g. color, size, style) This value cannot be null. public void tokenText(String text, int start, int end, float x, float y, Paint paint) Draw the text, with origin at (x,y), using the specified paint. The origin is interpreted based on the Align setting in the paint. Parameters text string: The text to be drawn This value cannot be null. start int: The index of the first in text to draw endint: (end - 1) is the index of the last character in text to draw x float: The x-coordination of the origin of the text that floats y: The y coordination of the baseline of the text drawn paint: The paint used for the text (e.g., color, size, style) This value cannot be null. public void pullTextOnPath(String text, Path path, float hOffset, float vOffset, Paint paint) Draw the text, with origin at (x,y), using the specified paint, along the specified path. The paint's Align setting determines where along the way to start the text. Parameters text string: The text to be drawn This value cannot be null. path path: The path that must follow the text for its baseline This value cannot be null. hOffset float: The distance along the way to add to the text's starting position vOffset float: The distance above (-) or below (+) the path to position the text paint Paint: The paint used for the text (e.g. color, size, style) This value cannot be null. public void pullTextOnPath(char[] text, int index, int count, Path path, float hOffset, float vOffset, Paint paint) Draw the text, with origin at (x,y), using the specified paint, along the specified path. The paint's Align setting determines where along the way to start the text. Parameters text char: The text to be drawn This value cannot be null. indexint: The starting index within the text to be drawn counts int: From index, the number of characters can pull path path: The path that the text must follow for its baseline This value cannot be null. hOffset float: The distance along the way to add to the text's starting position vOffset float: The distance above (-) or below (+) the path to position the text paint Paint: The paint used for the text (e.g. color, size, style) This value cannot be null. public void tokenTextRun(MeasuredText text, int start, int end, int contextStart, int contextEnd, float x, float y, boolean isRtl, Paint paint) Draw a run of text, all in a single direction, with optional context for complex text formation. See drawTextRun(java.lang.CharSequence, int, int, int, int, float, float, boolean, android.graphics.Paint) for more details. This method uses a MeasuredText rather than CharSequence to represent the string. Parameters text MeasuredText: the text to render this value cannot be null. start int: render the beginning of the text. Data before this position can be used for forming context. endint: the end of the text to render. Data at or after this position can be used for forming context. contextStart int: the index of the beginning of the formation contextEnd int: the index of the end of the formation context x float: the x at which to draw the text y drives: the y position at which the text isRtl boolean sign: whether the run is in RTL direction paint Paint: the paint This value cannot be null. public void pullTextRun(char[] text, text, index, int count, int contextIndex, int contextCount, float x, float y, boolean isRtl, Paint paint) Draw a run of text, all in a single direction, with optional context for complex text formation. See drawTextRun(java.lang.CharSequence, int, int, int, int, float, float, boolean, android.graphics.Paint) for more details. This method uses a character array rather than CharSequence to represent the string. Also, to be consistent with the pattern established in drawText(char[], int, int, float, float, Paint), in this method count and contextCount usage rather than offset the end position; count = end - start, contextCount = contextEnd - contextStart. Parameters text char: the text to render this value cannot be null. index int: the beginning of the text to render count int: the count of chars contextIndex int render: the beginning of the context for the formation. Must be no bigger than index. contextCount int: the number of characters in the context for the formation. contexIndex + contextCount should be no less than index + score. x float: the x position at which to draw the text y float: the y position at which the text isRtl to draw boolean: whether the run is in RTL directional paint Paint: the paint This value cannot be null. public void tokenTextRun(CharSequence text, int start, int end, int contextStart, int contextEnd, float x, float y, boolean isRtl, Paint paint) Draw a run of text, all in a single direction, with optional context for complex text formation. The course of text includes the characters in the text from start to finish. In addition, the range contextStart to contextEnd is used as context for the purpose of complex text formation, such as Arabic text potentially formed differently based on the text next to it. All text outside the range contextStart.. contextEnd is ignored. The text between beginning and end will be laid out and drawn. The context series is useful for contextual formation, e.g. The direction of the run is specified explicitly by isRtl. Thus, this method is suitable only for runs from a single direction. Alignment of the text is as determined by the Paint's TextAlign value. Furthermore, 0 <= contextStart <= begin < = end <= contextEnd <= text.length must stick to enrollment. Also see Paint.getRunAdvance(char[], int, int, int, int, boolean, int) for a corresponding method to measure the text; the advance width of the text drawn matches the value obtained from that method. Parameters text CharSequence: the text to render this value cannot be null. start int: render the beginning of the text. Data before this position can be used for forming context. endint: the end of the text to render. Data at or after this position can be used for forming context. contextStart int: the index of beginning of the formation contextEnd int: the index of the end of the formation context x floats: the x position at which to draw the text y the y position at which the text isRtl draws boolean: whether the run is in RTL direction paint: the paint This value cannot be null. public void pullVertical(Canvas.VertexMode mode, int vertexCount, float[] vertical, int vertOffset, float[] texs, int texOffset, int[] colors, int colorOffset, short[] indexes, int indexOffset, int indexCount, Paint paint) Draw the array of vertical, interpreted as triangle The verts array is required, and specifies the x,y pairs for each vertex. If texs is non-zero, it is used to specify the coordinate in shadow coordinates to use at each vertex (the paint must have a shader in this case). If there is no texs array, but there is a color array, then each color over its corresponding triangle is in a gradient. If both texs and colors arrays are present, then they behave as before, but the resulting color at each pixels is the result of multiplying the colors of the shade and the color-gradient together. The indexes array is optional, but if it is present, it is used to specify the index of each triangle, rather than just running through the arrays in order. Parameters off Canvas.VertexMode: How to interpret the array of verticals This value cannot be null. vertexCount int: The number of values in the vertical array (and corresponding texs and colors arrays if non-zero). Each logical vertex is two values (x, y), vertexCount must be a multiple of 2. Verts drives: Variety vertical for the mesh This value cannot be zero. vertOffset int: Number of values in the verts to skip before drawing. texs float: May be insidious. If not null, specify the coordinates to sample in the current shadow (e.g., bitmap tile or gradient) This value can be null andull. texOffset int: Number of values in texs to skip before signing. colors int: May be insular. If not null, specify a color for each vertex, to be bled across the triangle. This value can be nite. colorOffset int: Number of values in colors to skip before drawing. indexes short: If not null, array of indexes to refer to the vertex (texs, colors) array. This value can be nite. indexOffset int indexCount int: number of entries in the indexes array (if not null). Paint Paint: Specifies the shader to use if the texs array is non-zero. This value cannot be null. public void EnableZ() Enables Z support which defaults to be disabled. It allows For RenderNodes drawn with drawRenderNode (android.graphics.RenderNode) to be re-arranged based on their RenderNode #getElevation() and RenderNode#getTranslationZ() values. It also enables the delivery of shadows for RenderNodes with a height or able. Any draw reordering will not be moved before this call. A typical use of this can look something like: void draw (Canvas cloth) { // Draw any background content canvas.drawColor(backgroundColor); // Start drawing that could be based on Z canvas.enableZ(); for (RenderNode child : children) { canvas.drawRenderNode(child); } // End token that can be reordered based on Z canvas.disableZ(); Draw any overlay canvas.drawText (I'm on top of everything!, 0, 0, paint); } Note: This is not affected by any storage() or recovery() calls, as it is not considered part of the current matrix or clip. See disabledZ() public boolean getClipBounds (Rect borders) Return the boundaries of the current clip (in local coordinates) in the borders parameter and return true if it is not empty. This can be useful in a way similar to quickReject, soaking that it tells you that the drawing outside of these boundaries will be cut out. Parameters boundary Rect: Returns the clip boundaries here. If it isn't, ignore it, but still return true if the current clip isn't empty. This value can be nite. Returns boolean if the current clip is not empty. public final Rect getClipBounds() Retrieves the boundaries of the current clip (in local coordinates). Give Rect the clip boundaries, or [0, 0, 0, 0] if the clip is empty. This value cannot be null. public int getDensity() Returns the target density of the canvas. The default density is derived from the density of its backing bitmap, or Bitmap #DENSITY_NONE if there is not one. Returns int Returns the current target density of the canvas, which is used to determine the scale factor when a bitmap is pulled into it. See also: setDensity(int)Bitmap.getDensity() public int getHeight() Returns the height of the current token layer Returns int the height of the current token layer Added in API level 1 Decimated in API level 16 public void getMatrix(Matrix ctm) This method is depreated in API level 16. Hardware accelerated diaper can have any matrix when passed to a view or tokenable, since it is implementation that is defined where in the hierarchy such dians are created. It is recommended in such cases to either pull content regardless of the current matrix, or to detect relevant transform state outside the canvas. Return, in ctm, the current transformation matrix. It doesn't change the matrix in the canvas, but only returns a copy of it. Parameters ctm Matrix: This value cannot be null. Added in API level 1 Decimated in API level 16 public final Matrix getMatrix() This method is depreated in API level 16. Hardware accelerated diaper can have any matrix when passed to a view or tokenable, since it is implementation that is defined where in the hierarchy such dians are created. It is recommended in such cases to either pull content regardless of the current matrix, or to detect relevant transform state outside the canvas. Returns a new matrix with a copy of the canvas's current Returns Matrix This value cannot be null. public int getMaximumBitmapHeight() Returns the maximum allowed height for bitmaps drawn with this canvas. Try to having a bitmap longer than this value will result in an error. See also: public int getMaximumBitmapWidth() Returns the maximum allowed width for bitmaps drawn with this canvas. Attempting to draw with a bitmap wider than this value will result in an error. See also: public int getSaveCount() Returns the number of matrix/clip states on the Canvas' private stack. This will equal # storage() calls - # recovery() calls. public int getWidth() Returns the width of the current token layer Returns int the width of the current token layer of public boolean isHardwareAccelerated() Indicates whether these Canvas use hardware acceleration. Note that this method does not define what type of hardware acceleration may or may not be used. Returns boolean True if drawing operations hardware is accelerated, false otherwise. public boolean isOpaque() Return true if the device that pulls the current layer in is opaque (that is, does not support per-pixel alpha). Returns boolean true if the device that pulls the current layer in is added opaquely in API level 1 Decimated in API level 30 public boolean quickReject (float left, float top, float right, float bottom, Canvas.EdgeType type) This method is depreated in API level 30. The EdgeType is ignored. Use quickReject (float, float, float, float) instead. Returns as the specified rectangle, having been converted by the current matrix, would lie completely outside the current clip. Call it to check if an area you intend to move in is cut out (which is why you can skip the draw calls). Parameters left float: The left side of the rectangle to compare to the current clip top drives: The top of the rectangle to compare to the current clip right float: The right side of the rectangle to compare with the current clip bottom drives: The bottom of the rectangle to compare with the current clip type Canvas.EdgeType: Canvas.EdgeType #AA as the path should be considered antialiased, since this means it can affect a larger area (more pixels) than non-antialiased (Canvas.EdgeType #BW). This value cannot be null. Returns boolean if the rack (converted by the canvas' matrix) does not cross with the canvas's clip public boolean quickReject (Padpad) Returns true as the specified path, having been converted by the current matrix, would lie completely outside the current clip. Call it to check if an area you intend to move in is cut out (which is why you can skip the draw calls). Note: for speed, it can return false, even if the road itself might not cross the clip (i.e. cross the boundaries of the road, but the road does not cross). Parameters path path: The path to compare with the current clip This value cannot be null. Give boolean true as the road (converted by the canvas's not cross with the canvas's clip public boolean quickReject (RectF rect) Return true true the specified rectangle, having been converted by the current matrix, would lie completely outside the current clip. Call it to check if an area you intend to move in is cut out (which is why you can skip the draw calls). Parameters rect RectF: the elastic to compare to the current clip This value cannot be null. Returns boolean true if the elastic (converted by the canvas's matrix) does not cross with the canvas's clip Added into API level 1 Decimated in API level 30 public boolean quickReject(RectF rect, Canvas.EdgeType type) This method is depreciated in API level 30. The EdgeType is ignored. Use quickReject instead (android.graphics.RectF). Returns as the specified rectangle, having been converted by the current matrix, would lie completely outside the current clip. Call it to check if an area you intend to move in is cut out (which is why you can skip the draw calls). Parameters rect RectF: the elastic to compare to the current clip This value cannot be null. Type Canvas.EdgeType: Canvas.EdgeType #AA if the path should be considered antialiased as it means it can affect a larger area (more pixels) than non-antialiased (Canvas.EdgeType #BW). This value cannot be null. Returns boolean true if the elastic (converted by the canvas's matrix) does not cross with the canvas's clip Added into API level 1 Decimated in API level 30 public boolean quickReject(Path Path, Canvas.EdgeType type) This method is depreciated in API level 30. The EdgeType is ignored. Instead, use quickReject(android.graphics.Path) Returns as the specified path, after being converted by the current matrix, would lie completely outside the current clip. Call it to check if an area you intend to move in is cut out (which is why you can skip the draw calls). Note: for speed, it can return false, even if the road itself might not cross the clip (i.e. cross the boundaries of the road, but the road does not cross). Parameters path path: The path to compare with the current clip This value cannot be null. Type Canvas.EdgeType: Canvas.EdgeType #AA if the path should be considered antialiased as it means it can affect a larger area (more pixels) than non-antialiased (Canvas.EdgeType #BW). This value cannot be null. Returns boolean where if the path (transformed by the canvas's matrix) does not cross with the canvas's clip public boolean quickReject (float left, float top, float right, float bottom) Return true as the specified rectangle, having been converted by the current matrix, would lie completely outside the current clip. Call it to check if an area you intend to move in is cut out (which is why you can skip the draw calls). Parameters left drives: The left side of the rectangle to compare to the current clip top drives: The from the rectangle to compare to the current clip on the right The right side of the rectangle to compare to the current cut bottom float: The bottom of the rectangle to compare to the current clip Returns boolean where if the rack (converted by the canvas' matrix) does not cross with the canvas 's clip public void recovery() This call balances a previous call to save (), and is used to remove all changes to the matrix/clip This is an error calling recovery() more times than what storage() was called. public void restoreToCount(int saveCount) Efficient way to pop any calls to save() that happened after the storage count reached SaveCount. It's an error for saveCount to be less than 1. Example: int count = canvas.save(); ...// more calls potentially to save() canvas.restoreToCount(count); now the canvas is back in the same state it was before the initial // call to save (). Parameters saveCount int: The storage tier to restore to. public void rotate (float grades) Preconcat the current matrix with the specified rotation. Parametersgrade drives: The amount to rotate, in degrees public final void turn (float grades, float px, float py) Preconcat the current matrix with the specified rotation. Parameters grades float: The amount to rotate, in degrees px float: The x-co-op for the pivot (unchanged by the rotation) py float: The y-co-op for the pivot (unchanged by the rotation) public int save () Save the current matrix and clip onto a private stack. Subsequent calls to translate, scale, twist, skew, concat or clipRect, clipPath will all function as usual, but when the balancing call to restore () is made, those calls will be forgotten, and the settings that existed before storage() will be reinstated. Return int The value to pass to RestoreToCount() to balance this storage() Added in API level 1 Decimated in API level 26 public int saveLayer(RectF boundaries, Paint paint, int saveFlags) This method is depreciated in API level 26. Use SaveLayer instead (android.graphics.RectF, android.graphics.Paint). It acts the same as save(), but beyond that it assigns and redirects drawing to an offscreen bitmap. Note: this method is very expensive, incurs more than double delivery charges for contained content. Avoid using this method, especially if the boundaries provided are large. It is recommended to use a hardware layer on a view to apply an xfermode, color filter, or alpha, as it will perform much better than this method. All drawing calls are directed to a newly assigned offscreen bitmap. Only when the balancing call to restore () is that offscreen buffer is withdrawn to the current target of the Canvas (either the screen, it's target Bitmap, or the previous layer). Characteristics of the Paint - Paint #getAlpha(), Paint and Paint #getColorFilter() is applied when the offscreen bitmap is withdrawn when repair() is called. From api-level API level Build.VERSION_CODES. P the only valid storageFlags is ALL_SAVE_FLAG. All other flags are ignored. Parameters border RectF: May be nint. The maximum size that the offscreen bitmap should be (in local coordinates) This value can be nrupted. paint Paint: It is copied, and is applied to the shield when repair() is called. This value can be nite. saveFlags int: _SAVE_FLAG new constants, over the ALL_SAVE_FLAG it is recommended for performance reasons. Value is either 0 or ALL_SAVE_FLAG Returns int value to pass toToCount() to restore this storage() public int saveLayer(RectF boundaries, Paint paint) It behaves the same as storage(), but in addition it assigns and redirects drawing to an offscreen rendering target. Note: this method is very expensive, incurs more than double delivery charges for contained content. Avoid using this method whenever possible and instead use a hardware layer on a view to apply an xfermode, color filter, or alpha, as it will perform much better than this method. All token calls are directed to a newly assigned offscreen rendering target. Only when the balancing call to restore() is that offscreen buffer is withdrawn to the current target of the Canvas (which could potentially be a previous layer if these calls are nested). Properties of the Paint - Paint #getAlpha(), Paint #getXfermode(), and Paint #getColorFilter() are applied when the offscreen rendering target is revoked when restore() is called. Parameters border RectF: May be nint. The maximum size that the offscreen render target should be (in local coordinates) This value can be nite. paint Paint: It is copied, and is applied to the shield when repair() is called. This value can be nite. Returns int value to restoreToCount() to balance this storage() Added in API level 1 Decimated in API level 26 public int saveLayer(float left, float top, float right, float bottom, Paint paint, int saveFlags) This method is depreciated in API level 26. Use saveLayer (float, float, float, float, android.graphics.Paint) instead. Helper version of saveLayer() that takes 4 values rather than a RectF. From which API level API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P the only valid storageFlags is ALL_SAVE_FLAG. All other flags are ignored. Parameters left float upper drives right float bottom drives paint: This value cannot be. saveFlags int: Value is either 0 or ALL_SAVE_FLAG Added in API level 1 Decimated in API level 26 public int saveLayerAlpha (float left, float top, float right, float bottom, int alpha, int saveFlags) This method is depreciated in API level 26. Use saveLayerAlpha (float, float, float, float, int) instead. Helper for saveLayerAlpha() that takes 4 values instead of a RectF. Proven API Level P Build.VERSION_CODES. P the only valid storageFlags is ALL_SAVE_FLAG. All other flags are ignored. Parameters remaining top float right float alpha int redFlags int: Value is either 0 or ALL_SAVE_FLAG public int saveLayerAlpha (RectF borders, int alpha) Convenience for saveLayer(android.graphics.RectF, android.graphics.Paint) but instead of taking a whole paint object it only takes the alpha parameter. Parameters border RectF: The maximum size that the offscreen bitmap must be (in local coordinates) This value can be nint. alpha int: The alpha to apply to the offscreen when it is drawn during recovery () Added in API level 1 Decimated in API level 26 public int saveLayerAlpha (RectF boundaries, int alpha, int saveFlags) This method is depreated in API level 26. Use SaveLayerAlpha instead (android.graphics.RectF, int). It acts the same as save(), but beyond that it assigns and redirects drawing to an offscreen bitmap. Note: this method is very expensive, incurs more than double delivery charges for contained content. Avoid using this method, especially if the boundaries provided are large. It is recommended to use a hardware layer on a view to apply an xfermode, color filter, or alpha, as it will perform much better than this method. All drawing calls are directed to a newly assigned offscreen bitmap. Only when the balancing call to restore () is that offscreen buffer is withdrawn to the current target of the Canvas (either the screen, it's target Bitmap, or the previous layer). The alpha parameter is applied when the offscreen bitmap is revoked when repair() is called. From which API level API android.os.Build.VERSION_CODES. P Build.VERSION_CODES. P the only valid storageFlags is ALL_SAVE_FLAG. All other flags are ignored. Parameters border RectF: The maximum size that the offscreen bitmap must be (in local coordinates) This value can be nint. alpha int: The alpha to apply for the offscreen when drawn during recovery () redFlags int: see _SAVE_FLAG constants, generally ALL_SAVE_FLAG is recommended for performance reasons. Value is either 0 or ALL_SAVE_FLAG Returns int value to pass toToCount() to restore this call public int saveLayerAlpha (float left, float top, float right, float bottom, int alpha) Convenience for saveLayerAlpha (android.graphics.RectF, int) that takes the four float coordinates of the borders rectangle. Parameters left float upper drives bottom drives under float alpha int public final void scale (float sx, float silk, float px, float py) Preconcat the current matrix with the specified scale. Parameters sx drives: The amount to scale in X its float: The amount to scale in Y px float: The x-co-op for the pivot (unchanged by the scale) py float: The y-co-op for the pivot (unchanged by the scale) public void scale (float sx, float its) Preconcat the current matrix with specified scale. Parameters sx drives: The amount to scale in X its drives: The amount to scale in Y public void setBitmap (Bitmap bitmap) Specifies a n for the canvas to move in. All canvas state such as layers, filters, and the storage/recovery stack are reset. In addition, the canvas's target density is updated to match that of the bitmap. Before API android.os.Build.VERSION_CODES. Oh Build.VERSION_CODES. O the current matrix and clip stack has been preserved. Parameters bitmap Bitmap: Specifies a stable bitmap for the canvas to pull in. This value can be nite. See also: setDensity(int)getDensity() public void setDensity(int density) Specifies the density for this Canvas's backing bitmap. This changes the target density of the canvas itself, as well as the density of its backing bitmap via Bitmap #setDensity (int). Parameters density int: The new target density of the canvas, used to determine the scale factor when a bitmap is pulled into it. Use Bitmap #DENSITY_NONE eliminate bitmap scale. See also: getDensity()Bitmap.setDensity(int) public void setDrawFilter(DrawFilter filter) Parameters filter DrawFilter: This value can be nint. public void skew (float sx, float its) Preconcat the current matrix with the specified skew. Parameters sx drives: The amount to skew in X side drives: The amount to skew in Y public void translate (float dx, float dy) Preconcat the current matrix with the specified translation Parameters dx float: The distance to translate into X thigh drives: The distance to translate into Y Y