Graphics

TrelGraphics2

-textures: TrelTexture*
-size: int
-index: int
-gWindow: SDL_Window*
-gRenderer: SDL_Renderer*
-windowWidth: int
-windowHeight: int

+start(): void
+close(): void
+TrelGraphics2( )
+createPictureFromFile( ): int
+createPictureFromFileColor( ): int
+drawPicture ( ):void
+addPictureToFrame(): void
+addPictureToFrameResize(): void
+addPictureToFrameRotation(): void
+addRectToFrame (): void
+drawFrame(): void
+clearScreen():void
+clearFrame ():void
+getArraySize(): int
+getNumberOfImages (): int
+getImageWidth (): int
+getImageHeight (): int
+setPictureColor(): void

Trel_Texture

-picWidth: int
-picHeight: int
-picTexture: SDL_Texture*

+TrelTexture( )
+~TrelTexture( )
+loadFromFile( ): bool
+loadFromFileColor( ): bool
+free( ): bool
+render( ): void
+renderResize( ): void
+getWidth( ): int
+getHeight( ): int
+setColor( ): void

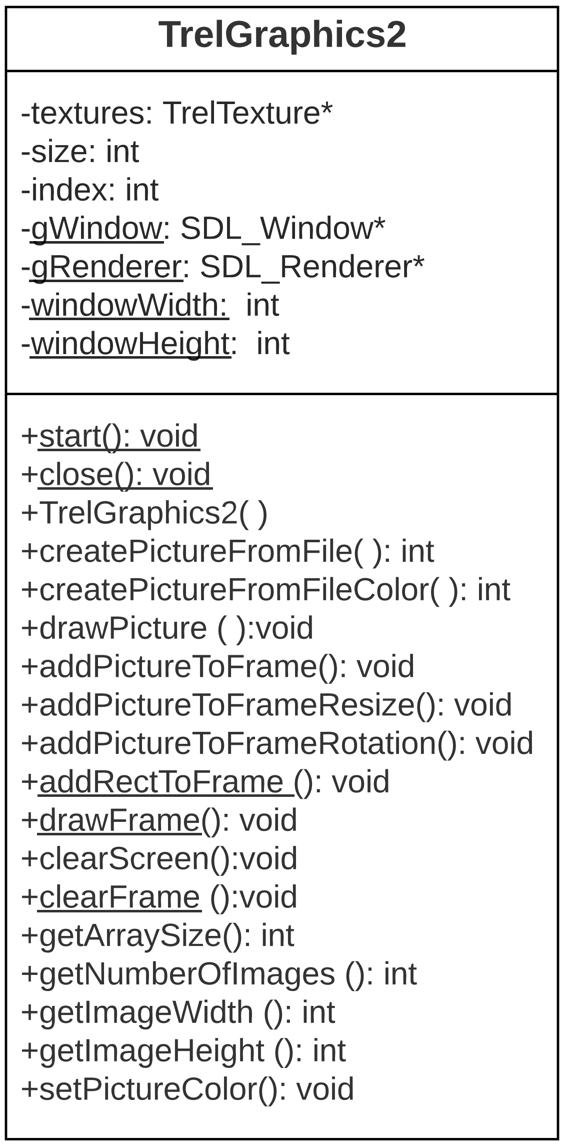
DrawGameBoard

+drawGameBoard(): void

UML Key

Top Box: Class name
Middle: Class variables
Bottom: Class methods

Public (+)
Private (-)
Protected (#)
Derived (/)
Static (underlined)

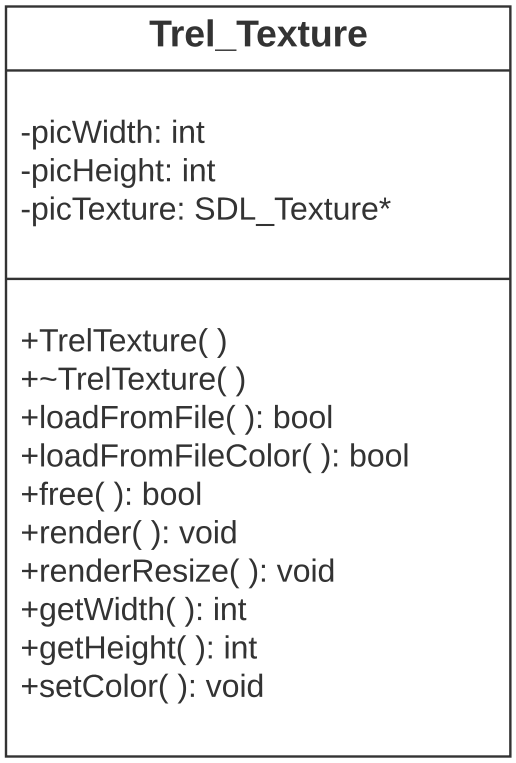


TrelGraphics2

-textures: TrelTexture*
-size: int
-index: int
-gWindow: SDL_Window*
-gRenderer: SDL_Renderer*
-windowWidth: int
-windowHeight: int

+start(): void
+close(): void
+TrelGraphics2( )
+createPictureFromFile( ): int
+createPictureFromFileColor( ): int
+drawPicture ( ):void
+addPictureToFrame(): void
+addPictureToFrameResize(): void
+addPictureToFrameRotation(): void
+addRectToFrame (): void
+drawFrame(): void
+clearScreen():void
+clearFrame ():void
+getArraySize(): int
+getNumberOfImages (): int
+getImageWidth (): int
+getImageHeight (): int
+setPictureColor(): void

```
#pragma once
#include<iostream>
#include<SDL.h>
#include<vector>
#include<string>
#include<fstream>
#include"TrelTexture.h"
usingnamespace std;

// A graphics object designed to handle the details of running SDL2.
// start() and close() methods MUST be called at the start and end of main
// and main MUST have the parameters (int argc, char* args[]) and return an int.
class TrelGraphics2
private:
 TrelTexture *textures;  // the array for loaded pictures
 int size;  // the size of the textures array
 int index;  // assigned to be the number of the left most EMPTY array space.
 static SDL_Window* gWindow;  // the window loaded and printed to
 static SDL_Renderer* gRenderer;  // the renderer that handles texture printing
 staticint windowWidth;    // width of window in pixels
 staticint windowHeight;         //height of window in pixels

public:
 // Initializes SDL, Opens a window and sets window title/width/height, and starts the renderer, must be called at the start of main.
 static void start( string windowTitle, int width, int height);
 // closes renderer, window, and SDL and frees their memory, must be called at the end of main.
 static void close( );
 // Creates a TrelGraphics2 object and assigns memory for size number of images.
 TrelGraphics2(int size);

 // Creates a TrelGraphics2 object, reads an image list text file, assigns memory for them all and loads them all.
 TrelGraphics2( string fileName );

 // Creates a TrelGraphics2 object, reads an image list text file, assigns memory for them all and loads them all.
 // Then color keys them to given colors,
 TrelGraphics2( string fileName, Uint8 r, Uint8 g, Uint8 b );

 // Closes the TrelGraphics2 object and deletes all the assigned memory, must be called at the end of the program
 ~TrelGraphics2( );

 // creates a picture from a file, and returns it's pictureID, can only load .bmp files
 int createPictureFromFile( string fileName );

//same as createPictureFromFilebut will set the color key for transparency to the given r,g,b values.
// r,g,b must be from 0 to 255.
 int createPictureFromFileColor( string fileName, Uint8 r, Uint8 g, Uint8 b );

 // reads a file and opens images writen in that file until file is done or assigned memory is reached
 void readImageListFromFile( string fileName );

//same as readImageListFromFile(string) but will set the color key for transparency to the given r,g,b values.
// r,g,b must be from 0 to 255
 void readImageListFromFileColor( string fileName, Uint8 r, Uint8 g, Uint8 b );

 // draws a picture at position pictureID in the vector to screen location (x,y)
 // Do not use in conjunction with addPictureToFrame() and drawFrame()
 void drawPicture(in tpictureID, int x, int y );

 // adds a picture to a frame, but does not draw it, call drawFrame() to draw all pictures on frame.
 void addPictureToFrame(int pictureID, int x, int y );

 // adds a picture to a frame, but does not draw it, call drawFrame() to draw all pictures on frame.
 void addPictureToFrameResize(int pictureID, int x, int y, int w, int h );

 // adds a picture to a frame, but does not draw it, call drawFrame() to draw all pictures on frame.
 // Allows for roation and flipping of image.
 void addPicturetoFrameRotation(int pictureID, int x, int y, double degrees, bool vFlip, bool hFlip );

 // adds a rectangle to the frame with the given color and transparancy
 static void addRectToFrame(int x, int y, int w, int h, Uint8 r, Uint8 g, Uint8 b, Uint8 a, bool filled=true);

 // draws frame to screen
 static void drawFrame( );

 // clears the screen to a white image
 void clearScreen( );

 // clears the frame to a white image
 static void clearFrame( );

 // returns number of images allowed to be created
 int getArraySize( );

 // returns number of loaded images
 int getNumberOfImages( );

 // returns width of image at pictureID
 int getImageWidth(int picutreID );

 // returns hight of image at pictureID
 int getImageHeight(int pictureID );

//set an additional color value multiplied into render copy operations
//of the specified(from pictureID) texture
 void setPictureColor(intpictureID, Uint8 red, Uint8 green, Uint8 blue );
};
```



Trel_Texture

```
-picWidth: int
-picHeight: int
-picTexture: SDL_Texture*
```

```
+TrelTexture( )
+~TrelTexture( )
+loadFromFile( ): bool
+loadFromFileColor( ): bool
+free( ): bool
+render( ): void
+renderResize( ): void
+getWidth( ): int
+getHeight( ): int
+setColor( ): void
```

```
#pragma once
#include<SDL.h>
#include<string>
#include<iostream>
usingnamespace std;


// A class designed to handle the details of texture use in SDL2
class TrelTexture
{
private:
 int picWidth;          //stores the width of the loaded image
 int picHeight;         ////stores the height of the loaded image
 SDL_Texture* picTexture;      // The actual texture hardware

public:
 // Constructor
 TrelTexture( );

 // Deallocates memory
 ~TrelTexture( );

 // loads image from file path
 bool loadFromFile( string fileName, SDL_Renderer* gRenderer );

//load image from file path
//Also sets the color key of the pixel that needs to be transparent
 bool loadFromFileColor( string fileName, SDL_Renderer* gRenderer, Uint8 r, Uint8 g, Uint8 b );

 // Deallocates texture
 void free( );

// renders texture at given location
// x,y = position where top left corner of texture will be placed
//  The rest of the parameters will not need to be changed.
 void render(int x, int y, SDL_Renderer* gRenderer, double angle=0, SDL_RendererFlip
flip=SDL_FLIP_NONE, SDL_Point* center =NULL);

//resize an image and renders texture at given location
//x,y = position where top left corner of texture will be placed
// w,h the width/height to resize the image to
////  The rest of the parameters will not need to be changed.
 void renderResize(int x, int y, SDL_Renderer* gRenderer, int w, int h, double angle =0,
SDL_RendererFlip flip = SDL_FLIP_NONE, SDL_Point* center =NULL);

 // gets image dimentions
 int getWidth( );
 int getHeight( );

//set an additional color value multiplied into render copy operations of the specified(from pictureID)
//texture
 void setColor( Uint8 red, Uint8 green, Uint8 blue );

};
```

DrawGameBoard

+drawGameBoard: void

```
#include"TrelGraphics2_1.h"

// Concept for brick break draw method
// This can be finished when Logic writes the object classes for:
                                                   // Ball
                                                   // Paddle
                                                   // Bricks
// This is the only method that will need to be called in main for drawing everything to the
// screen during game play. Drawing the power up to screen will use methods from the
//      TrelGraphic2      class.

// global variables
TrelGraphics2ballPictures("ballpictures.txt");
TrelGraphics2paddlePictures("paddlepictures.txt");
TrelGraphics2brickPictures("brickpictures.txt");

void drawGameBoard( std::vector<Ball> balls, intx, inty, Paddle paddle, intx, inty,
std::vector<Brick> bricks )
{
 // these numbers are just there to exist, obviously will be changed to match actual values.
 // plenty of the math will be adjusted based on exact nature of some numbers and values.
 conststaticintBALL_WIDTH =20;
 conststaticintBALL_HEIGHT =20;
 conststaticintPADDLE_WIDTH =20;
 conststaticintPADDLE_HEIGHT =20;
 conststaticintBRICK_WIDTH =20;
 conststaticintBRICK_HEIGHT =20;
 int x, y;
 for( Ball ball : balls )
 {
  x = ball.getCenterX( )-BALL_WIDTH;
  y = ball.getCenterY( )-BALL_HEIGHT;
  ballpictures.addPictureToFrameRotation(0, x, y, ball.getDirection( ),false,false);
  // doesn't include proper centering yet, will add that in final version
 }
 x = paddle.getCenterX( ) - PADDLE_WIDTH;
 y = paddle.getCenterY( ) - PADDLE_HEIGHT;
 paddlePictures.addPictureToFrameRotation(0, x, y, paddle.getDirection( ),false,false);
 for( Brick brick : bricks )
 {
  x = brick.getCenterX( ) - BRICK_WIDTH;
  y = brick.getCenterY( ) - BRICK_HEIGHT;
  brickPictures.addPictureToFrameRotation(0, x, y, brick.getDirection( ),false,false);
 }
// presents everything rendered to screen by calling SDL_RenderPresent( gRenderer );
 TrelGraphics2::drawFrame( );
}
```