3DxFrameWork Documentation

Student authors and contributors to software:

**Nathan Harbor - 2008**

**Heath Wickman - 2009**

**Cameron Ferguson - 2009**

Table of Contents

[Overview 2](#_Toc255328612)

[Methods 2](#_Toc255328613)

[General Methods 2](#_Toc255328614)

[Triangle Methods 3](#_Toc255328615)

[Other Shape Methods 5](#_Toc255328616)

[MyGameWorld::KeyPress(WPARAM keyCode) 5](#_Toc255328617)

[keyCode List 5](#_Toc255328618)

[gspOut 6](#_Toc255328619)

# Overview

Any code that needs to be called at the beginning of the program should be placed in the MyGameWorld::Initialize() method. Code that needs to be called at the end is placed in the destructor. Any code that needs to be executed during the main game/simulation loop goes inside MyGameWorld::Update. Additional methods may be created in the MyGameWorld class, just as long as the names and parameter lists of the existing variables and methods are not altered.

There is a global MyGameWorld object ( gMyGameWorld ) that can be used to call the below methods outside of the MyGameWorld class itself. Any calls made using this global object will be reflected in the main game world. To use gMyGameWorld in other files, #include “MyGameWorld.h”

* There are 4 types of shapes that you can draw using the 3DxFrameWork Documentation: Lines, Spheres, Triangles, and Squares.
* User-input is received via the mouse and by key presses.
* A graphical debug window is available to help you easily output numerical values and messages while the program is running to help resolve logic errors.
* This framework was put together and enhanced overtime by DeVry GSP students. Further enhancements are ENCOURAGED.
* This documentation is probably out-of-date. So any mistakes, problems you find, please send comments, questions, and corrections to cnicholson@devry.edu

## 

# Methods

## General Methods

#### void PauseWorld()

* This method will pause the gameworld, preventing it from calling its update method until un-paused.
* The gameworld will continue to be rendered to the screen, and will continue to accept mouse and keyboard input (including the MyGameWorld::KeyPress method.)
* Calling the method again will un-pause the gameworld.

#### void SetBackground(number R, number G, number B)

* Sets the background color to the given RGB values.

#### void SetBackgroundRed(number R)

* Sets the background color’s red component to the given value.

#### void SetBackgroundGreen(number G)

* Sets the background color’s green component to the given value.

#### void SetBackgroundBlue(number B)

* Sets the background color’s blue component to the given value.

#### int BackgroundRed()

* Returns the background color’s red component value.

#### int BackgroundGreen()

* Returns the background color’s green component value.

#### int BackgroundBlue()

* Returns the background color’s blue component value.

#### void SetCameraDistance(number distance)

* Sets the camera’s distance from the scene to the specified value (minimum 1.0.)

#### void ChangeCameraDistance(number deltaZ)

* Adds the given value to the camera’s distance.

#### float CameraDistance()

* Returns the distance of the camera from the scene.

## Triangle Methods

#### int CreateTriangle(int id)

* Creates a new Triangle, and sets its ID to the given value.
* If id is not specified, the Triangle will be given the next available ID.
* Different shapes do not share IDs. (It is possible to have a Triangle, Square, and Sphere, all with the same ID number.)
* The return value is the ID of the Triangle. (Helpful when the ID of the Triangle is not specified.)

#### void DestroyTriangles()

* Deletes all Triangles.
* DestroyTriangles no longer needs to be called in the destructor. This is done automatically when exiting the program.

#### void RemoveTriangle(unsigned int TriangleID)

* Deletes the specified Triangle.

#### void SetTriangleScale(unsigned int TriangleID, number scale)

* Sets the Triangle's scale.

#### float GetTriangleScale(unsigned int TriangleID)

* Returns the Triangle's scale.

#### void SetTrianglePosition(unsigned int TriangleID, number x, number y)

* Sets the location of the Triangle.

#### float GetTrianglePositionY(unsigned int TriangleID)

* Returns x value of the specified Triangle’s position.

#### float GetTrianglePositionY(unsigned int TriangleID)

* Returns y value of the specified Triangle’s position.

#### void SetTriangleHeading(unsigned int TriangleID, number angle)

* Sets the direction the specified Triangle points.
* Angle should be in radians.

#### float GetTriangleHeading(unsigned int TriangleID)

* Returns the specified Triangle’s heading (in radians.)

#### void SetTriangleColor(unsigned int TriangleID, int R, int G, int B)

* Sets the specified Triangle’s color to the given RGB values.

#### void AddTriangleText(char\* str, bool newline = false)

* Adds the specified text to the Triangle.
* The text appears directly above the center of the Triangle. (May be hard to read if the Triangle’s scale is large)
* newLine is an optional bool value that, if true, increases the distance above the Triangle that the text is drawn at by one line. Defaults to false if not specified.

#### void AddTriangleText(char\* str, number n, bool newLine = false)

* Adds the specified text followed by the number specified.
* The text appears directly above the center of the Triangle. (May be hard to read if the Triangle’s scale is large and it’s color is dark)
* Number may be an int, float or double.
* newLine is an optional bool value that, if true, increases the distance above the Triangle that the text is drawn at by one line. Defaults to false if not specified.

#### int TriangleCount()

* Returns the number of Triangles that currently exist.

#### int GetLastTriangleID()

* Returns the id of the last Triangle created.

## Other Shape Methods

The other shapes, Lines, Squares, and Spheres each have similar functionality as listed for the Triangle methods. The difference being that the names are changed accordingly. For example, the four versions of “Create” are listed in the following table. The other functions follow a similar pattern.

|  |  |  |  |
| --- | --- | --- | --- |
| Triangle | Line | Sphere | Square |
| int CreateTriangle(int id) | int CreateLine(int id) | int CreateSphere(int id) | int CreateSquare(int id) |

## MyGameWorld::KeyPress(WPARAM keyCode)

* This method will be called any time a key is pressed. The WPARAM keyCode can be treated like a char when checking which key was pressed (NOT case-sensitive.)
* Ex: if ( keyCode == ‘a’ ) 🡨 will detect if the a key was pressed.
* Keyboard input will continue to be recognize even when the program is paused with PauseWorld().

# keyCode List

* Here is a list of keyCodes for some of the keys that do not have a standard character representation:
* Ex: if ( keyCode == VK\_RIGHT ) 🡨 will detect if right arrow key was pressed.
  + VK\_BACK
  + VK\_TAB
  + VK\_RETURN
  + VK\_ESCAPE
  + VK\_SPACE
  + VK\_LEFT
  + VK\_UP
  + VK\_RIGHT
  + VK\_DOWN
  + VK\_NUMPAD0
  + VK\_NUMPAD1
  + VK\_NUMPAD2
  + VK\_NUMPAD3
  + VK\_NUMPAD4
  + VK\_NUMPAD5
  + VK\_NUMPAD6
  + VK\_NUMPAD7
  + VK\_NUMPAD8
  + VK\_NUMPAD9
  + VK\_MULTIPLY
  + VK\_ADD
  + VK\_SUBTRACT
  + VK\_DECIMAL
  + VK\_DIVIDE
  + VK\_F1
  + VK\_F2
  + VK\_F3
  + VK\_F4
  + VK\_F5
  + VK\_F6
  + VK\_F7
  + VK\_F8
  + VK\_F9
  + VK\_F10
  + VK\_F11
  + VK\_F12

# gspOut

This is the output debugger window. If you have any problems with this, or have any new ideas for it,

please let Cameron Ferguson know.

VERSION INFO: Current Version 1.0.0

VERSION HISTORY:

V1.0.0 - debut of the program. Basic output of text in the file.

INSTALLATION AND USE:

This program will work with any windows API program that has a message loop.

For GSP at DeVry, main uses will be in:

* PGraphics
* MovingOBjects
* My3dGameWorld (by Dustin)

1. Include the gspOut.h file in your projects and add to your project.
2. Instantiate a gspOut object with the 2 parameters: BOOL,STRING
   1. The first parameter will show or hide the window when the program is executed
   2. The second parameter is the title of the window.
   3. For example: **gspOut myObject(true, "My Window Title");**
   4. You can create multiple objects in different files to have your output separated into multiple windows.
3. Use the object just as you would a cout statement.

For example: **myObject << "This is my text" << endl << 98\*4/2 << "this is more text";**