3D Globe Overview

The only class you have to use to implement the globe into a project is EAGLView. You need to include Hemisphere.h and GlobeConstants.h but both have been separated out for convenience and could be included in EAGLView.h

You need the Hemisphere.h that contains the 3D model and the texture map. Hemisphere is only for one half of the Globe model, which is then used twice.

You will need to use one of the included texture files or use one of your own for the actual texture of the globe.

You also need the GlobeConstants.h file that contains many of the constants that you can modify such as maximum zoom level. The speed of the rotation, etc.

The SmoothRotator class is a convenience class that creates a smooth rotation between two points. This class is only need if you are programmatically creating rotations. If the globe will only move when a user touches it this class can be left out.

The Textures and GlobeMarker classes are used to draw markers on the globe. And MarkerForGlobe and MarkersForGlobeList use these classes for the demo example. If markers are not going to be drawn on the globe than none of these classes need to be included.

Textures

Technically you can use any images you want. The code is expecting two 1024x1024 images one for each hemisphere. The maximum texture size on device prior to the 3GS and third gen Touch is 1024. On new devices including the iPad the maximum texture size has been raised to 2048. The code includes logic than can automatically switch between textures based on the device being used.

List of textures for the globe you see in the demo:

- **GlobeWithCountries** is used to create the globe that has the country names and boundaries on it.

- **PlainGlobe** is the version of the globe without the names and boundaries on it.
- **NightTimeGlobe** is used to create the night time version
- **OldWorld** is the antique looking map
- **Stars** is used to create the stars.
- glow1.jpg, glow2.jpg, glow3.jpg images create the glowing effect. The three images are randomly drawn for each marker on the globe, which gives the “glow” effect.

Other notes about the texture files.

The code can use jpegs, pngs images or images that have been converted to pvr textures. By default the code uses pvrs since there is a reduced memory footprint. There is a constant at the top of the EAGLview class to switch between using pvr or image based textures. The stars are drawn on the inside of a second sphere. Right now we draw the stars twice using the same image so if you looked close enough you might be able to tell it is the same image twice although personally I can't see it. You could create a custom star file if you wanted to create a different effect such as larger stars, more or less stars, constellations, etc. The second sphere rotates a slightly different rate than the globe so you get the feeling of 3D depth.

If you don't need multiple version of the globe then remove the ones you are not using since each set is almost 3 MB and adds a lot to your apps overall size.

Classes

EAGLView

This is the main class that does all the work. The easiest way to integrate the Globe into an app is to add a UIView to a ViewController class in Interface Builder or programmatically. The embed UIView can be full screen if you want or it could just take up a smaller portion of the view if it is being embed alongside other elements that make up the overall page.

Key functions in EAGLView:

```
// Add a globe marker to the surface of the globe.
- (void) addGlobeMarker:(GlobeMarker*)theMarker;

// This function is called by "addGlobeMarker" which translates
the GPS coordinates
// of the "GlobeMarker" and translates them into x,y,z 3D space
- (void) addMarkerX: (float)xIn Y: (float) yIn Z: (float) zIn;

// Begins the animation process.  This MUST be called for the
globe to be interactive
- (void) startAnimation;

// Stops the animation.  This is useful if you want to pause the
animation while you transfer
// to a different view if this class will still be in memory.
- (void) stopAnimation;

// This is the main draw loop. This is where the majority of the
opengl rendering code is
- (void) drawView;

// This function loads a texture by name and calls one of the two
functions below based
// on if PVR or PNG images are being used
- (void) loadTexture: (int)index Name: (NSString*) name;

// PVR Textures take half the memory that using PNGs do.  The PVR
must be created prior
// to distribution or downloaded dynamically since you can't
create them on the fly
- (void) loadPVRTexture: (int)index Name: (NSString*) name;

// Requires more memory but allows for dynamically created images
to be used as the globe texture
- (void) loadPNGTexture: (int)index Name: (NSString*) name;

// main function call to draw the globe markers
- (void) drawMarkers;

// call defines the render effect. If you had different marker
types this function would be called
// for example if you had push pins, glowing dots, flags, etc
this could switch between them
- (void) setGlobeMarkerRenderEffect: (GlobeMarkerRenderEffect)
effect;

// The "GlobeMarker" contains a GPS location.  This function uses
the SmoothRotator class to perform
```

```
// a smooth "Google Earth" type rotation to that point on the
globe
- (void) lookAtMarker:(GlobeMarker*)theMarker;

// This function is called from the touch event to translate the
x,y location of a persons finger
// on the screen to x,y,z coordinates and also the GPS coords of
that touch
- (BOOL) raySphereIntersectX:(float)x andY:(float)y
andVector:(Vector3 *)loc;

// Initializes the Marker List
- (void) initializeTheMarkerList;

// Helper function to get a random marker
- (void) getAsingleMarker;

// Uses the built in MKReverseGeocoder class to translate the
users touch in Country,State,City data
- (void) reverseGeocodeLocation:(CLLocation*)theLocation;

// Pushes a Google Map view onto the screen matching dropping a
pin on the location matching the users
// touch on the 3D globe
- (void) showMarkerOnGoogleMaps:(MKPlacemark*)thePlace;

// Helper function used for the iPad to rotate between landscape
and portrait
- (void)
initGlobeSizeByOrientation:(UIInterfaceOrientation)interfaceOrien
tation;
```

Textures

This class is used to create a static reference to any textures that are needed. For example in the globe there are glow markers. The texture can be any 2D graphic. It could be a car or a plane or a little stick person. As the globe is rendered it will call this class to draw the markers on the class. By using a static reference the texture only has to be loaded into memory once. Since it is a 2D texture to keep from seeing the edge of the object as the globe rotates the texture is dynamically rotated to always face forward.

The purpose of this class is to allow clients to add in simple custom 2D Textures to be rendered as markers on the Globe.

GlobeMarker

This is the container class for Globe Markers. Inside of EAGLView an array of 5000 GlobeMarkers is created to hold the markers. This class could be combined with the “MarkerForGlobe” but to keep it separate from a client specific data model it has been separated.

MarkerForGlobe

This class is mostly for the demo. If markers are being used on the globe it should be replaced with the client specific data model. Ex. Cities, airports, hotels, countries, etc.

MarkersForGlobeList

This class is again an example for the demo. It is just an array of “MarkerForGlobe” that is used for the demo.

Hemisphere.h

The 3D model and texture maps. This file should never be modified.

GlobeConstants.h

A lot of the constant are setup in GlobeConstants.h

A couple that are commonly modified are:

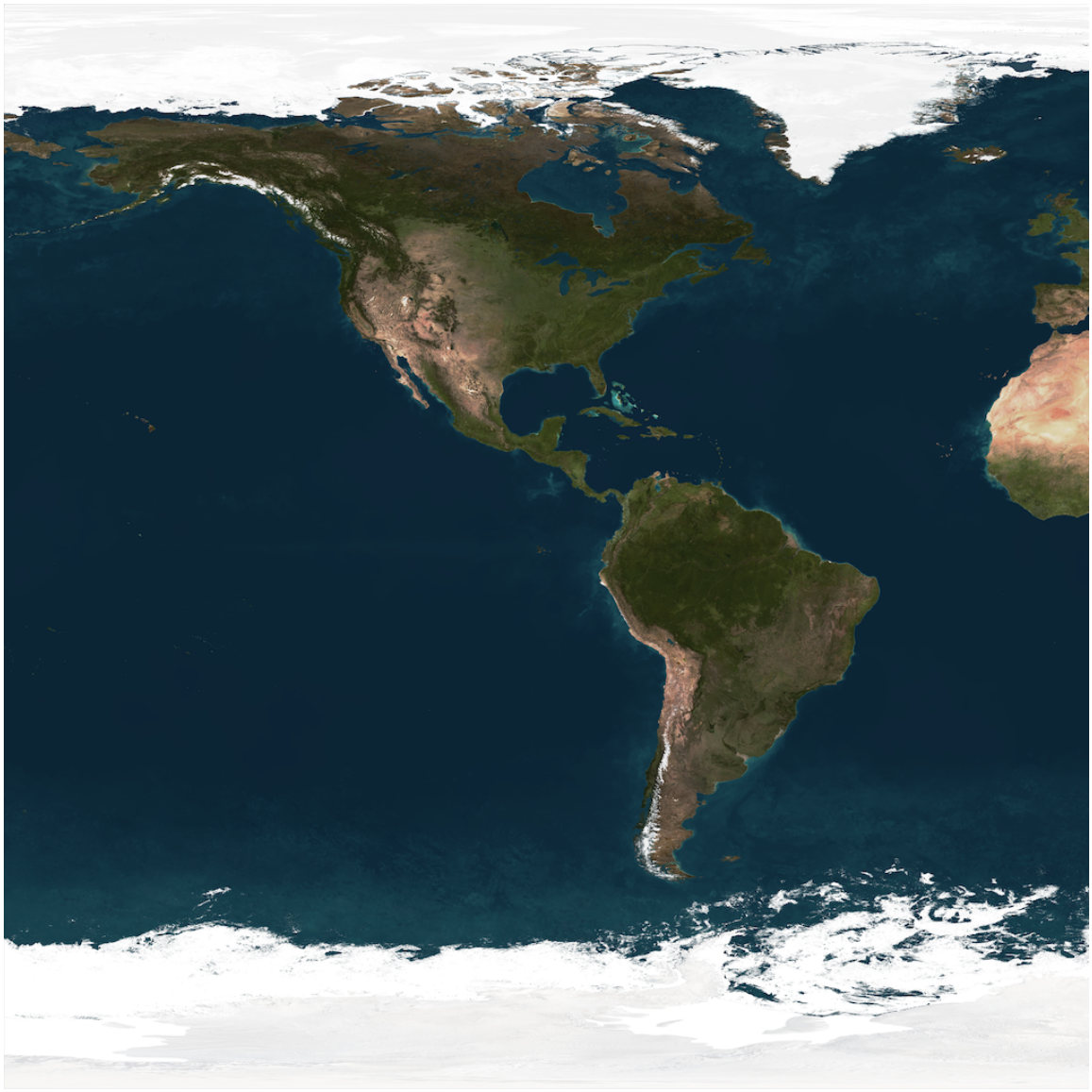
MAX_ZOOM_OUT and MAX_ZOOM_IN which as the names suggest control how far in and out you can zoom. You technically could zoom all the way down to the ground but the image gets really fuzzy the closer you get.

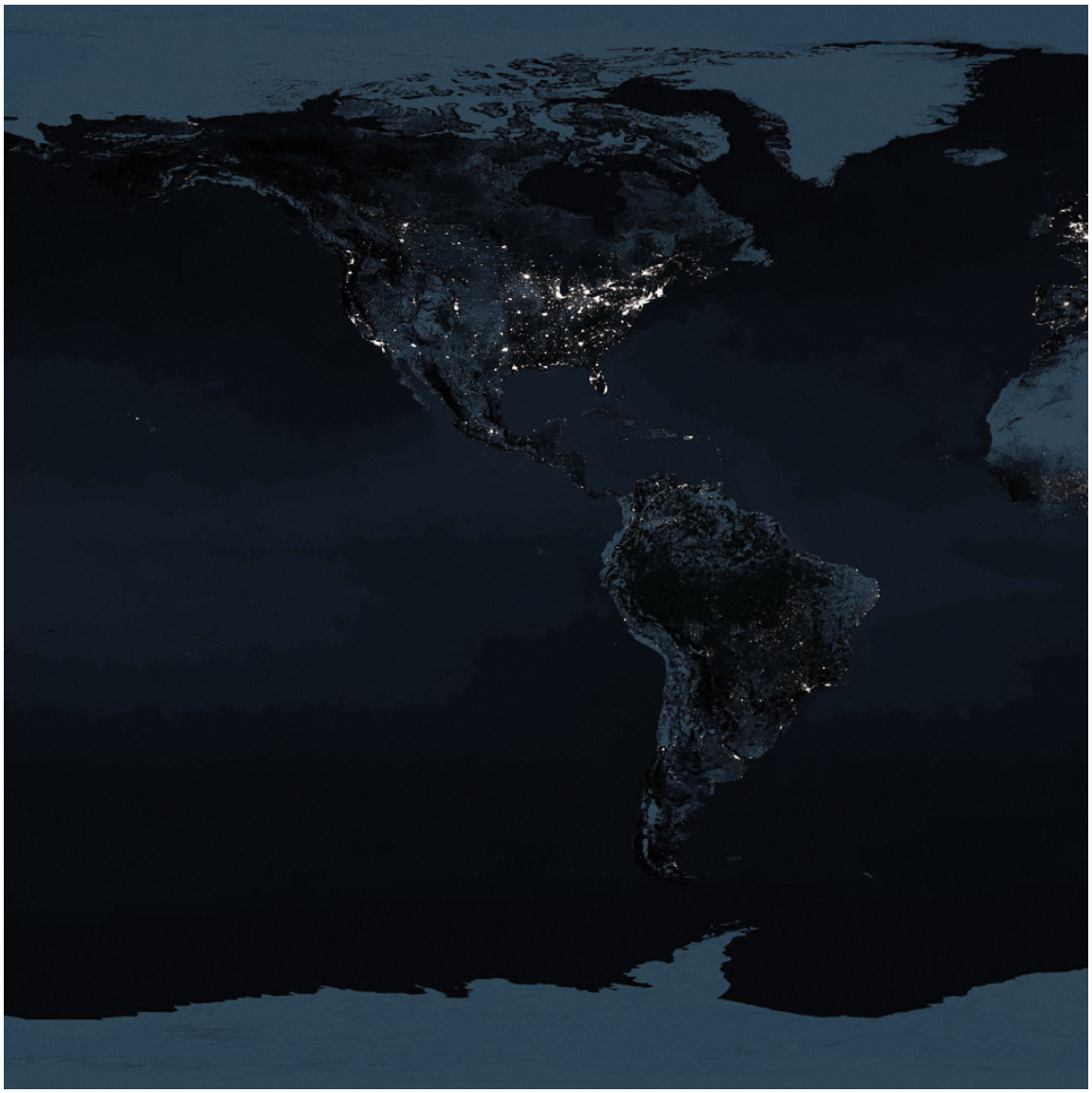
ZOOM_TOUCH_MULTIPLIER which affect the speed at which you zoom in and out as you pinch gestures.

SmoothRotator

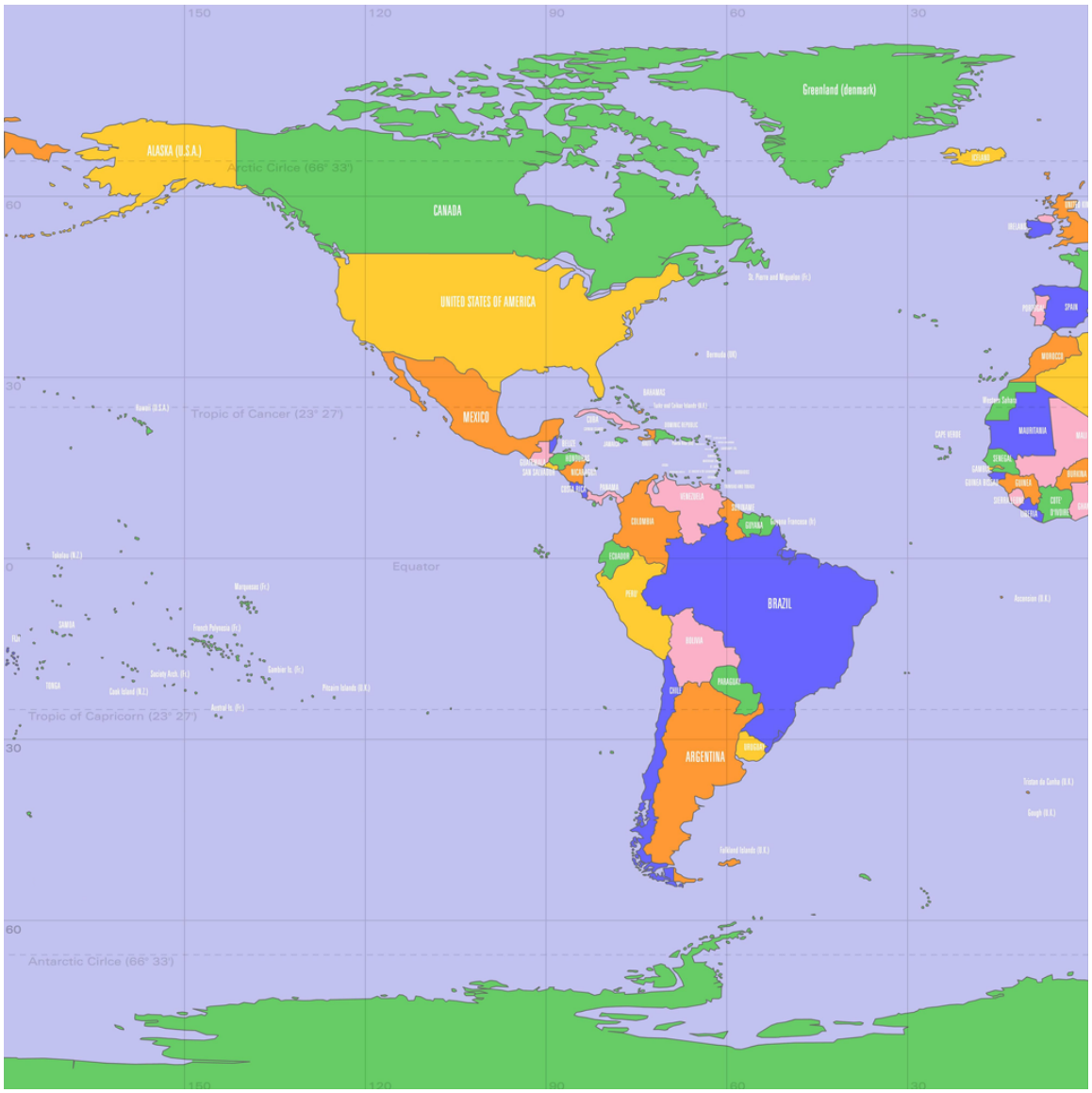
This is a convenience class encapsulating the mathematically formulas needed to programmatically create a smooth rotation between two points in 3D space.

Examples of some of the textures we use:






Greenland
(Den.)
US
Iceland
Canada
Ireland
United States
Portugal
Morocco
Western Sahara
Mexico
Cuba
Dominican republic
Jamaica
Haiti
Belize
Guatemala
Honduras
El Salvador
Nicaragua
Trinidad
and Tobago
Costa Rica
Venezuela
Guyana
Suriname
French Guiana
Panama
Colombia
Ecuador
Peru
Brazil
Bolivia
Paraguay
Uruguay
Argentina
Chile
Mauritania
Senegal
Guinea-Bissau
Guinea
Sierra Leone
Liberia
Cote
Divore


Greenland (denmark)
ALASKA (U.S.A.)
Arctic Circle (66° 33')
CANADA
UNITED STATES OF AMERICA
MEXICO
Tropic of Cancer (23° 27')
Equator
BRAZIL
ARGENTINA
Tropic of Capricorn (23° 27')
Antarctic Circle (66° 33')
SPAIN
MOROCCO
MAURITANIA
SENEGAL
COLOMBIA
ECUADOR
PERU
BOLIVIA
PARAGUAY
CHILE
GUYANA
SAMOA
TONGA
150
120
90
60
30