Creating a Continuously Updated Animated KML Loop with a PHP Mashup

Bruce D. Muller
University of Nebraska at Omaha
bmuller@unomaha.edu

This is the first of three Practical Cartographer's Corner pieces in this issue which come from a graduate course entitled "Cartographic Methods," taught by Michael Peterson (mpeterson@unomaha.edu) at the University of Nebraska at Omaha during the 2013 Spring semester.

INTRODUCTION

Google Earth (GE) can display timestamped features sequentially to create an animation on its interactive globe. The GE TimeSpan capability has been utilized to show a variety of imagery such as historical maps, or before/during/after imagery of remotely sensed landscapes for easy comparison of changes due to a particular event such as flooding. The period between such events can be minutes, days, or even years, and possibly be available at irregular user-specified intervals. As such, TimeSpan is well-suited to properly display imagery that can be georeferenced, encoded into a Keyhole Markup Language (KML) file, and timestamped for sequential viewing. For imagery that is available at regular intervals, an automated process can be employed to create a continuously updated KML file. Here I describe such a process, a mashup that generates a loop of national weather radar imagery for viewing on Google Earth.

PROCEDURAL OVERVIEW

A mashup is typically a combination of a several different processes, such as a script that generates output that is then compatible for viewing in another application. In this case, the server-side scripting language known as PHP is used to generate a KML file that can be opened and viewed in GE. In order to provide a continuously updated product, a Cron function (a time-based scheduler) is set up on a recurring basis and run as often as needed, usually when a new "most recent" image becomes available that can be added to the animated loop.

The National Weather Service (NWS) offers weather imagery, for example from satellites (clouds) or radar (precipitation), as near-real time static images or as animated loops, the latter showing perhaps the most recent 1–2 hours of recorded imagery. While provided via a web page, users can also download the latest imagery via FTP from the NWS or other sources. Websites showing animated radar or satellite loops usually display the imagery by utilizing a Java applet or similar browser/client-side program. In 2009, the NWS began to provide a limited selection of their products in a KML format in an experimental fashion. However, the vast majority of weather imagery, including loops, is still supplied through traditional web pages. Due to the limited availability of current animated weather products in the KML format that can then take advantage of GE's functionality including zooming, panning, etc., I implemented a mashup that published a continually updating KML file. When opened in GE, it displayed a weather radar loop comprised of approximately 150 NEXRAD images over the contiguous United States (CONUS). Additionally, it is even possible to overlay other layers in GE such as a satellite (cloud) layer where those graphics may come in at different intervals (e.g., every 30 minutes). Each additional layer added will need its own KML file, perhaps even created by its own separate PHP mashup on a different schedule.

THE MASHUP

The mashup procedure is controlled with the Cron function and programs that run a PHP script at user-specified intervals. In this case, due to the availability of a new CONUS radar composite every 10 minutes, the Cron function called the PHP script every 10 minutes around the clock (at every :00, :10, :20, etc. past the hour). An example Cron command line to run the radar.php script is as follows:

```
*/10**** php -f /home/bdmuller/
 public_html/cgi-bin/radar.php
```

where `*/10****` specifies running the timing interval of every 10 minutes, and `php -f` directs the running of a PHP script, in this case `radar.php` located in the specified folder. However, various web hosting services may include a graphical user interface to enter the timing interval for Cron jobs, as well as a *browse* option to navigate the user to the correct folder for the PHP script, so knowledge of the intricacies of the UNIX Cron command may not be necessary.

Once launched, the PHP script accomplishes the following steps, in this order:

1. Define time variables
2. Construct TimeSpan variables
3. Write to KML file
4. Rename each previously downloaded image to the next older version, and retrieve the most recent new radar image
5. (Optional) FTP (or move/copy) output files to another location, if desired

STEP 1: DEFINING THE TIME VARIABLES

The PHP code starts out by defining the time variables necessary to accomplish all the tasks (Example 1). Time variables are defined for each time step in the TimeSpan loop sequence: one step for every 10 minutes (600 seconds) in this case. Variables for year, month, day, hour and minute need to be defined at each time step to build the TimeSpan variable text string in step 2 (below). Thus, each additional set of variables uses the time 10 minutes prior from the previous time step, up to 4 hours prior to the present time, in order to define time variables. This results in 25 total time steps (including the beginning and ending times of the loop), but only 24 frames are shown in the animation as the first frame of the animation will be valid for the 10 minute window of time between 4 hours and 3 hours 50 minutes prior to the end of the loop.

```php
<?php
// compute time variables for previous imagery
// at 10 minute intervals

date_default_timezone_set('UTC');
$b = time () - 600;
$c = time () - 1200;
$d = time () - 1800;
$e = time () - 2400;
$f = time () - 3000;
$g = time () - 3600; // one hour ago
$h = time () - 4200;
$i = time () - 4800;
$j = time () - 5400;
$k = time () - 6000;
$l = time () - 6600;
$m = time () - 7200; // two hours ago
$n = time () - 7800;
$o = time () - 8400;
$p = time () - 9000;
$q = time () - 9600;
$r = time () - 10200;
$s = time () - 10800; // three hours ago
$t = time () - 11400;
$u = time () - 12000;
$v = time () - 12600;
$w = time () - 13200;
$x = time () - 13800;
$y = time () - 14400; // four hours ago

$curYear1 = date('Y'); // now
$curMon1 = date('m');
$curDay1 = date('d');
$curHour1 = date('H');
$curMin1 = date('i');

$curYear2 = date('Y',$b);
$curMon2 = date('m',$b);
$curDay2 = date('d',$b);
$curHour2 = date('H',$b);
$curMin2 = date('i',$b); // this last block of
// code is repeated to define all time variables
```

Example 1: Sample of code that defines all the time variables.

STEP 2: ASSEMBLING THE TIMESPAN TEXT STRINGS

Next, the PHP code (Example 2) assembles the TimeSpan variables in the format that Google Earth requires, in order to read them properly. The TimeSpan variable is formatted as follows:

YYYY-MM-DDTHH:MM:SSZZZZZZ

Where:

- "YYYY" represents the 4-digit year (e.g. 2013)
- "MM" represents the 2-digit month (e.g. 01 = January, … 12 = December)
- "DD" represents the 2-digit day (e.g. 01, 02, … up to 31)
- "T" is a text divider between the date and time segments of the TimeSpan variable
- "HH" represents the 2-digit hour (e.g. 00, 01, … up to 23)
- "MM" represents the 2-digit minute (e.g. 00, 01, … up to 59)
- "SS" represents the 2-digit second (e.g. 00, 01, up to 59)
- "ZZZZZZ" represents the difference from UTC in ± HH:MM (e.g. +04:00 if 4 hours ahead of UTC. Note: this is optional; it can be left blank if conversion to/from UTC and local time is not necessary or they are the same. If left blank GE will assume it is the local time.)

STEP 3: WRITING TO THE KML FILE

The heart of the code is the generation of the KML output file, and this is accomplished through a series of `fwrite` commands that write to the file. Notice that in the first line of this section of code (Example 3), `unlink` is used as the PHP command to remove a file. After the file is recreated, `fwrite` commands specify the text to enter line-by-line to build the KML file using proper KML formatting

```
$T = "T";//create text strings for timespan variables
$H = "-";
$C = ":";
$Z = "00";
$tv1 = "{$curYear1}{$H}{$curMon1}{$H}{$curDay1}{$T}{$curHour1}{$C}{$curMin1}{$C}{$Z}";// now
$tv2 = "{$curYear2}{$H}{$curMon2}{$H}{$curDay2}{$T}{$curHour2}{$C}{$curMin2}{$C}{$Z}";// 10 minutes ago
$tv3 = "{$curYear3}{$H}{$curMon3}{$H}{$curDay3}{$T}{$curHour3}{$C}{$curMin3}{$C}{$Z}";// 20 minutes ago
$tv4 = "{$curYear4}{$H}{$curMon4}{$H}{$curDay4}{$T}{$curHour4}{$C}{$curMin4}{$C}{$Z}";// 30 minutes ago
$tv5 = "{$curYear5}{$H}{$curMon5}{$H}{$curDay5}{$T}{$curHour5}{$C}{$curMin5}{$C}{$Z}";// etc.
$tv6 = "{$curYear6}{$H}{$curMon6}{$H}{$curDay6}{$T}{$curHour6}{$C}{$curMin6}{$C}{$Z}";// etc.
$tv7 = "{$curYear7}{$H}{$curMon7}{$H}{$curDay7}{$T}{$curHour7}{$C}{$curMin7}{$C}{$Z}";
$tv8 = "{$curYear8}{$H}{$curMon8}{$H}{$curDay8}{$T}{$curHour8}{$C}{$curMin8}{$C}{$Z}";
$tv9 = "{$curYear9}{$H}{$curMon9}{$H}{$curDay9}{$T}{$curHour9}{$C}{$curMin9}{$C}{$Z}";
$tv10 = "{$curYear10}{$H}{$curMon10}{$H}{$curDay10}{$T}{$curHour10}{$C}{$curMin10}{$C}{$Z}";
$tv11 = "{$curYear11}{$H}{$curMon11}{$H}{$curDay11}{$T}{$curHour11}{$C}{$curMin11}{$C}{$Z}";
$tv12 = "{$curYear12}{$H}{$curMon12}{$H}{$curDay12}{$T}{$curHour12}{$C}{$curMin12}{$C}{$Z}";
$tv13 = "{$curYear13}{$H}{$curMon13}{$H}{$curDay13}{$T}{$curHour13}{$C}{$curMin13}{$C}{$Z}";
$tv14 = "{$curYear14}{$H}{$curMon14}{$H}{$curDay14}{$T}{$curHour14}{$C}{$curMin14}{$C}{$Z}";
$tv15 = "{$curYear15}{$H}{$curMon15}{$H}{$curDay15}{$T}{$curHour15}{$C}{$curMin15}{$C}{$Z}";
$tv16 = "{$curYear16}{$H}{$curMon16}{$H}{$curDay16}{$T}{$curHour16}{$C}{$curMin16}{$C}{$Z}";
$tv17 = "{$curYear17}{$H}{$curMon17}{$H}{$curDay17}{$T}{$curHour17}{$C}{$curMin17}{$C}{$Z}";
$tv18 = "{$curYear18}{$H}{$curMon18}{$H}{$curDay18}{$T}{$curHour18}{$C}{$curMin18}{$C}{$Z}";
$tv19 = "{$curYear19}{$H}{$curMon19}{$H}{$curDay19}{$T}{$curHour19}{$C}{$curMin19}{$C}{$Z}";
$tv20 = "{$curYear20}{$H}{$curMon20}{$H}{$curDay20}{$T}{$curHour20}{$C}{$curMin20}{$C}{$Z}";
$tv21 = "{$curYear21}{$H}{$curMon21}{$H}{$curDay21}{$T}{$curHour21}{$C}{$curMin21}{$C}{$Z}";
$tv22 = "{$curYear22}{$H}{$curMon22}{$H}{$curDay22}{$T}{$curHour22}{$C}{$curMin22}{$C}{$Z}";
$tv23 = "{$curYear23}{$H}{$curMon23}{$H}{$curDay23}{$T}{$curHour23}{$C}{$curMin23}{$C}{$Z}";
$tv24 = "{$curYear24}{$H}{$curMon24}{$H}{$curDay24}{$T}{$curHour24}{$C}{$curMin24}{$C}{$Z}";
$tv25 = "{$curYear25}{$H}{$curMon25}{$H}{$curDay25}{$T}{$curHour25}{$C}{$curMin25}{$C}{$Z}";// 4 hours ago
```

Example 2: Sample of code that assembles the TimeSpan text strings.

```
unlink('radar.kml');
$filename = 'radar.kml';
$handle = fopen($filename, "w");

fwrite($handle,  '<?xml version="1.0" encoding="UTF-8"?>'."\n");
fwrite($handle,  '<kml xmlns="http://earth.google.com/kml/2.1">'."\n");
fwrite($handle,  '<Folder>'."\n");
fwrite($handle,  '  <name>TEST: GE animation of CONUS Radar Mosaic</name>'."\n");
fwrite($handle,  '  <description><![CDATA['."\n");
fwrite($handle,  'Before animating, shrink time window in slider to minimum.'."\n");
fwrite($handle,  'Animation only works in latest GE beta 4.0 (after Sept 13, or 14 for Mac).'."\n");
fwrite($handle,  'Overlay shows NWS CONUS Radar Mosaic '."\n");
fwrite($handle,  'NOTE: This is experimental.'."\n");
fwrite($handle,  ']]></description>'."\n");
fwrite($handle,  '  <LookAt>'."\n");
fwrite($handle,  '    <longitude>-100.00</longitude>'."\n");
fwrite($handle,  '    <latitude>50.406626367301044</latitude>'."\n");
fwrite($handle,  '    <range>10000000</range>'."\n");
fwrite($handle,  '    <tilt>0</tilt>'."\n");
fwrite($handle,  '    <heading>0</heading>'."\n");
fwrite($handle,  '    <TimeSpan>'."\n");
fwrite($handle,  '    <begin>'."$tv25".'</begin>'."\n");
fwrite($handle,  '    <end>'."$tv24".'</end>'."\n");
fwrite($handle,  '    </TimeSpan>'."\n");
fwrite($handle,  '  </LookAt>'."\n");
fwrite($handle,  '  <TimeSpan>'."\n");
fwrite($handle,  '    <begin>'."$tv25".'</begin>'."\n");
fwrite($handle,  '    <end>'."$tv1".'</end>'."\n");
fwrite($handle,  '  </TimeSpan>'."\n");
fwrite($handle,  '  <Folder>'."\n");
fwrite($handle,  '    <name>frames</name>'."\n");
fwrite($handle,  '    <description>Animation frames NWS CONUS Radar</description>'."\n");
```

Example 3: Sample of code that creates and opens the new KML file and writes the header information.

and language. Text inside single quote marks (`'`) is written to the KML file as-is, while variables like the TimeSpan values (`$tv1`, `$tv2`, etc.) are translated to their text string equivalent values when inserted between double quote marks (`"`). Each `fwrite` line ends with a carriage return command (`"\n"`) so the output KML file is human-readable in a text editor or word processor.

After writing the opening section of the KML file, the PHP code becomes very repetitive (Example 4), repeating the segment of code between the lines `<groundoverlay>` and `</groundoverlay>` 24 times, once for each step of the animation. These repeating sections are identical except for the different TimeSpan variables in each section, as well as different hyperlink references to the correct graphical file for that time step. This section of code then closes the KML file with an `fclose` command once the final line of KML is written.

STEP 4: FILE RENAMING SEQUENCE

The last required step of the mashup is to rename each image of the loop to the next older version of the file, starting with the oldest image first (Example 5). This has the effect of keeping only the last 24 images (one every 10 minutes for 4 hours). This was done so as not to keep an endlessly large archive of all downloaded images. Once the renaming sequence is completed, the newest radar composite image is retrieved from the NWS website using the `file_put_contents` and `file_get_contents` commands shown in the Example 5 code.

STEP 5 (OPTIONAL): FTP OUTPUT FILES TO ALTERNATE LOCATION

Although not shown here, one optional step, given PHP's capabilities, is to use FTP or similar PHP file move/copy commands to transfer the updated files to a new server

```
fwrite($handle,  '    <GroundOverlay>'."\n");
fwrite($handle,  '      <name>CONUS - '."$tv24".'</name>'."\n");
fwrite($handle,  '      <description>CONUS Mosaic</description>'."\n");
fwrite($handle,  '      <TimeSpan>'."\n");
fwrite($handle,  '        <begin>'."$tv25".'</begin>'."\n");
fwrite($handle,  '        <end>'."$tv24".'</end>'."\n");
fwrite($handle,  '      </TimeSpan>'."\n");
fwrite($handle,  '      <Icon>'."\n");
fwrite($handle,  '     <href>http://bdmuller.freetzi.com/Presentation/latest_radaronly-23.gif</href>'."\n");
fwrite($handle,  '      </Icon>'."\n");
fwrite($handle,  '      <LatLonBox id="BBOX">'."\n");
fwrite($handle,  '     <south>21.652538062803</south>'."\n");
fwrite($handle,  '     <north>50.406626367301044</north>'."\n");
fwrite($handle,  '     <west>-127.620375523875420</west>'."\n");
fwrite($handle,  '     <east>-66.517937876818</east>'."\n");
fwrite($handle,  '     <rotation>0</rotation>'."\n");
fwrite($handle,  '      </LatLonBox>'."\n");
fwrite($handle,  '    </GroundOverlay>    <GroundOverlay>'."\n");
// the section of code above is repeated for each frame of timespan loop and variable $tvxx is changed for each
// the section of code below is the final section to the end of the file
fwrite($handle,  '    </GroundOverlay>    <GroundOverlay>'."\n");
fwrite($handle,  '      <name>CONUS - '."$tv1".'</name>'."\n");
fwrite($handle,  '      <description>northeast Mosaic</description>'."\n");
fwrite($handle,  '      <TimeSpan>'."\n");
fwrite($handle,  '     <begin>'."$tv2".'</begin>'."\n");
fwrite($handle,  '     <end>'."$tv1".'</end>'."\n");
fwrite($handle,  '      </TimeSpan>'."\n");
fwrite($handle,  '      <Icon>'."\n");
fwrite($handle,  '     <href>http://bdmuller.freetzi.com/Presentation/latest_radaronly.gif</href>'."\n");
fwrite($handle,  '      </Icon>'."\n");
fwrite($handle,  '      <LatLonBox id="BBOX">'."\n");
fwrite($handle,  '     <south>21.652538062803</south>'."\n");
fwrite($handle,  '     <north>50.406626367301044</north>'."\n");
fwrite($handle,  '     <west>-127.620375523875420</west>'."\n");
fwrite($handle,  '     <east>-66.517937876818</east>'."\n");
fwrite($handle,  '     <rotation>0</rotation>'."\n");
fwrite($handle,  '      </LatLonBox>'."\n");
fwrite($handle,  '    </GroundOverlay></Folder>'."\n");
fwrite($handle,  '</Folder>'."\n");
fwrite($handle,  '</kml>'."\n");
fclose($handle);
```

Example 4: Sample of code that writes each groundoverlay *segment of the KML file, repeated 24 times (once for each time step).*

or location such as your own public web site folder, if not already there. Commands like `ftp_open`, `ftp_put` and `ftp_close` are used to open an FTP connection, transfer the selected files and then close the FTP connection when completed.

RESULTING ANIMATION

Once the PHP code execution has completed, the new KML file is ready to open in GE. Upon opening the KML file, GE will recognize the TimeSpan commands and individual frames from each groundoverlay section. While GE loads each of the 24 frames, it works best to manually advance the time steps on the TimeSpan slider bar the first time through, allowing a few moments for each frame to load. Once they have loaded, you can open the *Date* *and Time Options* (click on the wrench icon on the TimeSpan control bar), then adjust the loop animation speed and check the *Loop Animation* box if desired. Once started, the animation will run in a continuous loop, and the user can zoom in, pan and rotate around any feature of interest, utilizing the full capabilities of GE. As an example, several images from the animated loop of April 22, 2013 can be seen in Figure 1. This particular KML file is available for download at: **http://bdmuller.comeze.com/Project/radar.kml**.

```
//Previous files are renamed to the next older version each
//time the code executes until they are each overwritten keeping
//only the most recent 4 hours of the loop
rename("latest_radaronly-23.gif", "latest_radaronly-24.gif");
rename("latest_radaronly-22.gif", "latest_radaronly-23.gif");
rename("latest_radaronly-21.gif", "latest_radaronly-22.gif");
rename("latest_radaronly-20.gif", "latest_radaronly-21.gif");
rename("latest_radaronly-19.gif", "latest_radaronly-20.gif");
rename("latest_radaronly-18.gif", "latest_radaronly-19.gif");
rename("latest_radaronly-17.gif", "latest_radaronly-18.gif");
rename("latest_radaronly-16.gif", "latest_radaronly-17.gif");
rename("latest_radaronly-15.gif", "latest_radaronly-16.gif");
rename("latest_radaronly-14.gif", "latest_radaronly-15.gif");
rename("latest_radaronly-13.gif", "latest_radaronly-14.gif");
rename("latest_radaronly-12.gif", "latest_radaronly-13.gif");
rename("latest_radaronly-11.gif", "latest_radaronly-12.gif");
rename("latest_radaronly-10.gif", "latest_radaronly-11.gif");
rename("latest_radaronly-9.gif", "latest_radaronly-10.gif");
rename("latest_radaronly-8.gif", "latest_radaronly-9.gif");
rename("latest_radaronly-7.gif", "latest_radaronly-8.gif");
rename("latest_radaronly-6.gif", "latest_radaronly-7.gif");
rename("latest_radaronly-5.gif", "latest_radaronly-6.gif");
rename("latest_radaronly-4.gif", "latest_radaronly-5.gif");
rename("latest_radaronly-3.gif", "latest_radaronly-4.gif");
rename("latest_radaronly-2.gif", "latest_radaronly-3.gif");
rename("latest_radaronly-1.gif", "latest_radaronly-2.gif");
rename("latest_radaronly.gif", "latest_radaronly-1.gif");

//After renaming old files then retrieve newest CONUS Radar Mosaic
file_put_contents("latest_radaronly.gif",
  file_get_contents("http://radar.weather.gov/ridge/Conus/
       RadarImg/latest_radaronly.gif"));
```

Example 5: Sample of code that renames each previously downloaded image to the next older version in the sequence ("23" becomes "24," "22" becomes "23," etc.)

SUMMARY

Through the use of a mashup method, a computer user familiar with Cron functionality, PHP commands, KML, and GE can create their own animated loops using different types of graphics that update on regular or even irregular intervals and display these within Google Earth. Just about any time-stamped graphical image that is produced in a recurring fashion can be assembled into a loop and viewed as a GE TimeSpan animation. Finally, the mashup can be customized to operate on any number of animation frames or timing intervals that the user desires, creating a custom animation suitable for viewing through Google Earth.

REFERENCES

Google. 2013. "Keyhole Markup Language, Time and Animation." Last modified November 14. **https://developers.google.com/kml/documentation/time**.

IBM. 2013. "UNIX Cron Format." *IBM DB2 Database for Linux, UNIX, and Windows Information Center.* Accessed April 15. **http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=%2Fcom.ibm.db2.luw.sql.rtn.doc%2Fdoc%2Fc0054381.html**.

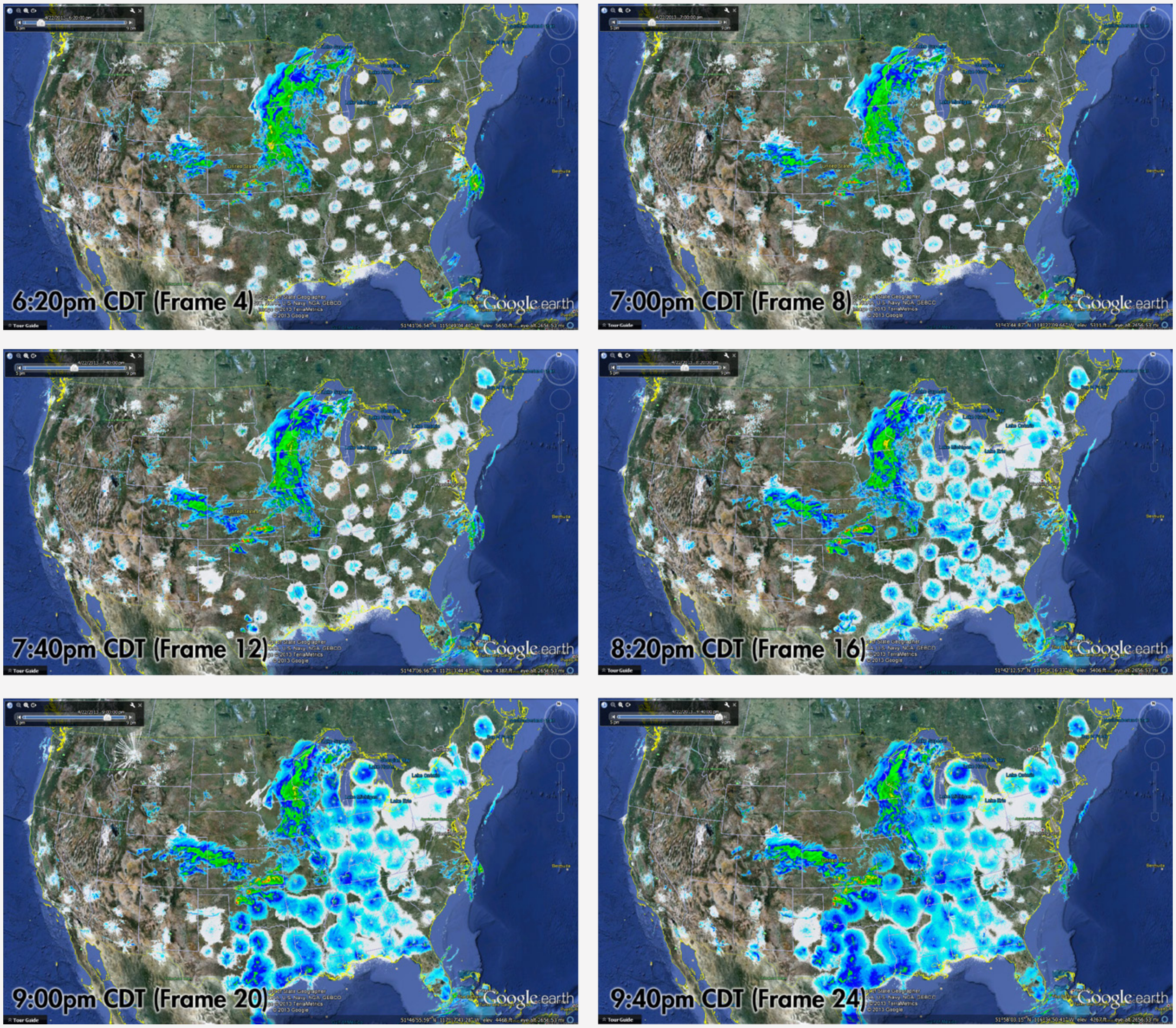

Figure 1: Selected frames from the animated loop of April 22, 2013.

National Weather Service. 2013. "National Weather Data in KML/KMZ Formats." Last modified August 14. http://www.srh.noaa.gov/gis/kml/.

The PHP Documentation Group. 2014. "PHP Manual." Last modified March 18. http://www.php.net/manual/en/index.php.