Sample SML Scripts

Movie Generation Scripts

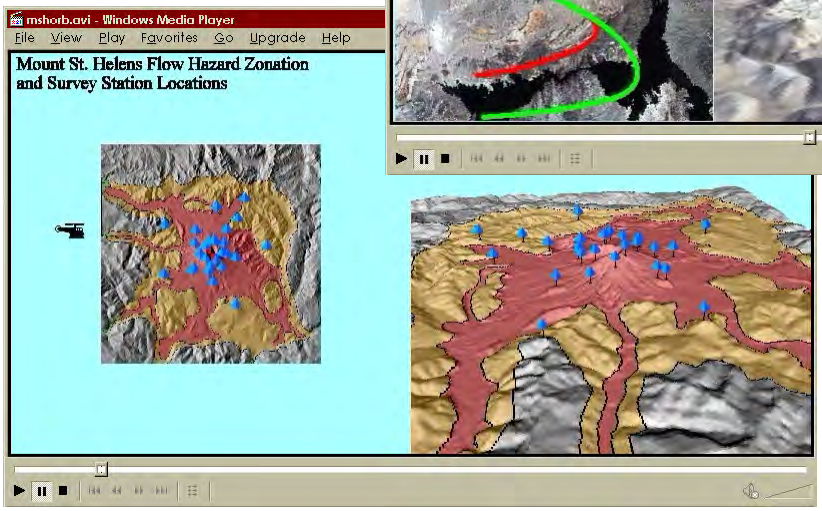
The Spatial Manipulation Language (SML) now puts you in the director's chair. Using new movie generation SML functions in TNTmips 6.50 you can create scripts that set up and record custom animations of your geospatial data. You can record these animations in either MPEG format (any computer platform) or AVI format (Windows computers only) and set both frame rate and recording time. MicroImages has prepared a number of sample movie generation scripts, one of which is excerpted on the reverse side of this page. Although these movie generation scripts were prepared after the TNTmips 6.50 Products CD was mastered, you can download any of the scripts as well as sample movie files from the Downloads page of the MicroImages web site:

www.microimages.com/freestuf/.

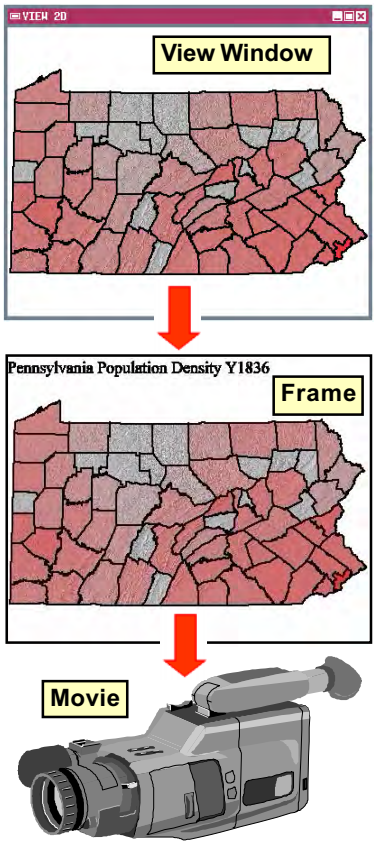
The 30 new SML functions and class methods that implement movie generation incorporate many of the capabilities of the 3D Simulation process in TNTmips. But they also give you more control over the 3D viewing parameters. You can specify viewer position and view direction independently, so that the 3D view can look ahead along the flight path (as in the standard 3D Simulation process), or to the side, straight down, or backward along the flight path. You can create a single 3D movie that incorporates elements of the path, orbit, and pan modes of the 3D Simulation process.

But animations are not limited to 3D simulations. An animation merely consists of a gradually varying series of static frames. Each frame is rendered from one or more View windows created by the script and then copied into the output MPEG or AVI file. The movie therefore can record any sequential change in the view windows used to create the frames.

A simple example would be a sequence of 2D views showing a change in some mapped parameter through time. The script can sequentially add and then remove a series of pre-prepared layers to and from the view, or modify the display parameters for a single continuing layer (such as a set of vector polygons with attached attributes) to create the change from frame to frame. More complex examples might sequentially display the result of some process computed in the script, such as a series of viewsheds computed for different positions along a traverse line through a terrain model. The possibilities are limited only by your source data and your imagination!



In a 3D simulation script you can render both 2D and 3D views into each frame, as shown in the above illustration. After each frame is captured you can also draw symbols into it marking the current viewer position and view center position, and draw a trail of previous position symbols behind the moving symbol. Sample 3D simulation scripts are available to show how to script these features, and to set up panning, a spiral orbit, and contant-altitude and constant-height flight paths.



The main processing loop of any movie script must perform three major functions:

1. Change the content of one or more view windows and redraw. This step might involve altering the viewing parameters for a 3D view, or adding new layers or updating existing layers in a 2D view.
2. Copy the contents of the view window(s) to a frame. Additional graphic symbols or annotation text can be drawn directly into the frame if you wish.
3. Copy the frame to the output movie file.

New functions in the Frame and Movie function groups are used to set up the generic frame and movie parameters, capture the View window contents to a frame, and copy the frame to the output AVI or MPEG file. For 3D animations, new class methods in the VIEWPOINT3D class are used to manipulate the settings for the 3D view. You can set viewer and view center position coordinates explicitly for each frame, or move either position a specified distance or direction relative to the previous position. Either position can be rotated around the other. You can also set either position and then use an azimuth angle, elevation angle, and distance to define the other.

Sample scripts have been prepared to illustrate how you might use the new features of TNTmips' Spatial Manipulation Language (SML). Key sections of one script are reproduced below for your quick perusal. The entire script can be downloaded from the SML script exchange at www.microimages.com/sml/ftpsmllink/TNT_Products_V6.5_CD.

Excerpts from Constant Height Flight Path Script (PATHcHT2.sml)

```
string format$;
format$ = "AVI";

string framerate$;
framerate$ = "MOVIE_FRAMERATE_24";

numeric time;
time = 60;

GetInputRaster(Surface);
GetInputRaster(RastDrape);
GetInputVector(FlightPathVec);
GetInputVector(ViewCenterVec);

string styleFilename$;
string styleObjectname$;
GetInputObject("Style","Select style object for center and viewer
        point symbols:", styleFilename$, styleObjectname$)
string viewer$;
viewer$ = "VIEWER";
string center$;
center$ = "CENTER"

print("START");

numeric size;
size = 320;

numeric zoomfactor;
zoomfactor = 1.0;

class GROUP group;
group = GroupCreate();

string flags$;
flags$ = "NoScalePosLine,NoIconBar,NoScrollbars,NoStatusLine";

class XmForm dialog2d;
class VIEW view2d;
dialog2d = CreateFormDialog("VIEW 2D");
view2d = GroupCreateView(group,dialog2d,"",size,size,flags$);
view2d.BackgroundColor.red = 67;
view2d.BackgroundColor.green = 100;
view2d.BackgroundColor.blue = 100;

class XmForm dialog3d;
class VIEW3D view3d;
dialog3d = CreateFormDialog("VIEW 3D");
view3d = GroupCreate3DView(group,dialog3d,"",size,size,flags$);
view3d.BackgroundColor.red = 67;
view3d.BackgroundColor.green = 100;
view3d.BackgroundColor.blue = 100;

GroupQuickAddRasterVar(group,Surface,1);
GroupQuickAddRasterVar(group,RastDrape,0);

DialogOpen(dialog2d);
DialogOpen(dialog3d);

ViewRedrawFull(view2d);
ViewRedrawFull(view3d);
ViewZoomOut(view2d,zoomfactor,1);

x2d = 0;
y2d = 0;
x3d = size;
y3d = 0;
w = 2 * size;
h = size;

numeric fontsize;
fontsize = 16;

class Frame frame;
frame = FrameCreate(w,h);

ActivateGC(FrameCreateGC(frame));
DrawTextSetHeightPixels(fontsize);
DrawUseStyleObject(styleFilename$,styleObjectname$);

class Movie movie;
movie = MovieInit();

MovieSetFormat(movie,format$);
MovieSetFrameRate(movie,framerate$);
MovieSetFrameWidth(movie,w);
MovieSetFrameHeight(movie,h);
```

Set movie format, frame rate, and recording time

Select input DEM for surface, raster drape, and two vector objects containing ground traces of flight path and view center path

Select style object containing point symbol styles for viewer position and view center position

Variables to set size of 2D and 3D view windows and zoom-out factor for 2D view

Create display group. Create flag to create view without iconbar, scrollbars, status line, and scale/position line; important to maintain fixed window size during movie generation

Create dialog and 2D view

Create dialog and 3D view

Add surface and raster drape to group

Open both views

Full redraw of both views

Parameters to set location and size of each view in movie frame

Set fontsize for text annotation in frame

Create blank frame

Create graphics context for frame

Initialize movie

Set more movie parameters

```
string ext$;
ext$ = MovieGetFileExt(movie);
string filename$;
filename$ = GetOutputFileName("","Make filename for movie",ext$);

if (time <= 1.0) time = 1.0;

numFrames = time * rate;

class Georef georefS;
georefS = GetLastUsedGeorefObject(Surface);
GeorefSetProjection(georefS,group.Projection);

class Georef georefFlight;
georefFlight = GetLastUsedGeorefObject(FlightPathVec)
GeorefSetProjection(georefFlight,group.Projection);

class Georef georefCent;
georefCent = GetLastUsedGeorefObject(ViewCenterVec);
GeorefSetProjection(georefCent,group.Projection);

MovieStart(movie,filename$);

for i = 1 to numFrames {
    class POINT3D fpt;
    fpt.x = xarrayf_eq[i];
    fpt.y = yarrayf_eq[i];
    fpt.z = zarrayf_eq[i];
    vp.SetViewerPosition(fpt);

    class POINT3D cpt;
    cpt.x = xarrayc_eq[i];
    cpt.y = yarrayc_eq[i];
    cpt.z = zarrayc_eq[i];
    vp.SetCenter(cpt);

    ViewRedraw(view3d);

    FrameCopyFromView(frame,view2d,0,0,size,size,x2d,y2d);
    FrameCopyFromView(frame,view3d,0,0,size,size,x3d,y3d);

    for j = 1 to (i - 1) {
       SetColor(colorc)
       FillCircle(xarraycs[j],yarraycs[j],2)
    }

    class POINT2D point;
    point.x = vp.CenterPoint.x;
    point.y = vp.CenterPoint.y;
    point =
TransPoint2D(point,ViewGetTransMapToView(view2d,group.Projection));
    point = TransPoint2D(point,ViewGetTransViewToScreen(view2d));
    DrawSetPointStyle(center$);
    DrawPoint(point.x,point.y);

    xarraycs[i] = point.x;
    yarraycs[i] = point.y;

    for j = 1 to (i - 1) {
       SetColor(colorf)
       FillCircle(xarrayfs[j],yarrayfs[j],2)
    }

    point.x = vp.ViewPos.x;
    point.y = vp.ViewPos.y;
    point =
TransPoint2D(point,ViewGetTransMapToView(view2d,group.Projection));
    point = TransPoint2D(point,ViewGetTransViewToScreen(view2d));
    DrawSetPointStyle(viewer$);
    DrawPoint(point.x,point.y);

    xarrayfs[i] = point.x;
    yarrayfs[i] = point.y;

    DrawTextSetColors(black);
    DrawTextSimple("Muddy Mountains, NV",2,fontsize);
    DrawTextSetColors(colorc);
    DrawTextSimple("View center path",2,fontsize*2.1);
    DrawTextSetColors(colorf);
    DrawTextSimple("Flight path",2,fontsize*3.3);

    MovieAddFrame(movie,frame);
}

MovieStop(movie);
MovieExit(movie);
DialogClose(dialog2d);
DialogClose(dialog3d);
```

Make output file

Check recording time and calculate number of frames

Get georeference parameters for layers and reset to group projection defined by raster drape layer

Section computing arrays of viewer and center positions from input vectors is omitted here; see script

Start recording movie

Begin loop for each frame

Set viewer position along flight path

Set view center position

Redraw both views

Copy both views to frame

Loop to draw previous center points in frame

Draw current center point in frame

Update arrays of previous center point coordinates

Loop to draw previous viewer positions in frame

Draw current viewer position in frame

Update arrays of previous viewer position coordinates

Draw text annotation in frame

Add frame to movie

End of loop recording frames

Stop and exit movie, close dialogs