Street View Service

Overview

Google Street View provides panoramic 360 degree views from designated roads throughout its coverage area. Street View's API coverage is the same as that for the Google Maps application (http://maps.google.com/). The list of currently supported cities for Street View is available at the Google Maps Help Center.

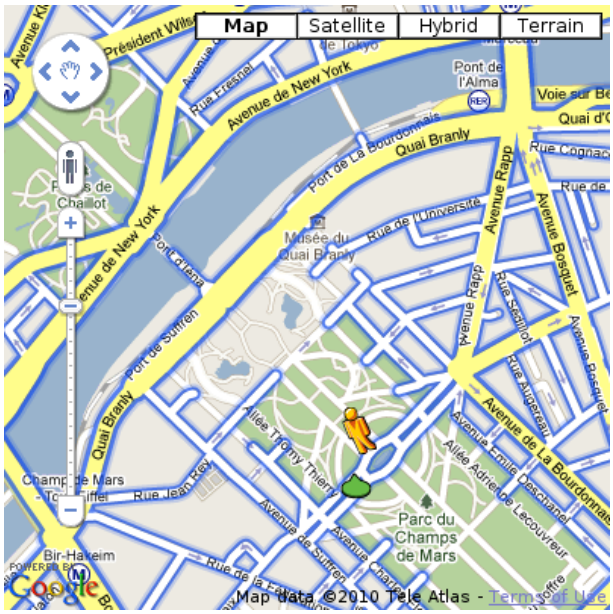
A sample Street View image is shown below.

The Google Maps Javascript API provides a Street View service for obtaining and manipulating the imagery used in Google Maps Street View. Unlike in the V2 API, the Street View service in the Maps Javascript API V3 is supported natively within the browser.

Street View Map Usage

Although Street View can be used within a standalone DOM element, it is most useful when indicating a location on a map. By default, Street View is enabled on a map, and a Street View *Pegman control* appears integrated within the navigation (zoom and pan) controls. You may hide this control within the map's MapOptions by setting streetViewControl to false. You may also change the default position of the Street View control by setting the Map's streetViewControlOptions.position property to a new ControlPosition.

The Street View Pegman control allows you to view Street View panoramas directly within the map. When clicking and holding the Pegman, the map updates to show Street View-enabled streets using blue outlines on the map:



Drop the Pegman marker onto a street and the map will update to display a Street View panorama of the indicated location.

The following sample adds a streetViewControl to a map of Boston near Fenway Park:

```
var fenway = new google.maps.LatLng(42.345573,-71.098326);
 var mapOptions = {
   center: fenway,
   zoom: 14,
   mapTypeId: google.maps.MapTypeId.ROADMAP
 };
var map = new google.maps.Map(document.getElementById("map_canvas"), mapOptions);
```

View example (streetview-map.html)

Street View Panoramas

Street View images are supported through use of the StreetViewPanorama object, which provides an API interface to a Street View "viewer." Each map contains a default Street View panorama, which you can retrieve by calling the map's getStreetView() method. When you add a Street View control to the map by setting its streetViewControl option to true, you automatically connect the Pegman control to this default Street View panorama.

You may also create your own StreetViewPanorama object and set the map to use that instead of the default, by setting the map's streetView property explicitly to that constructed object. You may wish to override the default panorama if you want to modify default behavior, such as the automatic sharing of overlays between the map and the panorama. (See Overlays within Street View below.)

Street View Containers

You may instead wish to display a StreetViewPanorama within a separate DOM element, often a element. Simply pass the DOM element within the StreetViewPanorama's constructor. For optimum display of images, we recommend a minimum size of 200 pixels by 200 pixels.

Note: although Street View functionality is designed to be used in conjunction with a map, this usage is not required. You may use a standalone Street View object without a map.

Street View Locations and Point-of-View (POV)

The StreetViewPanorama constructor also allows you to set the Street View location and point of view using the StreetViewOptions parameter. You may call setPosition() and setPov() on the object after construction to change its location and POV.

The Street View location defines the placement of the camera locus for an image, but it does not define the orientation of the camera for that image. For that purpose, the StreetViewPov object defines three properties:

- heading (default 0) defines the rotation angle around the camera locus in degrees relative from true north. Headings are measured clockwise (90 degrees is true east).

- pitch (default 0) defines the angle variance "up" or "down" from the camera's initial default pitch, which is often (but not always) flat horizontal. (For example, an image taken on a hill will likely exhibit a default pitch that is not horizontal.) Pitch angles are measured with positive values looking up (to +90 degrees straight up and orthogonal to the default pitch) and negative values looking down (to -90 degrees straight down and orthogonal to the default pitch).
- zoom (default 1) defines the zoom level of this view (effectively proscribing the "field of view") with 0 being fully zoomed-out. Most Street View locations support zoom levels from 0 to 3, inclusive.

The following code displays a map of Boston with an initial view of Fenway Park. Selecting the Pegman and dragging it to a supported location on the map will change the Street View panorama:

```
var fenway = new google.maps.LatLng(42.345573,-71.098326);
var mapOptions = {
 center: fenway,
 zoom: 14,
 mapTypeId: google.maps.MapTypeId.ROADMAP
};
var map = new google.maps.Map(
   document.getElementById("map_canvas"), mapOptions);
var panoramaOptions = {
 position: fenway,
 pov: {
  heading: 34,
  pitch: 10,
  zoom: 1
 }
};
var panorama = new  google.maps.StreetViewPanorama(document.getElementById("pano"), panoramaOptions);
map.setStreetView(panorama);
```

View example (streetview-simple.html)

Overlays within Street View

The default StreetViewPanorama object supports the native display of map overlays. Overlays generally appear at "street level" anchored at LatLng positions. (Markers will appear with their tails anchored to the location's horizontal plane within the Street View panorama for example.)

Currently, the types of overlays which are supported on Street View panoramas are limited to Markers, InfoWindows and custom OverlayViews. Overlays which you display on a map may be displayed on a Street View panorama by treating the panorama as a substitute for the Map object, calling setMap() and passing the StreetViewPanorama as an argument instead of a map. Info windows similarly may be opened within a Street View panorama by calling open(), passing the StreetViewPanorama() instead of a map.

Additionally, when creating a map with a default StreetViewPanorama, any markers created on a map are shared automatically with the map's associated Street View panorama, provided that panorama is visible. To retrieve the default Street View panorama, call getStreetView() on the Map object. Note that if you explicitly set the map's streetView property to a StreetViewPanorama of your own construction, you will override the default panorama and disable automatic overlay sharing.

The following example shows markers denoting various locations around Astor Place, New York City. Toggle the display to Street View to show the shared markers displaying within the StreetViewPanorama.

```
var map;
var panorama;
var astorPlace = new google.maps.LatLng(40.729884, -73.990988);
var busStop = new google.maps.LatLng(40.729559678851025, -73.99074196815491);
var cafe = new google.maps.LatLng(40.730031233910694, -73.99142861366272);
var bank = new google.maps.LatLng(40.72968163306612, -73.9911389350891);

function initialize() {

 // Set up the map
 var mapOptions = {
  center: astorPlace,
  zoom: 18,
  mapTypeId: google.maps.MapTypeId.ROADMAP,
  streetViewControl: false
 };
 map = new google.maps.Map(document.getElementById('map_canvas'), mapOptions);

 // Setup the markers on the map
```

```
  var cafeMarkerImage =
      new google.maps.MarkerImage('http://chart.apis.google.com/chart?chst=d_map_pin_icon&chld=cafe|FFFF00');
  var cafeMarker = new google.maps.Marker({
      position: cafe,
      map: map,
      icon: cafeMarkerImage,
      title: 'Cafe'
  });

  var bankMarkerImage =
      new google.maps.MarkerImage('http://chart.apis.google.com/chart?chst=d_map_pin_icon&chld=dollar|FFFF00'
  var bankMarker = new google.maps.Marker({
      position: bank,
      map: map,
      icon: bankMarkerImage,
      title: 'Bank'
  });

  var busMarkerImage =
      new google.maps.MarkerImage('http://chart.apis.google.com/chart?chst=d_map_pin_icon&chld=bus|FFFF00');
  var busMarker = new google.maps.Marker({
      position: busStop,
      map: map,
      icon: busMarkerImage,
      title: 'Bus Stop'
  });

  // We get the map's default panorama and set up some defaults.
  // Note that we don't yet set it visible.
  panorama = map.getStreetView();
  panorama.setPosition(astorPlace);
  panorama.setPov({
    heading: 265,
    zoom:1,
    pitch:0}
  );
}

function toggleStreetView() {
  var toggle = panorama.getVisible();
  if (toggle == false) {
    panorama.setVisible(true);
  } else {
    panorama.setVisible(false);
  }
}
```

View example (streetview-overlays.html)

Street View Events

When navigating between Street View or manipulating its orientation, you may wish to monitor several events that indicate changes to the StreetViewPanorama's state:

- pano_changed fires whenever the individual pano ID changes. This event does not guarantee that any associated data within the panorama (such as the links) has also changed by the time this event is triggered; this event only indicates that a pano ID has changed. Note that the pano ID (which you can use to reference this panorama) is only stable within the current browser session.
- position_changed fires whenever the underlying (LatLng) position of the panorama changes. Rotating a panorama will not trigger this event. Note that you could change a panorama's underlying position without changing the associated pano ID, since the API will automatically associate the nearest pano ID to the panorama's position.
- pov_changed fires whenever the Street View's StreetViewPov changes. Note that this event may fire while the position, and pano ID, remain stable.
- links_changed fires whenever the Street View's links change. Note that this event may fire asynchronously after a change in the pano ID indicated through pano_changed.
- visible_changed fires whenever the Street View's visibility changes. Note that this event may fire asynchronously after a change in the pano ID indicated through pano_changed.

The following code illustrates how these events can be handled to collect data about the underlying StreetViewPanorama:

```
var caffe = new google.maps.LatLng(37.869085,-122.254775);

function initialize() {

 var panoramaOptions = {
  position:caffe,
  pov: {
   heading: 270,
   pitch:0,
   zoom:1
  },
  visible:true
 };
 var panorama = new google.maps.StreetViewPanorama(document.getElementById("pano"), panoramaOptions);

 google.maps.event.addListener(panorama, 'pano_changed', function() {
    var panoCell = document.getElementById('pano_cell');
    panoCell.firstChild.nodeValue = panorama.getPano();
 });

 google.maps.event.addListener(panorama, 'links_changed', function() {
    var linksTable = document.getElementById('links_table');
    while(linksTable.hasChildNodes()) {
      linksTable.removeChild(linksTable.lastChild);
    };
    var links =  panorama.getLinks();
    for (var i in links) {
      var row = document.createElement("tr");
      linksTable.appendChild(row);
      var hCell = document.createElement("td");
      var hText = "Link: " + i + "";
      hCell.innerHTML = hText;
      var vCell = document.createElement("td");
      var vText = links[i].description;
      vCell.innerHTML = vText;
      linksTable.appendChild(hCell);
      linksTable.appendChild(vCell);
    }
 });

 google.maps.event.addListener(panorama, 'position_changed', function() {
    var positionCell = document.getElementById('position_cell');
        positionCell.firstChild.nodeValue = panorama.getPosition();
 });

 google.maps.event.addListener(panorama, 'pov_changed', function() {
    var headingCell = document.getElementById('heading_cell');
    var pitchCell = document.getElementById('pitch_cell');
    headingCell.firstChild.nodeValue = panorama.getPov().heading;
    pitchCell.firstChild.nodeValue = panorama.getPov().pitch;
 });

}
```

View example (streetview-events.html)

Street View Controls

When displaying a StreetViewPanorama, a variety of controls appear on the panorama by default. You can enable or disable these controls by setting their appropriate fields within the Street View's StreetViewPanoramaOptions to true or false:

- A panControl provides a way to rotate the panorama. This control appears by default as a standard integrated compass and pan control. You may alter the control's position by providing PanControlOptions within the panControlOptions field.
- A zoomControl provides a way to zoom within the image. This control appears by default below the pan control. You may alter the control's appearance by providing ZoomControlOptions within the zoomControlOptions field.
- An addressControl provides a textual overlay indicating the address of the associated location. You may alter the control's appearance by providing StreetViewAddressControlOptions within the addressControlOptions field.
- A linksControl provides guide arrows on the image for traveling to adjacent panorama images.

The following example alters the controls displayed within the associated Street View and removes the view's links:

```
var fenway = new google.maps.LatLng(42.345573,-71.098326);

// Note: constructed panorama objects have visible: true
// set by default.
var panoOptions = {
  position: fenway,
  addressControlOptions: {
    position: google.maps.ControlPosition.BOTTOM
  },
  linksControl: false,
  panControl: false,
  zoomControlOptions: {
    style: google.maps.ZoomControlStyle.SMALL
  },
  enableCloseButton: false,
  visible:true
};

var panorama = new google.maps.StreetViewPanorama(
  document.getElementById("pano"), panoOptions);
```

View example (streetview-controls.html)

Directly Accessing Street View Data

You may wish to programmatically determine the availability of Street View data, or return information about particular panoramas, without requiring direct manipulation of a map/panorama. You may do so using the StreetViewService object, which provides an interface to the data stored in Google's Street View service.

Street View Service Requests

Accessing the Street View service is asynchronous, since the Google Maps API needs to make a call to an external server. For that reason, you need to pass a *callback* method to execute upon completion of the request. This callback method processes the result.

You may initiate two types of requests to the StreetViewService:

- getPanoramaById() returns panorama data given a reference ID which uniquely identifies the panorama. Note that these reference IDs are only stable within the current browser session.
- getPanoramaByLocation() searches for panorama data over a given area, given a passed LatLng and the radius (in meters) over which to search. If the radius is 50 meters or less, the panorama returned will be the nearest panorama to the given location.

Street View Service Responses

Both getPanoramaByLocation() and getPanoramaById() specify a *callback* function to execute upon retrieval of a result from the Street View service. This callback function returns a set of panorama data within a StreetViewPanoramaData object and a StreetViewStatus code denoting the status of the request, in that order.

A StreetViewPanoramaData object specification contains meta-data about a Street View panorama of the following form:

```
{
  "location": {
    "latLng": LatLng,
    "description": string,
    "pano": string
  },
  "copyright": string,
  "links": [{
      "heading": number,
      "description": string,
      "pano": string,
      "roadColor": string,
      "roadOpacity": number
    }],
  "tiles": {
    "worldSize": Size,
    "tileSize": Size,
```

```
    "centerHeading": number
  }
}
```

Note that this data object is not a StreetViewPanorama object itself. To create a Street View object using this data, you would need to create a StreetViewPanorama and call setPano(), passing it the ID as noted in the returned location.pano field.

The status code may return one of the following values:

- OK indicates that the service found a matching panorama.
- ZERO_RESULTS indicates that the service could not find a matching panorama with the passed criteria.
- UNKNOWN_ERROR indicates that a Street View request could not be processed, though the exact reason is unknown.

The following code creates a StreetViewService that responds to user clicks on a map by creating markers which, when clicked, display a StreetViewPanorama of that location. The code uses the contents of StreetViewPanoramaData returned from the service.

```
var map;
var berkeley = new google.maps.LatLng(37.869085,-122.254775);
var sv = new google.maps.StreetViewService();

var panorama;

function initialize() {

  panorama = new google.maps.StreetViewPanorama(document.getElementById("pano"));

  // Set up the map
  var mapOptions = {
    center: berkeley,
    zoom: 16,
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    streetViewControl: false
  };
  map = new google.maps.Map(document.getElementById('map_canvas'),
      mapOptions);

  // getPanoramaByLocation will return the nearest pano when the
  // given radius is 50 meters or less.
  google.maps.event.addListener(map, 'click', function(event) {
      sv.getPanoramaByLocation(event.latLng, 50, processSVData);
  });
}

function processSVData(data, status) {
  if (status == google.maps.StreetViewStatus.OK) {
    var marker = new google.maps.Marker({
      position: data.location.latLng,
      map: map,
      title: data.location.description
    });

    google.maps.event.addListener(marker, 'click', function() {

      var markerPanoID = data.location.pano;
      // Set the Pano to use the passed panoID
      panorama.setPano(markerPanoID);
      panorama.setPov({
        heading: 270,
        pitch: 0,
        zoom: 1
      });
      panorama.setVisible(true);
    });
  }
}
```

View example (streetview-service.html)

Providing Custom StreetView Panoramas

The Maps Javascript API V3 supports the display of custom panoramas within the StreetViewPanorama object. Using custom panoramas, you can display the interior of buildings, views from scenic locations, or anything from your imagination. You can even link these custom panoramas to Google's existing Street View panoramas.

Setting up a set of custom panorama imagery involves the following steps:

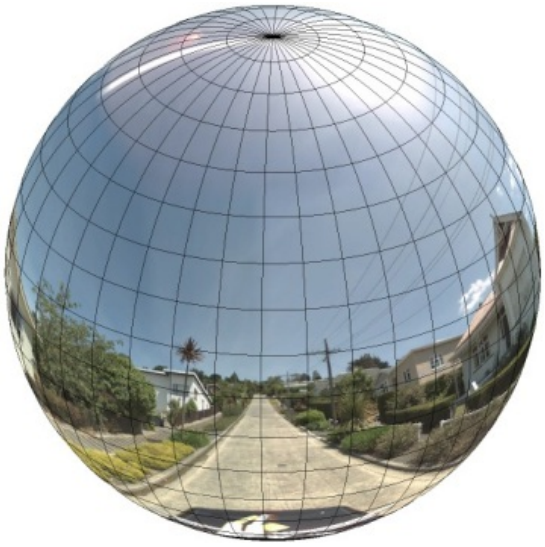
- Create a base panoramic image for each custom panorama. This base image should be at the highest resolution image with which you wish to serve zoomed in imagery.
- (Optional, but recommended) Create a set of panoramic tiles at different zoom levels from the basic image.
- Create links between your custom panoramas.
- (Optional) Designate "entry" panoramas within Google's existing Street View imagery and customize links to/from the custom set to the standard set.
- Define metadata for each panorama image within a StreetViewPanoramaData object.
- Implement a method which determines the custom panorama data and images and designate that method as your custom handler within the StreetViewPanorama object.

The following sections explain this process.

Creating Custom Panoramas

Each Street View panorama is an image or set of images that provides a full 360 degree view from a single location. The StreetViewPanorama object uses images that conform to the equirectangular (Plate Carrée) projection. Such a projection contains 360 degrees of horizontal view (a full wrap-around) and 180 degrees of vertical view (from straight up to straight down). These fields of view result in an image with an aspect ratio of 2:1. A full wrap-around panorama is shown below.

Panorama images are generally obtained by taking multiple photos from one position and stitching them together using panorama software. (See Wikipedia's Comparison of photo stitching applications for more information.) Such images should share a single "camera" locus, from which each of the panorama images are taken. The resulting 360 degree panorama can then define a projection on a sphere with the image wrapped to the two-dimensional surface of the sphere.



Treating the panorama as a projection on a sphere with a rectilinear coordinate system is advantageous when dividing up the image into rectilinear *tiles*, and serving images based on computed tile coordinates.

Creating Custom Panorama Tiles

Street View also supports different levels of image detail through the use of a zoom control, which allows you to zoom in and out from the default view. Generally, Street View provides five levels of zoom resolution for any given panorama image. If you were to rely on a single panorama image to serve all zoom levels, such an image would either necessarily be quite large and significantly slow down your application, or be of such poor resolution at higher zoom levels that you would serve a poorly pixellated image. Luckily, however, we can use a similar design pattern used to serve Google's map tiles at different zoom levels to provide appropriate resolution imagery for panoramas at each zoom level.

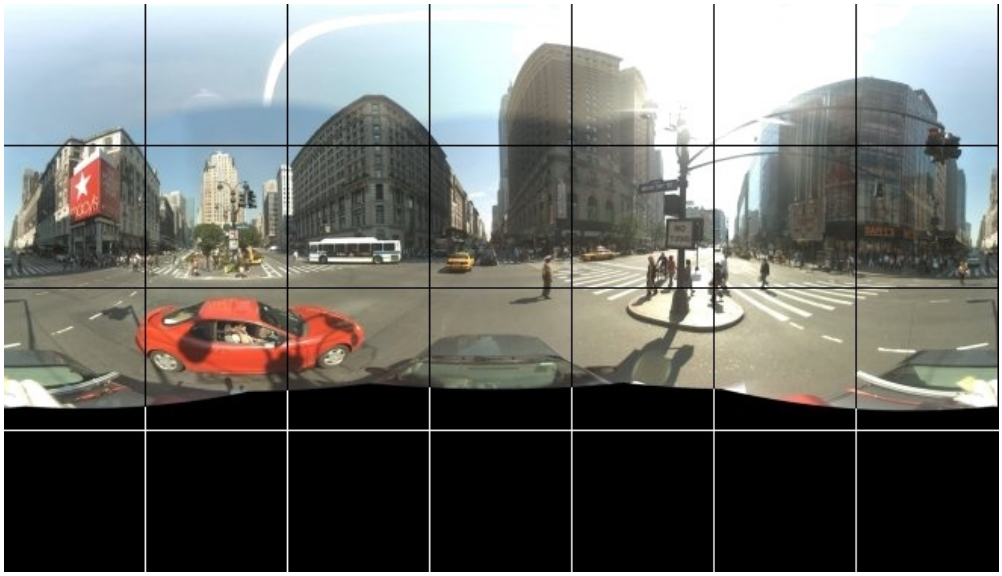
When a StreetViewPanorama first loads, by default it displays an image consisting of 25% (90 degrees of arc) of the horizontal breadth of the panorama at zoom level 1. This view corresponds roughly with a normal human field of view. Zooming "out" from this default view essentially provides a wider arc, while zooming in narrows the field of a view to a smaller arc. The StreetViewPanorama automatically calculates the appropriate field of view for the selected zoom level, and then selects imagery most appropriate for that resolution by selecting a tile set that roughly matches the dimensions of the horizontal field of view. The following fields of view map to Street View zoom levels:

Street View zoom level	Field of View (degrees)
0	180
1 (default)	90
2	45
3	22.5
4	11.25

Note that the size of the image shown within Street View is entirely dependent on the screen size (width) of the Street View container. If you provide a wider container, the service will still provide the same field of view for any given zoom level, though it may select tiles more appropriate for that resolution instead.

Because each panorama consists of an equirectangular projection, creating panorama tiles is relatively easy. As the projection provides an image with an aspect ratio of 2:1, tiles with 2:1 ratios are easier to use, though square tiles may provide better performance on square maps (since the field of view will be square).

For 2:1 tiles, a single image encompassing the entire panorama represents the entire panorama "world" (the base image) at zoom level 0, with each increasing zoom level offering $4^{zoomLevel}$ tiles. (E.g. at zoom level 2, the entire panorama consists of 16 tiles.) Note: zoom levels in Street View tiling do not match directly to zoom levels as provided using the Street View control; instead the Street View control zoom levels select a Field of View (FoV), from which appropriate tiles are selected.



Generally, you will want to name your image tiles so they can be selected programmatically. Such a naming scheme is discussed below in Handling Custom Panorama Requests.

Handling Custom Panorama Requests

Custom Panorama usage is indicated by registering a custom panorama method within the StreetViewPanoramaOptions panoProvider field or explicitly calling StreetViewPanorama.registerPanoProvider(). The panorama provider method is a function that returns a StreetViewPanoramaData object and has the following signature:

```
Function(pano,zoom,tileX,tileY):StreetViewPanoramaData
```

A StreetViewPanoramaData is an object of the following form:

```
{
  copyright: string,
  location: {
    description: string,
    latLng: google.maps.LatLng,
    pano: string
  },
  tiles: {
    tileSize: google.maps.Size,
    worldSize: google.maps.Size,
    heading: number,
    getTileUrl: Function
  },
  links: [
    description: string,
    heading: number,
    pano: string,
    roadColor: string,
    roadOpacity: number
  ]
}
```

You can display a custom panorama simply by setting the StreetViewPanorama's pano propety to a custom value, setting the panoProvider, and then handling that custom pano value within the custom panorama provider method, constructing a StreetViewPanoramaData object and returning it.

> Note: do not directly set a position on the StreetViewPanorama when you wish to display custom panoramas, as such a position will instruct the Street View service to request the default Street View imagery close to that location. Instead, set this position within the custom StreetViewPanoramaData's location.latLng field.

The following example displays a custom Panorama of the Google Sydney office. Note that we don't use a map (or default StreetView imagery) here at all:

```
var panorama;

function initialize() {

  // Set up Street View and initially set it visible. Register the
  // custom panorama provider function. Set the StreetView to display
  // the custom panorama 'reception' which we check for below.
  var panoOptions = {
    pano: 'reception',
    visible: true,
    panoProvider: getCustomPanorama
  }

  panorama = new google.maps.StreetViewPanorama(
      document.getElementById('pano_canvas'),panoOptions);
}

// Return a pano image given the panoID.
function getCustomPanoramaTileUrl(pano,zoom,tileX,tileY) {

  // Note: robust custom panorama methods would require tiled pano data.
  // Here we're just using a single tile, set to the tile size and equal
  // to the pano "world" size.
  return 'https://developers.google.com/maps/documentation/javascript/examples/images/panoReception1024.jpg'
}

// Construct the appropriate StreetViewPanoramaData given
// the passed pano IDs.
function getCustomPanorama(pano,zoom,tileX,tileY) {

   switch(pano) {

    case 'reception':
```

```
    return {
      location: {
        pano: 'reception',
        description: "Google Sydney - Reception",
        latLng: sydneyOffice
      },
      // The text for the copyright control.
      copyright: 'Imagery (c) 2010 Google',
      // The definition of the tiles for this panorama.
      tiles: {
        tileSize: new google.maps.Size(1024, 512),
        worldSize: new google.maps.Size(1024, 512),
        // The heading at the origin of the panorama tile set.
        centerHeading: 105,
        getTileUrl: getCustomPanoramaTileUrl
      }
    };
    break;
  }
}
```

View example (streetview-custom-simple.html)

Note that we only returned one image in the previous example and that zooming in using that image resulted in poor resolution. Instead, we could offer a tile set by creating tile images and modifying the panoProvider to return the appropriate tile given the passed panorama ID, zoom level, and panorama tile coordinate.

Since image selection depends on these passed values, it is useful to name images that can be selected programmatically given those passed values, such as *pano_zoom_tileX_tileY*.png.

The following example is slightly augmented to include two levels of zoom, and additionally modifies the default Street View links to add a link to the custom imagery:

```
var map;
var panorama;

// The latlng of the entry point to the Google office on the road.
var sydneyOffice = new google.maps.LatLng(-33.867386, 151.195767);

// The panorama that will be used as the entry point to the custom
// panorama set. We will test for the presence of this value when
// updating the entry pano to add our custom links.
var entryPanoId = null;

function initialize() {

  startApplication();

  // Define how far to search for an initial pano from a location, in meters.
  var panoSearchRadius = 50;

  // Create a StreetViewService object.
  var client = new google.maps.StreetViewService();

  // Compute the nearest panorama to the Google Sydney office
  // using the service and store that pano ID. Once that value
  // is determined, load the application.
  client.getPanoramaByLocation(sydneyOffice, panoSearchRadius, function(result, status) {
    if (status == google.maps.StreetViewStatus.OK) {
      entryPanoId = result.location.pano;
    }
  });
}

function startApplication() {

  // Set up the map and enable the Street View control.
  var mapOptions = {
    center: sydneyOffice,
    zoom: 16,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
```

```
  map = new google.maps.Map(document.getElementById('map_canvas'), mapOptions);
   // Get the default StreetViewPanorama object.
  panorama = map.getStreetView();
   // Set up Street View and initially set it visible. Register the
  // custom panorama provider function. Update the StreetViewPanorama.
  var panoOptions = {
    position: sydneyOffice,
    visible: true,
    panoProvider:  getCustomPanorama
  }
  panorama.setOptions(panoOptions);

  // We'll monitor the links_changed event to check if the current pano is either
  // a custom pano or our entry pano.
  google.maps.event.addListener(panorama, 'links_changed', createCustomLinks);
}

// Return a pano image given the panoID.
function getCustomPanoramaTileUrl(pano,zoom,tileX,tileY) {
  return 'images/panoReception1024-' + zoom + '-' + tileX + '-' +tileY + '.jpg';
}

// Construct the appropriate StreetViewPanoramaData given
// the passed pano IDs. Note that if no cases match here,
// we return null, which instructs the service to serve
// default pano data.
function getCustomPanorama(pano,zoom,tileX,tileY) {
   var center;
   switch(pano) {
     case 'reception':
      return {
        location: {
          pano: 'reception',
          description: "Google Sydney - Reception",
          latLng: sydneyOffice
        },
        links: [
          ],
        // The text for the copyright control.
        copyright: 'Imagery (c) 2010 Google',
        // The definition of the tiles for this panorama.
        tiles: {
          tileSize: new google.maps.Size(1024, 512),
          worldSize: new google.maps.Size(1024, 512),
          // The heading at the origin of the panorama tile set.
          centerHeading: 105,
          getTileUrl: getCustomPanoramaTileUrl
        }
      };
      break;
    default:
      return null;
  }
}

function createCustomLinks() {

  if (entryPanoId) {
    var links = panorama.getLinks();
    var panoId = panorama.getPano();
     switch(panoId) {
       case entryPanoId:
        // Adding a link in the view from the entrance of the building to
        // reception.
        links.push({
          'heading': 25,
          'description' : 'Google Sydney',
          'pano' : 'reception'
        });
        break;
      case 'reception':
        // Adding a link in the view from the entrance of the office
        // with an arrow pointing at 100 degrees, with a text of "Exit"
        // and loading the street entrance of the building pano on click.
```

```
        links.push({
          'heading': 195,
          'description' : 'Exit',
          'pano' : entryPanoId
        });
        break;
      default:
        return;
    }
  }
}
```

View example (streetview-custom-tiles.html)