```
    1: <!DOCTYPE html>
    2:
    3: <html>
    4:   <head>
    5:     <link href="service.css" rel="stylesheet" type="text/css">
    6:     <script src="http://www.google.com/jsapi?key=ABQIAAAA8igYd929VTmOEMLNjNyP1xQIE4MyTYdaqjM5EsvAZQBbaRMlYRS9jJaf64
VoDAABoTCl-_zJ-dl3vg" type="text/javascript"></script>
    7:     <script src="math3d.js" type="text/javascript"></script>
    8:     <script src="buildings.js" type="text/javascript"></script>
    9:     <script src="houses.js" type="text/javascript"></script>
   10:     <script src="passengers.js" type="text/javascript"></script>
   11:     <script src="shuttle.js" type="text/javascript"></script>
   12:     <script src="service.js" type="text/javascript"></script>
   13:     <title>CS50 Shuttle</title>
   14:   </head>
   15:   <body onkeydown="return keystroke(event, true);" onkeyup="return keystroke(event, false);" onload="load();" onunl
oad="unload();">
   16:     <div id="left">
   17:       <div id="earth"></div>
   18:     </div>
   19:     <div id="right">
   20:       <div id="logo">
   21:         CS50 Shuttle
   22:       </div>
   23:       <div id="instructions">
   24:         <a href="javascript:alert('Vroom vroom!  Don\'t click the 3D Earth, else the engine will stall, and you\'ll
 need to re-start it.\n\nMove Forward: W\nMove Backward: S\nTurn Left: left arrow\nTurn Right: right arrow\nSlide Left: A\
nSlide Right: D\nLook Downward: down arrow\nLook Upward: up arrow');">start engine</a>
   25:       </div>
   26:       <div id="controls">
   27:         <input onclick="pickup();" type="button" value="Pick Up">
   28:         <input onclick="dropoff();" type="button" value="Drop Off">
   29:       </div>
   30:       <div id="announcements">
   31:         no announcements at this time
   32:       </div>
   33:       <div id="seats">
   34:         <ol>
   35:           <li>Empty Seat</li>
   36:           <li>Empty Seat</li>
   37:           <li>Empty Seat</li>
   38:           <li>Empty Seat</li>
   39:           <li>Empty Seat</li>
   40:           <li>Empty Seat</li>
   41:           <li>Empty Seat</li>
```



```
   42:           <li>Empty Seat</li>
   43:           <li>Empty Seat</li>
   44:           <li>Empty Seat</li>
   45:           <li>Empty Seat</li>
   46:           <li>Empty Seat</li>
   47:           <li>Empty Seat</li>
   48:           <li>Empty Seat</li>
   49:           <li>Empty Seat</li>
   50:           <li>Empty Seat</li>
   51:           <li>Empty Seat</li>
   52:           <li>Empty Seat</li>
   53:           <li>Empty Seat</li>
   54:           <li>Empty Seat</li>
   55:           <li>Empty Seat</li>
   56:           <li>Empty Seat</li>
   57:           <li>Empty Seat</li>
   58:           <li>Empty Seat</li>
   59:           <li>Empty Seat</li>
   60:           <li>Empty Seat</li>
   61:           <li>Empty Seat</li>
   62:           <li>Empty Seat</li>
   63:           <li>Empty Seat</li>
   64:           <li>Empty Seat</li>
   65:           <li>Empty Seat</li>
   66:           <li>Empty Seat</li>
   67:           <li>Empty Seat</li>
   68:           <li>Empty Seat</li>
   69:           <li>Empty Seat</li>
   70:         </ol>
   71:       </div>
   72:       <div id="map"></div>
   73:     </div>
   74:   </body>
   75: </html>
```

```
 1: /*****************************************************************************
 2:  * service.css
 3:  *
 4:  * Computer Science 50
 5:  * Problem Set 8
 6:  *
 7:  * Global stylesheet.
 8:  *****************************************************************************/
 9:
10: body
11: {
12:     font-family: sans-serif;
13:     height: 100%;
14:     margin: 0;
15: }
16:
17: div#announcements
18: {
19:     font-size: smaller;
20:     margin: 5px;
21:     text-align: center;
22: }
23:
24: div#controls
25: {
26:     margin-bottom: 10px;
27:     margin-top: 10px;
28:     text-align: center;
29: }
30:
31: div#earth
32: {
33:     border-right: 1px #CCCCCC solid;
34:     height: 100%;
35:     margin-right: 320px;
36: }
37:
38: div#instructions
39: {
40:     font-size: smaller;
41:     text-align: center;
42: }
43:
44: div#left
45: {
```



```
46:     float: right;
47:     height: 100%;
48:     width: 100%
49: }
50:
51: div#logo
52: {
53:     margin-bottom: 10px;
54:     font-size: larger;
55:     font-weight: bold;
56:     margin-top: 10px;
57:     text-align: center;
58: }
59:
60: div#map
61: {
62:     bottom: 0;
63:     height: 320px;
64:     position: absolute;
65:     width: 320px;
66: }
67:
68: div#right
69: {
70:     float: right;
71:     height: 100%;
72:     margin-right: -100%;
73:     width: 320px;
74: }
75:
76: div#seats
77: {
78:     background-color: #FFFFFF;
79:     bottom: 320px;
80:     border-bottom: 1px #CCCCCC solid;
81:     border-top: 1px #CCCCCC solid;
82:     font-size: smaller;
83:     height: 160px;
84:     overflow-y: scroll;
85:     position: absolute;
86:     width: 320px;
87: }
88:
89: html
90: {
```

```
    height: 100%;
}
```



```
/**************************************************************************
 * service.js
 *
 * Computer Science 50
 * Problem Set 8
 *
 * Implements a shuttle service.
 **************************************************************************/


// default height
var HEIGHT = 0.8;

// default latitude
var LATITUDE = 42.3745615030193;

// default longitude
var LONGITUDE = -71.11803936751632;

// default heading
var HEADING = 1.757197490907891;

// default number of seats
var SEATS = 35;

// default velocity
var VELOCITY = 50;


// global reference to shuttle's marker on 2D map
var bus = null;

// global reference to 3D Earth
var earth = null;

// global reference to 2D map
var map = null;

// global reference to shuttle
var shuttle = null;


// load version 1 of the Google Earth API
google.load("earth", "1");

```

```
46: // load version 3 of the Google Maps API
47: google.load("maps", "3", {other_params: "sensor=false"});
48:
49:
50: /*
51:  * void
52:  * dropoff()
53:  *
54:  * Drops up passengers if their stop is nearby.
55:  */
56:
57: function dropoff()
58: {
59:     alert("TODO");
60: }
61:
62:
63: /*
64:  * void
65:  * failureCB(errorCode)
66:  *
67:  * Called if Google Earth fails to load.
68:  */
69:
70: function failureCB(errorCode)
71: {
72:     // report error unless plugin simply isn't installed
73:     if (errorCode != ERR_CREATE_PLUGIN)
74:     {
75:         alert(errorCode);
76:     }
77: }
78:
79:
80: /*
81:  * void
82:  * frameend()
83:  *
84:  * Handler for Earth's frameend event.
85:  */
86:
87: function frameend()
88: {
89:     shuttle.update();
90: }
```



```
91:
92:
93: /*
94:  * void
95:  * initCB()
96:  *
97:  * Called once Google Earth has loaded.
98:  */
99:
100: function initCB(instance)
101: {
102:     // retain reference to GEPlugin instance
103:     earth = instance;
104:
105:     // specify the speed at which the camera moves
106:     earth.getOptions().setFlyToSpeed(100);
107:
108:     // show buildings
109:     earth.getLayerRoot().enableLayerById(earth.LAYER_BUILDINGS, true);
110:
111:     // prevent mouse navigation in the plugin
112:     earth.getOptions().setMouseNavigationEnabled(false);
113:
114:     // instantiate shuttle
115:     shuttle = new Shuttle({
116:      heading: HEADING,
117:      height: HEIGHT,
118:      latitude: LATITUDE,
119:      longitude: LONGITUDE,
120:      planet: earth,
121:      velocity: VELOCITY
122:     });
123:
124:     // synchronize camera with Earth
125:     google.earth.addEventListener(earth, "frameend", frameend);
126:
127:     // synchronize map with Earth
128:     google.earth.addEventListener(earth.getView(), "viewchange", viewchange);
129:
130:     // update shuttle's camera
131:     shuttle.updateCamera();
132:
133:     // show Earth
134:     earth.getWindow().setVisibility(true);
135:
```

```
136:     // populate Earth with passengers and houses
137:     populate();
138: }
139:
140:
141: /*
142:  * boolean
143:  * keystroke(event, state)
144:  *
145:  * Handles keystrokes.
146:  */
147:
148: function keystroke(event, state)
149: {
150:     // ensure we have event
151:     if (!event)
152:     {
153:         event = window.event;
154:     }
155:
156:     // left arrow
157:     if (event.keyCode == 37)
158:     {
159:         shuttle.states.turningLeftward = state;
160:         return false;
161:     }
162:
163:     // up arrow
164:     else if (event.keyCode == 38)
165:     {
166:         shuttle.states.tiltingUpward = state;
167:         return false;
168:     }
169:
170:     // right arrow
171:     else if (event.keyCode == 39)
172:     {
173:         shuttle.states.turningRightward = state;
174:         return false;
175:     }
176:
177:     // down arrow
178:     else if (event.keyCode == 40)
179:     {
180:         shuttle.states.tiltingDownward = state;
```



```
181:         return false;
182:     }
183:
184:     // A, a
185:     else if (event.keyCode == 65 || event.keyCode == 97)
186:     {
187:         shuttle.states.slidingLeftward = state;
188:         return false;
189:     }
190:
191:     // D, d
192:     else if (event.keyCode == 68 || event.keyCode == 100)
193:     {
194:         shuttle.states.slidingRightward = state;
195:         return false;
196:     }
197:
198:     // S, s
199:     else if (event.keyCode == 83 || event.keyCode == 115)
200:     {
201:         shuttle.states.movingBackward = state;
202:         return false;
203:     }
204:
205:     // W, w
206:     else if (event.keyCode == 87 || event.keyCode == 119)
207:     {
208:         shuttle.states.movingForward = state;
209:         return false;
210:     }
211:
212:     return true;
213: }
214:
215:
216: /*
217:  * void
218:  * load()
219:  *
220:  * Loads application.
221:  */
222:
223: function load()
224: {
225:     // embed 2D map in DOM
```

```
226:     var latlng = new google.maps.LatLng(LATITUDE, LONGITUDE);
227:     map = new google.maps.Map(document.getElementById("map"), {
228:      center: latlng,
229:      disableDefaultUI: true,
230:      mapTypeId: google.maps.MapTypeId.ROADMAP,
231:      navigationControl: true,
232:      scrollwheel: false,
233:      zoom: 17
234:     });
235:
236:     // prepare shuttle's icon for map
237:     bus = new google.maps.Marker({
238:      icon: "http://maps.gstatic.com/intl/en_us/mapfiles/ms/micons/bus.png",
239:      map: map,
240:      title: "you are here"
241:     });
242:
243:     // embed 3D Earth in DOM
244:     google.earth.createInstance("earth", initCB, failureCB);
245: }
246:
247:
248: /*
249:  * void
250:  * pickup()
251:  *
252:  * Picks up nearby passengers.
253:  */
254:
255: function pickup()
256: {
257:     alert("TODO");
258: }
259:
260:
261: /*
262:  * void
263:  * populate()
264:  *
265:  * Populates Earth with passengers and houses.
266:  */
267:
268: function populate()
269: {
270:     // mark houses
```



```
271:     for (var house in HOUSES)
272:     {
273:         // plant house on map
274:         new google.maps.Marker({
275:          icon: "http://google-maps-icons.googlecode.com/files/home.png",
276:          map: map,
277:          position: new google.maps.LatLng(HOUSES[house].lat, HOUSES[house].lng),
278:          title: house
279:         });
280:     }
281:
282:     // get current URL, sans any filename
283:     var url = window.location.href.substring(0, (window.location.href.lastIndexOf("/")) + 1);
284:
285:     // scatter passengers
286:     for (var i = 0; i < PASSENGERS.length; i++)
287:     {
288:         // pick a random building
289:         var building = BUILDINGS[Math.floor(Math.random() * BUILDINGS.length)];
290:
291:         // prepare placemark
292:         var placemark = earth.createPlacemark("");
293:         placemark.setName(PASSENGERS[i].name + " to " + PASSENGERS[i].house);
294:
295:         // prepare icon
296:         var icon = earth.createIcon("");
297:         icon.setHref(url + "/passengers/" + PASSENGERS[i].username + ".jpg");
298:
299:         // prepare style
300:         var style = earth.createStyle("");
301:         style.getIconStyle().setIcon(icon);
302:         style.getIconStyle().setScale(5.0);
303:
304:         // prepare stylemap
305:         var styleMap = earth.createStyleMap("");
306:         styleMap.setNormalStyle(style);
307:         styleMap.setHighlightStyle(style);
308:
309:         // associate stylemap with placemark
310:         placemark.setStyleSelector(styleMap);
311:
312:         // prepare point
313:         var point = earth.createPoint("");
314:         point.setAltitudeMode(earth.ALTITUDE_RELATIVE_TO_GROUND);
315:         point.setLatitude(building.lat);
```

```
316:         point.setLongitude(building.lng);
317:         point.setAltitude(2.0);
318:
319:         // associate placemark with point
320:         placemark.setGeometry(point);
321:
322:         // add placemark to Earth
323:         earth.getFeatures().appendChild(placemark);
324:
325:         // add marker to map
326:         var marker = new google.maps.Marker({
327:          icon: "http://maps.gstatic.com/intl/en_us/mapfiles/ms/micons/man.png",
328:          map: map,
329:          position: new google.maps.LatLng(building.lat, building.lng),
330:          title: PASSENGERS[i].name + " at " + building.name
331:         });
332:     }
333: }
334:
335:
336: /*
337:  * void
338:  * viewchange()
339:  *
340:  * Handler for Earth's viewchange event.
341:  */
342:
343: function viewchange()
344: {
345:     // keep map centered on shuttle's marker
346:     var latlng = new google.maps.LatLng(shuttle.position.latitude, shuttle.position.longitude);
347:     map.setCenter(latlng);
348:     bus.setPosition(latlng);
349: }
350:
351:
352: /*
353:  * void
354:  * unload()
355:  *
356:  * Unloads Earth.
357:  */
358:
359: function unload()
360: {
```



```
361:     google.earth.removeEventListener(earth.getView(), "viewchange", viewchange);
362:     google.earth.removeEventListener(earth, "frameend", frameend);
363: }
```

```
 1: /**************************************************************************
 2:  * shuttle.js
 3:  *
 4:  * Computer Science 50
 5:  * Problem Set 8
 6:  *
 7:  * Implements a shuttle.  Based on
 8:  * http://earth-api-samples.googlecode.com/svn/trunk/demos/firstpersoncam/firstpersoncam.js.
 9:  **************************************************************************/
10:
11:
12: /*
13:  * void
14:  * Shuttle(config)
15:  *
16:  * Encapsulates a shuttle.
17:  */
18:
19: function Shuttle(config)
20: {
21:     // shuttle's position
22:     this.position = {
23:      altitude: 0,
24:      heading: config.heading,
25:      latitude: config.latitude,
26:      longitude: config.longitude
27:     };
28:
29:     // shuttle's states
30:     this.states = {
31:      flyingUpward: false,
32:      flyingDownward: false,
33:      movingBackward: false,
34:      movingForward: false,
35:      slidingLeftward: false,
36:      slidingRightward: false,
37:      tiltingDownward: false,
38:      tiltingUpward: false,
39:      turningLeftward: false,
40:      turningRightward: false
41:     };
42:
43:     // remember shuttle's planet
44:     this.planet = config.planet;
45:
```



```
46:     // remember shuttle's height
47:     this.height = config.height;
48:
49:     // remember shuttle's velocity
50:     this.velocity = config.velocity;
51:
52:     // initialize camera altitude to shuttle's height
53:     this.cameraAltitude = this.height;
54:
55:     // shuttle's initial Cartesian coordinates
56:     this.localAnchorCartesian =
57:      V3.latLonAltToCartesian([this.position.latitude, this.position.longitude, this.position.altitude]);
58:
59:     // heading angle and tilt angle are relative to local frame
60:     this.headingAngle = config.heading;
61:     this.tiltAngle = 0;
62:
63:     // initialize time
64:     this.lastMillis = (new Date()).getTime();
65:
66:     // used for bounce
67:     this.distanceTraveled = 0;
68: }
69:
70:
71: /*
72:  * number
73:  * distance(lat, lng)
74:  *
75:  * Calculates the distance in meters between the shuttle and specified coordinates.
76:  * Based on http://code.google.com/apis/maps/articles/mvcfun.html#makingitwork.
77:  */
78:
79: Shuttle.prototype.distance = function(lat, lng)
80: {
81:     var R = 6371;
82:     var dLat = (this.position.latitude - lat) * Math.PI / 180;
83:     var dLon = (this.position.longitude - lng) * Math.PI / 180;
84:     var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
85:      Math.cos(lat * Math.PI / 180) * Math.cos(this.position.latitude * Math.PI / 180) *
86:      Math.sin(dLon / 2) * Math.sin(dLon / 2);
87:     var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
88:     var d = R * c;
89:     return d * 1000;
90: };
```

```
 91:
 92:
 93: /*
 94:  * void
 95:  * Shuttle.prototype.update()
 96:  *
 97:  * Method that updates a shuttle's location.
 98:  */
 99:
100: Shuttle.prototype.update = function()
101: {
102:     this.planet.getWindow().blur();
103:
104:     // Update delta time (dt in seconds)
105:     var now = (new Date()).getTime();
106:     var dt = (now - this.lastMillis) / 1000.0;
107:     if (dt > 0.25)
108:     {
109:         dt = 0.25;
110:     }
111:     this.lastMillis = now;
112:
113:     // Update orientation and then position of camera based on user input.
114:     this.updateOrientation(dt);
115:     this.updatePosition(dt);
116:
117:     // Update camera.
118:     this.updateCamera();
119: };
120:
121:
122: /*
123:  * void
124:  * Shuttle.prototype.updateCamera()
125:  *
126:  * Method that updates a shuttle's camera.
127:  */
128:
129: Shuttle.prototype.updateCamera = function()
130: {
131:     // Will put in a bit of a stride if the camera is at or below 1.7 meters
132:     var bounce = 0;
133:     if (this.cameraAltitude <= this.height)
134:     {
135:         bounce = 1.5 * Math.abs(Math.sin(4 * this.distanceTraveled * Math.PI / 180));
```



```
136:     }
137:
138:     // calculate heading; keep angle in [-180, 180]
139:     var heading = this.headingAngle * 180 / Math.PI;
140:     while (heading < -180)
141:     {
142:         heading += 360;
143:     }
144:     while (heading > 180)
145:     {
146:         heading -= 360;
147:     }
148:
149:     // Update camera position.  Note that tilt at 0 is facing directly downwards.
150:     // We add 90 such that 90 degrees is facing forwards.
151:     var la = this.planet.createLookAt("");
152:     la.set(
153:      this.position.latitude,
154:      this.position.longitude,
155:      this.cameraAltitude + bounce,
156:      this.planet.ALTITUDE_RELATIVE_TO_GROUND,
157:      heading,
158:      this.tiltAngle * 180 / Math.PI + 120, /* tilt */
159:      0 /* altitude is constant */
160:     );
161:     this.planet.getView().setAbstractView(la);
162: };
163:
164:
165: /*
166:  * void
167:  * Shuttle.prototype.updateOrientation(dt)
168:  *
169:  * Method that updates a shuttle's orientation.
170:  */
171:
172: Shuttle.prototype.updateOrientation = function(dt)
173: {
174:     // Based on dt and input press, update turn angle.
175:     if (this.states.turningLeftward || this.states.turningRightward)
176:     {
177:         var turnSpeed = 60.0; // radians/sec
178:         if (this.states.turningLeftward)
179:         {
180:             turnSpeed *= -1.0;
```

```
   181:         }
   182:         this.headingAngle += turnSpeed * dt * Math.PI / 180.0;
   183:     }
   184:     if (this.states.tiltingUpward || this.states.tiltingDownward)
   185:     {
   186:         var tiltSpeed = 60.0; // radians/sec
   187:         if (this.states.tiltingDownward)
   188:         {
   189:             tiltSpeed *= -1.0;
   190:         }
   191:         this.tiltAngle = this.tiltAngle + tiltSpeed * dt * Math.PI / 180.0;
   192:
   193:         // Clamp
   194:         var tiltMax = 50.0 * Math.PI / 180.0;
   195:         var tiltMin = -90.0 * Math.PI / 180.0;
   196:         if (this.tiltAngle > tiltMax)
   197:         {
   198:             this.tiltAngle = tiltMax;
   199:         }
   200:         if (this.tiltAngle < tiltMin)
   201:         {
   202:             this.tiltAngle = tiltMin;
   203:         }
   204:     }
   205: }
   206:
   207:
   208: /*
   209:  * void
   210:  * Shuttle.prototype.updatePosition(dt)
   211:  *
   212:  * Method that updates a shuttle's position.
   213:  */
   214:
   215: Shuttle.prototype.updatePosition = function(dt)
   216: {
   217:     // Convert local lat/lon to a global matrix.  The up vector is
   218:     // vector = position - center of earth.  And the right vector is a vector
   219:     // pointing eastwards and the facing vector is pointing towards north.
   220:     var localToGlobalFrame = M33.makeLocalToGlobalFrame([this.position.latitude, this.position.longitude, this.posi
tion.altitude]);
   221:
   222:     // Move in heading direction by rotating the facing vector around
   223:     // the up vector, in the angle specified by the heading angle.
   224:     // Strafing is similar, except it's aligned towards the right vec.
```



```
   225:     var headingVec = V3.rotate(localToGlobalFrame[1], localToGlobalFrame[2], -this.headingAngle);

   226:     var rightVec = V3.rotate(localToGlobalFrame[0], localToGlobalFrame[2], -this.headingAngle);
   227:
   228:     // Calculate strafe/forwards
   229:     var strafe = 0;
   230:     if (this.states.slidingLeftward || this.states.slidingRightward)
   231:     {
   232:         var strafeVelocity = this.velocity / 2;
   233:         if (this.states.slidingLeftward)
   234:         {
   235:             strafeVelocity *= -1;
   236:         }
   237:         strafe = strafeVelocity * dt;
   238:     }
   239:     var forward = 0;
   240:     if (this.states.movingForward || this.states.movingBackward)
   241:     {
   242:         var forwardVelocity = this.velocity;
   243:         if (this.states.movingBackward)
   244:         {
   245:             forwardVelocity *= -1;
   246:         }
   247:         forward = forwardVelocity * dt;
   248:     }
   249:     if (this.states.flyingUpward)
   250:     {
   251:         this.cameraAltitude += 1.0;
   252:     }
   253:     else if (this.states.flyingDownward)
   254:     {
   255:         this.cameraAltitude -= 1.0;
   256:     }
   257:     this.cameraAltitude = Math.max(this.height, this.cameraAltitude);
   258:
   259:     // remember distance traveled
   260:     this.distanceTraveled += forward;
   261:
   262:     // Add the change in position due to forward velocity and strafe velocity.
   263:     this.localAnchorCartesian = V3.add(this.localAnchorCartesian, V3.scale(rightVec, strafe));
   264:     this.localAnchorCartesian = V3.add(this.localAnchorCartesian, V3.scale(headingVec, forward));
   265:
   266:     // Convert cartesian to Lat Lon Altitude for camera setup later on.
   267:     var localAnchorLla = V3.cartesianToLatLonAlt(this.localAnchorCartesian);
   268:     this.position.latitude = localAnchorLla[0];
```

```
269:     this.position.longitude = localAnchorLla[1];
270:     this.position.altitude = localAnchorLla[2];
271: }
```