Chapter 10

Wrapping Up the Single-Player Campaign

Our game framework now has almost everything we need to build a very nice single-player campaign: a level system, various units and buildings, intelligent movement using pathfinding, an economy, and finally combat.

Now it's time to add the finishing touches and wrap up our single-player campaign. We will first add sound effects such as explosions and voices to our game. We will then build several levels by combining and using the various elements that we developed over the past few chapters. You will see how these building blocks fall into place to create a complete game.

Let's get started. We will continue where we left off at the end of Chapter 9.

Adding Sound

RTS games have a lot more happening at the same time than games in other genres such as the physics game we developed in the first few chapters. If we are not careful, there is a possibility of overwhelming a player with so much audio input that it becomes a distraction and takes away from their immersion. For our game, we will focus on sounds that will make the player aware of essential events within the game.

- *Acknowledging commands*: Any time the player selects a unit and gives it a command, we will have the unit acknowledge that it received the command.
- *Messages*: Whenever the player receives either a system warning or a story line–driven notification, we will alert the player with a sound.
- *Combat*: We will add sounds during combat so that players instantly know that they are under attack somewhere on the map.

Setting Up Sounds

We will start by creating a `sounds` object inside `sounds.js`, as shown in Listing 10-1.

Listing 10-1. Creating a sounds Object (sounds.js)

```
var sounds = {
    list:{
        "bullet":["bullet1","bullet2"],
        "heatseeker":["heatseeker1","heatseeker2"],
        "fireball":["laser1","laser2"],
```

```
        "cannon-ball":["cannon1","cannon2"],
        "message-received":["message"],
        "acknowledge-attacking":["engaging"],
        "acknowledge-moving":["yup","roger1","roger2"],
    },
    loaded:{},
    init: function(){
        for(var soundName in this.list){
            var sound = {};
            sound.audioObjects = [];
            for (var i=0; i < this.list[soundName].length; i++) {
                sound.audioObjects.push(loader.loadSound('audio/' + this.list[soundName][i]));
            };
            this.loaded [soundName] = sound;
        }
    },
    play:function(soundName){
        var sound = sounds.loaded[soundName];
        if(sound && sound.audioObjects && sound.audioObjects.length>0){
            if(!sound.counter || sound.counter>= sound.audioObjects.length){
                sound.counter = 0;
            }
            var audioObject = sound.audioObjects[sound.counter];
            sound.counter++;
            audioObject.play();
        }
    }
};
```

Within the `sound` object, we start by declaring a list, which maps a sound name to one or more sound files. For example, the bullet sound maps to two files: `bullet1` and `bullet2`. You will notice that we don't specify the file extension (`.ogg` or `.mp3`). We let the `loader` object handle selecting the appropriate audio file extension for the browser.

Next we declare an `init()` method that iterates through the list of sounds, uses the `loader.loadSound()` method to load each audio file, and then creates an `audioObjects` array for each sound name. We then add this `sound` object to the loaded object.

Finally, we declare a `play()` method that looks up the appropriate sound object from the `loaded` array and then plays the audio object using its `play()` method. You will notice that we use a counter for each `sound` object to ensure that we iterate through the sounds for a given sound name so that a different sound is played each time `play()` is called. This allows us to play different versions of sounds for an event instead of hearing the same monotonous sound each time.

Next, we will add a reference to `sounds.js` inside the `<head>` section of `index.html`, as shown in Listing 10-2.

Listing 10-2. Referring to sounds.js (index.html)

```
<script src="js/sounds.js" type="text/javascript" charset="utf-8"></script>
```

Finally, we will load all these sounds when the game is initialized by calling the `init()` method from inside the `game` object's `init()` method, as shown in Listing 10-3.

Listing 10-3. Initializing the sounds Object Inside the game.init() Method (game.js)

```
init:function(){
    loader.init();
    mouse.init();
    sidebar.init();
    sounds.init();

    $('.gamelayer').hide();
    $('#gamestartscreen').show();

    game.backgroundCanvas = document.getElementById('gamebackgroundcanvas');
    game.backgroundContext = game.backgroundCanvas.getContext('2d');

    game.foregroundCanvas = document.getElementById('gameforegroundcanvas');
    game.foregroundContext = game.foregroundCanvas.getContext('2d');

    game.canvasWidth = game.backgroundCanvas.width;
    game.canvasHeight = game.backgroundCanvas.height;
},
```

Now that the sounds object is in place, we can start adding sounds for each event, starting with acknowledging commands.

Acknowledging Commands

We allow the player to give units several types of commands: attack, move, deploy, and guard. Any time a unit is sent an attack command, we will play the acknowledge-attacking sound. When the unit is sent any other command such as move or guard, we will play the acknowledge-moving sound.

We will do this by calling sounds.play() from inside the click() method of the mouse object, as shown in Listing 10-4.

Listing 10-4. Acknowledging Commands Inside the click() Method (mouse.js)

```
click:function(ev,rightClick){
    // Player clicked inside the canvas
    var clickedItem = this.itemUnderMouse();
    var shiftPressed = ev.shiftKey;

    if (!rightClick){ // Player left clicked
        // If the game is in deployBuilding mode, left clicking will deploy the building
        if (game.deployBuilding){
            if(game.canDeployBuilding){
                sidebar.finishDeployingBuilding();
            } else {
                game.showMessage("system","Warning! Cannot deploy building here.");
            }
            return;
        }
```

```
        if (clickedItem){
            // Pressing shift adds to existing selection. If shift is not pressed, clear
existing selection
            if(!shiftPressed){
                game.clearSelection();
            }
            game.selectItem(clickedItem,shiftPressed);
        }
    } else { // Player right clicked
        // If the game is in deployBuilding mode, right clicking will cancel deployBuilding mode
        if (game.deployBuilding){
            sidebar.cancelDeployingBuilding();
            return;
        }
        // Handle actions like attacking and movement of selected units
        var uids = [];
        if (clickedItem){ // Player right clicked on something
            if (clickedItem.type != "terrain"){
                if (clickedItem.team != game.team){ // Player right clicked on an enemy item
                    // identify selected items from players team that can attack
                    for (var i = game.selectedItems.length - 1; i >= 0; i--){
                        var item = game.selectedItems[i];
                        if(item.team == game.team && item.canAttack){
                            uids.push(item.uid);
                        }
                    };
                    // then command them to attack the clicked item
                    if (uids.length>0){
                        game.sendCommand(uids,{type:"attack", toUid:clickedItem.uid});
                        sounds.play("acknowledge-attacking");
                    }
                } else  { // Player right clicked on a friendly item
                    //identify selected items from players team that can move
                    for (var i = game.selectedItems.length - 1; i >= 0; i--){
                        var item = game.selectedItems[i];
                        if(item.team == game.team && (item.type == "vehicles" || item.type ==
"aircraft")){
                            uids.push(item.uid);
                        }
                    };
                    // then command them to guard the clicked item
                    if (uids.length>0){
                        game.sendCommand(uids,{type:"guard", toUid:clickedItem.uid});
                        sounds.play("acknowledge-moving");
                    }
                }
            } else if (clickedItem.name == "oilfield"){ // Player right clicked on an oilfield
                // identify the first selected harvester from players team (since only one can
deploy at a time)
                for (var i = game.selectedItems.length - 1; i >= 0; i--){
                    var item = game.selectedItems[i];
```

```
                    // pick the first selected harvester since only one can deploy at a time
                    if(item.team == game.team && (item.type == "vehicles" && item.name ==
"harvester")){
                        uids.push(item.uid);
                        break;
                    }
                };
                // then command it to deploy on the oilfield
                if (uids.length>0){
                    game.sendCommand(uids,{type:"deploy", toUid:clickedItem.uid});
                    sounds.play("acknowledge-moving");
                }
            }
        } else { // Player right clicked on the ground
            //identify selected items from players team that can move
            for (var i = game.selectedItems.length - 1; i >= 0; i--){
                var item = game.selectedItems[i];
                if(item.team == game.team && (item.type == "vehicles" || item.type == "aircraft")){
                    uids.push(item.uid);
                }
            };
            // then command them to move to the clicked location
            if (uids.length>0){
                game.sendCommand(uids,{type:"move", to:{x:mouse.gameX/game.gridSize, y:mouse.gameY/
game.gridSize}});
                sounds.play("acknowledge-moving");
            }
        }
    }
},
```

We call the `sounds.play()` method with the appropriate sound name whenever we send a game command.

One interesting thing to point out is that we play the sound when the command is sent out, not when it is received and processed. While this makes very little difference during the single-player campaign, it becomes important during multiplayer.

Usually, network latency and other issues can cause a lag of up to few hundred milliseconds between the sending of a command and it actually being received by all the players. By playing the sound as soon as the mouse is clicked, we give the player the illusion that the command has been executed immediately and make the effect of lag less noticeable.

■ **Note** Some games use animation sequences in addition to sounds to indicate to the player that the unit is processing the command. Games such as first-person shooters often attempt to predict the unit movement and start moving the unit before receiving the server acknowledgment.

If you open and run the game now, you should hear the units acknowledge the command before they start moving or attacking. Next, let's add the message sound.

Messages

We will play a short beeping sound to notify players whenever they are shown a message. We will do this by playing the `message-received` sound from inside the `game` object's `showMessage()` method, as shown in Listing 10-5.

Listing 10-5. Message Notification Sound Inside the showMessage() Method (game.js)

```
showMessage:function(from,message){
    sounds.play('message-received');
    var character = game.characters[from];
    if (character){
        from = character.name;
        if (character.image){
            $('#callerpicture').html('<img src="'+character.image+'"/>');
            // hide the profile picture after six seconds
            setTimeout(function(){
                $('#callerpicture').html("");
            },6000)
        }
    }
    // Append message to messages pane and scroll to the bottom
    var existingMessage = $('#gamemessages').html();
    var newMessage = existingMessage+'<span>'+from+': </span>'+message+'<br>';
    $('#gamemessages').html(newMessage);
    $('#gamemessages').animate( {scrollTop:$('#gamemessages').prop('scrollHeight')});
},
```

If you play the game now, you should hear beeping whenever a new message is displayed.

The last set of sounds we will implement is for combat.

Combat

You may have noticed that we declared four different sound types within our sounds list: `bullet`, `heatseeker`, `cannon-ball`, and `fireball`. These four sounds correspond to the four bullet types that we declared in the previous chapter. Any time we fire a bullet, we will play the sound for the appropriate bullet.

We can easily do this by modifying the `add()` method inside `game.js` to play the appropriate sound whenever a bullet is added, as shown in Listing 10-6.

Listing 10-6. Playing Sound When a Bullet Is Added (game.js)

```
add:function(itemDetails) {
    // Set a unique id for the item
    if (!itemDetails.uid){
        itemDetails.uid = game.counter++;
    }

    var item = window[itemDetails.type].add(itemDetails);

    // Add the item to the items array
    game.items.push(item);
    // Add the item to the type specific array
    game[item.type].push(item);
```

```
    if(item.type == "buildings" || item.type == "terrain"){
        game.currentMapPassableGrid = undefined;
    }

    if (item.type == "bullets"){
        sounds.play(item.name);
    }
    return item;
},
```

If you play the game now, you should hear the distinct sounds of the different weapons as they are being fired.

We could keep adding more sound to our game if we wanted, such as explosions, construction noises, conversation, and even background music. The process would remain the same. However, the sounds we have implemented so far are sufficient for now.

Now that we have sound in our game, it's time to start building the actual levels for our single-player campaign.

Building the Single-Player Campaign

We will build three levels in our game campaign. Each of the levels will get progressively harder, while building upon the story from the previous levels. These levels will illustrate the typical types of levels you would find in an RTS game.

The Rescue

The introductory level in our game will be a relatively easy mission so that the player can get comfortable with moving units around the map and attacking enemy units.

The player will need to navigate across a map populated with easily defeated enemies and then escort a convoy of transport vehicles back to that player's starting location. After the mission briefing, we will move the story line forward using character dialogue that is triggered by timed and conditional triggers.

We will start with a completely fresh `map` object inside `maps.js`, as shown in Listing 10-7.

Listing 10-7. Creating the First Level (maps.js)

```
var maps = {
    "singleplayer":[
        {
            "name":"Rescue",
            "briefing": "In the months since the great war, mankind has fallen into chaos. Billions
are dead with cities in ruins.\nSmall groups of survivors band together to try and survive as best
as they can.\nWe are trying to reach out to all the survivors in this sector before we join back
with the main colony.",

            /* Map Details */
            "mapImage":"images/maps/level-one.png",
            "startX":36,
            "startY":0,

            /* Map coordinates that are obstructed by terrain*/
            "mapGridWidth":60,
            "mapGridHeight":40,
            "mapObstructedTerrain":[
```

```
                [49,8], [50,8], [51,8], [51,9], [52,9], [53,9], [53,10], [53,11], [53,12], [53,13],
[53,14], [53,15], [53,16], [52,16], [52,17], [52,18], [52,19], [51,19], [50,19], [50,18], [50,17],
[49,17], [49,18], [48,18], [47,18], [47,17], [47,16], [48,16], [49,16], [49,15], [49,14], [48,14],
[48,13], [48,12], [49,12], [49,11], [50,11], [50,10], [49,10], [49,9], [44,0], [45,0], [45,1],
[45,2], [46,2], [47,2], [47,3], [48,3], [48,4], [48,5], [49,5], [49,6], [49,7], [50,7], [51,7],
[51,6], [51,5], [51,4], [52,4], [53,4], [53,3], [54,3], [55,3], [55,2], [56,2], [56,1], [56,0],
[55,0], [43,19], [44,19], [45,19], [46,19], [47,19], [48,19], [48,20], [48,21], [47,21], [46,21],
[45,21], [44,21], [43,21], [43,20], [41,22], [42,22], [43,22], [44,22], [45,22], [46,22], [47,22],
[48,22], [49,22], [50,22], [50,23], [50,24], [49,24], [48,24], [47,24], [47,25], [47,26], [47,27],
[47,28], [47,29], [47,30], [46,30], [45,30], [44,30], [43,30], [43,29], [43,28], [43,27], [43,26],
[43,25], [43,24], [42,24], [41,24], [41,23], [48,39], [49,39], [50,39], [51,39], [52,39], [53,39],
[54,39], [55,39], [56,39], [57,39], [58,39], [59,39], [59,38], [59,37], [59,36], [59,35], [59,34],
[59,33], [59,32], [59,31], [59,30], [59,29], [0,0], [1,0], [2,0], [1,1], [2,1], [10,3], [11,3],
[12,3], [12,2], [13,2], [14,2], [14,3], [14,4], [15,4], [15,5], [15,6], [14,6], [13,6], [13,5],
[12,5], [11,5], [10,5], [10,4], [3,9], [4,9], [5,9], [5,10], [6,10], [7,10], [8,10], [9,10], [9,11],
[10,11], [11,11], [11,10], [12,10], [13,10], [13,11], [13,12], [12,12], [11,12], [10,12], [9,12],
[8,12], [7,12], [7,13], [7,14], [6,14], [5,14], [5,13], [5,12], [5,11], [4,11], [3,11], [3,10],
[33,33], [34,33], [35,33], [35,34], [35,35], [34,35], [33,35], [33,34], [27,39], [27,38], [27,37],
[28,37], [28,36], [28,35], [28,34], [28,33], [28,32], [28,31], [28,30], [28,29], [29,29], [29,28],
[29,27], [29,26], [29,25], [29,24], [29,23], [30,23], [31,23], [32,23], [32,22], [32,21], [31,21],
[30,21], [30,22], [29,22], [28,22], [27,22], [26,22], [26,21], [25,21], [24,21], [24,22], [24,23],
[25,23], [26,23], [26,24], [25,24], [25,25], [24,25], [24,26], [24,27], [25,27], [25,28], [25,29],
[24,29], [23,29], [23,30], [23,31], [24,31], [25,31], [25,32], [25,33], [24,33], [23,33], [23,34],
[23,35], [24,35], [24,36], [24,37], [23,37], [22,37], [22,38], [22,39], [23,39], [24,39], [25,39],
[26,0], [26,1], [25,1], [25,2], [25,3], [26,3], [27,3], [27,2], [28,2], [29,2], [29,3], [30,3],
[31,3], [31,2], [31,1], [32,1], [32,0], [33,0], [32,8], [33,8], [34,8], [34,9], [34,10], [33,10],
[32,10], [32,9], [8,29], [9,29], [9,30], [17,32], [18,32], [19,32], [19,33], [18,33], [17,33],
[18,34], [19,34], [3,27], [4,27], [4,26], [3,26], [2,26], [3,25], [4,25], [9,20], [10,20], [11,20],
[11,21], [10,21], [10,19], [19,7], [15,7], [29,12], [30,13], [20,14], [21,14], [34,13], [35,13],
[36,13], [36,14], [35,14], [34,14], [35,15], [36,15], [16,18], [17,18], [18,18], [16,19], [17,19],
[18,19], [17,20], [18,20], [11,19], [58,0], [59,0], [58,1], [59,1], [59,2], [58,3], [59,3], [58,4],
[59,4], [59,5], [58,6], [59,6], [58,7], [59,7], [59,8], [58,9], [59,9], [58,10], [59,10], [59,11],
[52,6], [53,6], [54,6], [52,7], [53,7], [54,7], [53,8], [54,8], [44,17], [46,32], [55,32], [54,28],
[26,34], [34,34], [4,10], [6,11], [6,12], [6,13], [7,11], [8,11], [12,11], [27,0], [27,1], [26,2],
[28,1], [28,0], [29,0], [29,1], [30,2], [30,1], [30,0], [31,0], [33,9], [46,0], [47,0], [48,0],
[49,0], [50,0], [51,0], [52,0], [53,0], [54,0], [55,1], [54,1], [53,1], [52,1], [51,1], [50,1],
[49,1], [48,1], [47,1], [46,1], [48,2], [49,2], [50,2], [51,2], [52,2], [53,2], [54,2], [52,3],
[51,3], [50,3], [49,3], [49,4], [50,4], [50,5], [50,6], [50,9], [51,10], [52,10], [51,11], [52,11],
[50,12], [51,12], [52,12], [49,13], [50,13], [51,13], [52,13], [50,14], [51,14], [52,14], [50,15],
[51,15], [52,15], [50,16], [51,16], [51,17], [48,17], [51,18], [44,20], [45,20], [46,20], [47,20],
[42,23], [43,23], [44,23], [45,23], [46,23], [47,23], [48,23], [49,23], [44,24], [45,24], [46,24],
[44,25], [45,25], [46,25], [44,26], [45,26], [46,26], [44,27], [45,27], [46,27], [44,28], [45,28],
[46,28], [44,29], [45,29], [46,29], [11,4], [12,4], [13,4], [13,3], [14,5], [25,22], [31,22],
[27,23], [28,23], [27,24], [28,24], [26,25], [27,25], [28,25], [25,26], [26,26], [27,26], [28,26],
[26,27], [27,27], [28,27], [26,28], [27,28], [28,28], [26,29], [27,29], [24,30], [25,30], [26,30],
[27,30], [26,31], [27,31], [26,32], [27,32], [26,33], [27,33], [24,34], [25,34], [27,34], [25,35],
[26,35], [27,35], [25,36], [26,36], [27,36], [25,37], [26,37], [23,38], [24,38], [25,38], [26,38],
[26,39], [2,25], [9,19], [36,31]
            ],
```

```
            /* Entities to be loaded */
            "requirements":{
                "buildings":["base"],
                "vehicles":["transport","scout-tank","heavy-tank"],
                "aircraft":[],
                "terrain":[]
            },

            /* Economy Related*/
            "cash":{
                "blue":0,
                "green":0
            },

            /* Entities to be added */
            "items":[
                /* Slightly damaged base */
                {"type":"buildings","name":"base","x":55,"y":6,"team":"blue","life":100},

                {"type":"vehicles","name":"heavy-tank","x":57,"y":12,"direction":4,"team":
"blue","uid":-1},

                /* Two transport vehicles waiting just outside the visible map */
                {"type":"vehicles","name":"transport","x":-3,"y":2,"direction":2,"team":"blue",
"uid":-3,"selectable":false},
                {"type":"vehicles","name":"transport","x":-3,"y":4,"direction":2,"team":"blue",
"uid":-4,"selectable":false},

                /* Two damaged enemy scout-tanks patroling the area*/
                {"type":"vehicles","name":"scout-tank","x":40,"y":20,"direction":4,"team":"green",
"uid":-2,"life":20,"orders":{"type":"patrol","from":{"x":34,"y":20},"to":{"x":42,"y":25}}},
                {"type":"vehicles","name":"scout-tank","x":14,"y":0,"direction":4,"team":"green",
"uid":-5,"life":20,"orders":{"type":"patrol","from":{"x":14,"y":0},"to":{"x":14,"y":14}}},

            ],

            /* Conditional and Timed Trigger Events */
            "triggers":[
                {"type":"timed","time":3000,
                    "action":function(){
                        game.showMessage("op", "Commander!! We haven't heard from the last convoy in
over two hours. They should have arrived by now.");
                    }
                },
                {"type":"timed","time":10000,
                    "action":function(){
                        game.showMessage("op", "They were last seen in the North West Sector.
Could you investigate?");
                    }
                },
                {"type":"conditional",
                    "condition":function(){
```

```
                        return(isItemDead(-1)||isItemDead(-3)||isItemDead(-4));
                    },
                    "action":function(){
                        singleplayer.endLevel(false);
                    }
                },
            ],
        }
    ]
}
```

The first portion of the level consists of the same basic metadata that we saw in earlier levels. We start with the mission briefing, which gives the player a little background on the map. We then set the map image to the final version of the map instead of the debug version we have been using to build the game so far. We also set the map starting position to the top-right corner of the map. Finally, we set the map size and the `mapObstructedTerrain` properties.

Next we load a few essential items in the `requirements` array and set the starting cash balance for both players to 0.

Within the level's `items` array, we add a damaged base, a heavy-tank that the player will control, two enemy scout-tanks that patrol the area, and two transports. We set UIDs for each of them so that we can refer to them from the triggers.

Since this is the first level, we set the life of the scout-tanks so the player will find it easy to destroy them. The transports are positioned slightly outside the bounds of the top-left corner of the map so that they do not become visible to the player until the right time.

Within the `triggers` array, we define our first few triggers. The first two timed triggers show players a message from the operator, asking them to find the missing transport. The third is a conditional trigger that will end the mission as a failure if either the transports or the heavy tank get destroyed by using the `isItemDead()` method.

Next, we will add a few new characters to the `characters` object inside the `game` object, as shown in Listing 10-8.

Listing 10-8. Adding New Characters (game.js)

```
characters: {
    "system":{
        "name":"System",
        "image":"images/characters/system.png"
    },
    "op":{
        "name":"Operator",
        "image":"images/characters/girl1.png"
    },
    "pilot":{
        "name":"Pilot",
        "image":"images/characters/girl2.png"
    },
    "driver":{
        "name":"Driver",
        "image":"images/characters/man1.png"
    }
},
```

■ **Note** These new character images are Creative Commons–licensed artwork found at http://opengameart.org.

We will also define the `isItemDead()` method inside `common.js`, as shown in Listing 10-9.

Listing 10-9. The isItemDead() Method (common.js)

```
function isItemDead(uid){
    var item = game.getItemByUid(uid);
    return (!item || item.lifeCode == "dead");
}
```

We consider an item dead if we can no longer find it in the `game.items` array or if its `lifeCode` property is set to dead.

If you run the game so far, you should see the operator giving you your first mission task by asking you to investigate the situation, as shown in Figure 10-1.

Figure 10-1. *The first mission task*

You should be able to select the tank and move it around, with the fog of war slowly clearing up as you explore the map.

Now, we will introduce the enemy and the convoy by adding a few more triggers to the first map, as shown in Listing 10-10.

Listing 10-10. Introducing the Enemy and the Convoy (maps.js)

```
{"type":"conditional",
    "condition":function(){
        // Check if first enemy is dead
```

```
        return isItemDead(-2);
    },
    "action":function(){
        game.showMessage("op", "The rebels have been getting very aggressive lately. I hope the
convoy is safe. Find them and escort them back to the base.");
    }
},
{"type":"conditional",
    "condition":function(){
        var hero = game.getItemByUid(-1);
        return(hero && hero.x<30 && hero.y<30);
    },
    "action":function(){
        game.showMessage("driver", "Can anyone hear us? Our convoy has been pinned down by rebel
tanks. We need help.");
    }
},
```

In the first conditional trigger, we show a message from the operator discussing the rebels once the first enemy scout tank is destroyed. In the second conditional trigger, we show a message from the convoy driver once we enter the top-left corner of the map.

If we run the game now, we should see the operator urging us to hurry after the first fight with the rebels and the convoy driver calling for help when we approach the convoy location, as shown in Figure 10-2.

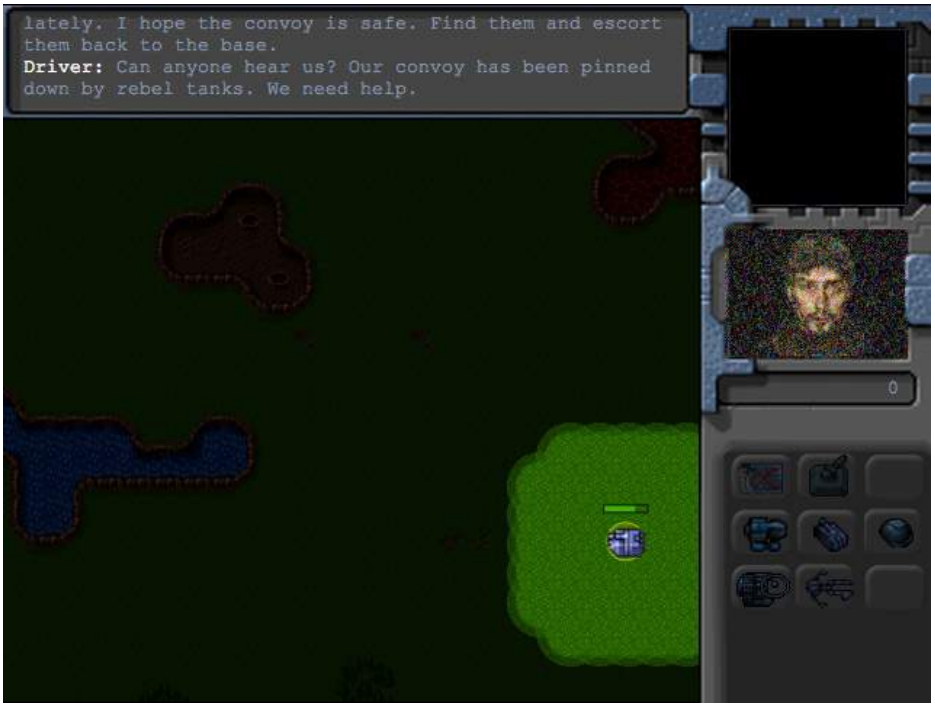

Figure 10-2. *Convoy driver asking for help*

Finally, we will add a few triggers to implement rescuing the convoy and completing the mission, as shown in Listing 10-11.

Listing 10-11. Rescuing the Convoy and Completing the Mission (maps.js)

```
{"type":"conditional",
    "condition":function(){
        var hero = game.getItemByUid(-1);
        return(hero && hero.x<10 && hero.y<10);
    },
    "action":function(){
        var hero = game.getItemByUid(-1);
        game.showMessage("driver", "Thank you. We thought we would never get out of here alive.");
        game.sendCommand([-3,-4],{type:"guard",to:hero});
    }
},
{"type":"conditional",
    "condition":function(){
        var transport1 = game.getItemByUid(-3);
        var transport2 = game.getItemByUid(-4);
        return(transport1 && transport2 && transport1.x>52 && transport2.x>52 && transport2.y<18
&& transport1.y<18);
    },
    "action":function(){
        singleplayer.endLevel(true);
    }
},
```

In the first conditional trigger, we show another message from the driver when the hero tank reaches the top-left corner of the map. We then command both the transports to guard the tank, which means they will follow the tank wherever it goes.

In the second conditional trigger, we end the game once both the transports reach the top-right corner of the map where the base is located.

If you run the game now, you should see the convoy driver thanking you for saving the convoy and then following you back to the base, as shown in Figure 10-3.

Figure 10-3. *Rescuing the convoy*

The return journey should be uneventful since all the enemies in this level are dead. Once the two transports reach the base, the mission will end. You have just completed your first mission in the single-player campaign.

Now it's time to make the next level, Assault.

Assault

The second level in our game will be a little more challenging than the first one. This time we will introduce the player to the idea of micromanaging units to attack the enemy, without having to worry about managing resources or the production of units.

Players will be provided a steady stream of reinforcements that they will need to use to locate and capture a small enemy base across the map. The enemy base will keep sending out steady waves of attacking units to make the mission more challenging.

We will create a new `level` object inside the `singleplayer` array of the `map` object, as shown in Listing 10-12. This new level will automatically be loaded once the first mission has been completed.

Listing 10-12. Creating the Second Level (maps.js)

```
{
    "name":"Assault",
    "briefing": "Thanks to the supplies from the convoy, we now have the base up and running.\n
The rebels nearby are proving to be a problem. We need to take them out. \n First set up the
base defences. Then find and destroy all rebels in the area.\n The colony will be sending us
reinforcements to help us out.",
```

```
    /* Map Details */
    "mapImage":"images/maps/level-one.png",
    "startX":36,
    "startY":0,

    /* Map coordinates that are obstructed by terrain*/
    "mapGridWidth":60,
    "mapGridHeight":40,
    "mapObstructedTerrain":[
        [49,8], [50,8], [51,8], [51,9], [52,9], [53,9], [53,10], [53,11], [53,12], [53,13],
[53,14], [53,15], [53,16], [52,16], [52,17], [52,18], [52,19], [51,19], [50,19], [50,18], [50,17],
[49,17], [49,18], [48,18], [47,18], [47,17], [47,16], [48,16], [49,16], [49,15], [49,14], [48,14],
[48,13], [48,12], [49,12], [49,11], [50,11], [50,10], [49,10], [49,9], [44,0], [45,0], [45,1],
[45,2], [46,2], [47,2], [47,3], [48,3], [48,4], [48,5], [49,5], [49,6], [49,7], [50,7], [51,7],
[51,6], [51,5], [51,4], [52,4], [53,4], [53,3], [54,3], [55,3], [55,2], [56,2], [56,1], [56,0],
[55,0], [43,19], [44,19], [45,19], [46,19], [47,19], [48,19], [48,20], [48,21], [47,21], [46,21],
[45,21], [44,21], [43,21], [43,20], [41,22], [42,22], [43,22], [44,22], [45,22], [46,22], [47,22],
[48,22], [49,22], [50,22], [50,23], [50,24], [49,24], [48,24], [47,24], [47,25], [47,26], [47,27],
[47,28], [47,29], [47,30], [46,30], [45,30], [44,30], [43,30], [43,29], [43,28], [43,27], [43,26],
[43,25], [43,24], [42,24], [41,24], [41,23], [48,39], [49,39], [50,39], [51,39], [52,39], [53,39],
[54,39], [55,39], [56,39], [57,39], [58,39], [59,39], [59,38], [59,37], [59,36], [59,35], [59,34],
[59,33], [59,32], [59,31], [59,30], [59,29], [0,0], [1,0], [2,0], [1,1], [2,1], [10,3], [11,3],
[12,3], [12,2], [13,2], [14,2], [14,3], [14,4], [15,4], [15,5], [15,6], [14,6], [13,6], [13,5],
[12,5], [11,5], [10,5], [10,4], [3,9], [4,9], [5,9], [5,10], [6,10], [7,10], [8,10], [9,10], [9,11],
[10,11], [11,11], [11,10], [12,10], [13,10], [13,11], [13,12], [12,12], [11,12], [10,12], [9,12],
[8,12], [7,12], [7,13], [7,14], [6,14], [5,14], [5,13], [5,12], [5,11], [4,11], [3,11], [3,10],
[33,33], [34,33], [35,33], [35,34], [35,35], [34,35], [33,35], [33,34], [27,39], [27,38], [27,37],
[28,37], [28,36], [28,35], [28,34], [28,33], [28,32], [28,31], [28,30], [28,29], [29,29], [29,28],
[29,27], [29,26], [29,25], [29,24], [29,23], [30,23], [31,23], [32,23], [32,22], [32,21], [31,21],
[30,21], [30,22], [29,22], [28,22], [27,22], [26,22], [26,21], [25,21], [24,21], [24,22], [24,23],
[25,23], [26,23], [26,24], [25,24], [25,25], [24,25], [24,26], [24,27], [25,27], [25,28], [25,29],
[24,29], [23,29], [23,30], [23,31], [24,31], [25,31], [25,32], [25,33], [24,33], [23,33], [23,34],
[23,35], [24,35], [24,36], [24,37], [23,37], [22,37], [22,38], [22,39], [23,39], [24,39], [25,39],
[26,0], [26,1], [25,1], [25,2], [25,3], [26,3], [27,3], [27,2], [28,2], [29,2], [29,3], [30,3],
[31,3], [31,2], [31,1], [32,1], [32,0], [33,0], [32,8], [33,8], [34,8], [34,9], [34,10], [33,10],
[32,10], [32,9], [8,29], [9,29], [9,30], [17,32], [18,32], [19,32], [19,33], [18,33], [17,33],
[18,34], [19,34], [3,27], [4,27], [4,26], [3,26], [2,26], [3,25], [4,25], [9,20], [10,20], [11,20],
[11,21], [10,21], [10,19], [19,7], [15,7], [29,12], [30,13], [20,14], [21,14], [34,13], [35,13],
[36,13], [36,14], [35,14], [34,14], [35,15], [36,15], [16,18], [17,18], [18,18], [16,19], [17,19],
[18,19], [17,20], [18,20], [11,19], [58,0], [59,0], [58,1], [59,1], [59,2], [58,3], [59,3], [58,4],
[59,4], [59,5], [58,6], [59,6], [58,7], [59,7], [59,8], [58,9], [59,9], [58,10], [59,10], [59,11],
[52,6], [53,6], [54,6], [52,7], [53,7], [54,7], [53,8], [54,8], [44,17], [46,32], [55,32], [54,28],
[26,34], [34,34], [4,10], [6,11], [6,12], [6,13], [7,11], [8,11], [12,11], [27,0], [27,1], [26,2],
[28,1], [28,0], [29,0], [29,1], [30,2], [30,1], [30,0], [31,0], [33,9], [46,0], [47,0], [48,0],
[49,0], [50,0], [51,0], [52,0], [53,0], [54,0], [55,1], [54,1], [53,1], [52,1], [51,1], [50,1],
[49,1], [48,1], [47,1], [46,1], [48,2], [49,2], [50,2], [51,2], [52,2], [53,2], [54,2], [52,3],
[51,3], [50,3], [49,3], [49,4], [50,4], [50,5], [50,6], [50,9], [51,10], [52,10], [51,11], [52,11],
[50,12], [51,12], [52,12], [49,13], [50,13], [51,13], [52,13], [50,14], [51,14], [52,14], [50,15],
[51,15], [52,15], [50,16], [51,16], [51,17], [48,17], [51,18], [44,20], [45,20], [46,20], [47,20],
[42,23], [43,23], [44,23], [45,23], [46,23], [47,23], [48,23], [49,23], [44,24], [45,24], [46,24],
[44,25], [45,25], [46,25], [44,26], [45,26], [46,26], [44,27], [45,27], [46,27], [44,28], [45,28],
```

```
[46,28], [44,29], [45,29], [46,29], [11,4], [12,4], [13,4], [13,3], [14,5], [25,22], [31,22],
[27,23], [28,23], [27,24], [28,24], [26,25], [27,25], [28,25], [25,26], [26,26], [27,26], [28,26],
[26,27], [27,27], [28,27], [26,28], [27,28], [28,28], [26,29], [27,29], [24,30], [25,30], [26,30],
[27,30], [26,31], [27,31], [26,32], [27,32], [26,33], [27,33], [24,34], [25,34], [27,34], [25,35],
[26,35], [27,35], [25,36], [26,36], [27,36], [25,37], [26,37], [23,38], [24,38], [25,38], [26,38],
[26,39], [2,25], [9,19], [36,31]
    ],

    /* Entities to be loaded */
    "requirements":{
        "buildings":["base","ground-turret","starport","harvester"],
        "vehicles":["transport","scout-tank","heavy-tank"],
        "aircraft":["chopper"],
        "terrain":[]
    },

    /* Economy Related*/
    "cash":{
        "blue":0,
        "green":0
    },

    /* Entities to be added */
    "items":[
        {"type":"buildings","name":"base","x":55,"y":6,"team":"blue","uid":-1},
        {"type":"buildings","name":"ground-turret","x":53,"y":17,"team":"blue"},
        {"type":"vehicles","name":"heavy-tank","x":55,"y":16,"direction":4,"team":"blue",
"uid":-2,"orders":{"type":"sentry"}},

        /* The first wave of attacks*/
        {"type":"vehicles","name":"scout-tank","x":55,"y":36,"direction":4,"team":"green","orders":
{"type":"hunt"}},
        {"type":"vehicles","name":"scout-tank","x":53,"y":36,"direction":4,"team":"green","orders":
{"type":"hunt"}},

        /* Enemies patroling the area */
        {"type":"vehicles","name":"scout-tank","x":5,"y":5,"direction":4,"team":"green","orders":
{"type":"patrol","from":{"x":5,"y":5},"to":{"x":20,"y":20}}},
        {"type":"vehicles","name":"scout-tank","x":5,"y":15,"direction":4,"team":"green","orders":
{"type":"patrol","from":{"x":5,"y":15},"to":{"x":20,"y":30}}},
        {"type":"vehicles","name":"scout-tank","x":25,"y":5,"direction":4,"team":"green","orders":
{"type":"patrol","from":{"x":25,"y":5},"to":{"x":25,"y":20}}},
        {"type":"vehicles","name":"scout-tank","x":35,"y":5,"direction":4,"team":"green","orders":
{"type":"patrol","from":{"x":35,"y":5},"to":{"x":35,"y":30}}},

        /* The Evil Rebel Base*/
        {"type":"buildings","name":"base","x":5,"y":36,"team":"green","uid":-11},
        {"type":"buildings","name":"starport","x":1,"y":30,"team":"green","uid":-12},
        {"type":"buildings","name":"starport","x":4,"y":32,"team":"green","uid":-13},
        {"type":"buildings","name":"harvester","x":1,"y":38,"team":"green","action":"deploy"},
        {"type":"buildings","name":"ground-turret","x":5,"y":28, "team":"green"},
```

```
        {"type":"buildings","name":"ground-turret","x":7,"y":33, "team":"green"},
        {"type":"buildings","name":"ground-turret","x":8,"y":37, "team":"green"},
    ],

    /* Conditional and Timed Trigger Events */
    "triggers":[
        {"type":"timed","time":8000,
            "action":function(){
                // Send in reinforcements to defend the base from the first wave
                game.showMessage("op", "Commander!! Reinforcements have arrived from the colony.");
                var hero = game.getItemByUid(-2);
                game.add ({"type":"vehicles","name":"scout-tank","x":61,"y":22, "team":"blue",
"orders":{"type":"guard","to":hero}});
                game.add ({"type":"vehicles","name":"scout-tank","x":61,"y":21, "team":"blue",
"orders":{"type":"guard","to":hero}});
            }
        },
        {"type":"timed","time":25000,
            "action":function(){
                // Supply extra cash
                game.cash["blue"] = 1500;
                game.showMessage("op", "Commander!! We have enough resources for another ground
turret. Set up the turret to keep the base safe from any more attacks.");
            }
        },
    ]
},
```

The first portion of the level has nearly the same metadata as the previous level. We are reusing the map from level 1. The only thing that changes is the mission briefing.

Next, we load all the essential items in the `requirements` array, and set the starting cash balance for both players to 0.

This time, we add a lot more items in the level's `items` array. We start with the base, a heavy tank that the player will control, and a ground turret to protect the base.

Next, we add two enemy scout tanks that are set to `hunt` mode so they will attack our base as soon as the game starts. We then add several more scout tanks that are set to patrol all around the map.

Finally, we add an enemy base that has two starports and a refinery. It is also well defended with several ground turrets and scout tanks patrolling nearby.

The `triggers` array contains two timed triggers that are set off within the first few seconds of the game. Within the first trigger, we add two friendly scout tanks to the game and notify the player that reinforcements have arrived.

In the second trigger, we give players 1,500 credits and tell them they have enough resources to build one ground turret. Placing this turret will be the only sidebar-related task players will perform in this game.

If you run the game, you will find that the second level starts in a much more exciting way than the first level. You will see that the base is under attack within the first few seconds and reinforcements arrive in the nick of time to save you, as shown in Figure 10-4.

Figure 10-4. *Saved by reinforcements*

Once the attack has been stopped, the operator will notify you that you have enough resources to build one more turret.

Now we will add a few more triggers to add waves of attacking units and reinforcements, as shown in Listing 10-13.

Listing 10-13. Adding Enemy Waves and Reinforcements (maps.js)

```
// Construct a couple of bad guys to hunt the player every time enemy has enough money
{"type":"timed","time":60000,"repeat":true,
    "action":function(){
        if(game.cash["green"]>1000){
            game.sendCommand([-12,-13],{type:"construct-unit", details:{type:"vehicles",
name:"scout-tank", oders:{"type":"hunt"}}});
        }
    }
},
// Send in some reinforcements every three minutes
{"type":"timed","time":180000,"repeat":true,
    "action":function(){
        game.showMessage("op", "Commander!! More Reinforcments have arrived.");
        game.add ({"type":"vehicles","name":"scout-tank","x":61,"y":22, "team":"blue","orders":
{"type":"move","to":{"x":55,"y":21}}});
        game.add ({"type":"vehicles","name":"heavy-tank","x":61,"y":23, "team":"blue","orders":
{"type":"move","to":{"x":56,"y":23}}});
    }
},
```

In the first timed trigger, we check whether the green player has enough money every 60 seconds, and if the green player does, we construct a couple of scout tanks in hunt mode. In the second timed trigger, we send the hero two reinforcing units every 180 seconds.

Unlike the first level, the enemy has a lot more units and turret defenses. A direct frontal assault on the enemy base will not work because the player is likely to lose all the units. The player also cannot wait too long since the enemy will keep sending out waves of enemies every few minutes.

A player's best strategy will be to make small attacks, chipping away at the opposition and then falling back to the base for reinforcements until ready to make the final attack.

Finally, we will add triggers to provide the player with air support and to complete the mission, as shown in Listing 10-14.

Listing 10-14. Adding Air Support and Completing the Mission(maps.js)

```
// Send in air support after 10 minutes
{"type":"timed","time":600000,
    "action":function(){
        game.showMessage("pilot", "Close Air Support en route. Will try to do what I can.");
        game.add ({"type":"aircraft","name":"chopper","x":61, "y":22,"selectable":false,
"team":"blue", "orders":{"type":"hunt"}});
    }
},
/* Lose if our base gets destroyed  */
{"type":"conditional",
    "condition":function(){
        return isItemDead(-1);
    },
    "action":function(){
        singleplayer.endLevel(false);
    }
},
/* Win if enemy base gets at least half destroyed */
{"type":"conditional",
    "condition":function(){
        var enemyBase = game.getItemByUid(-11);
        return(!enemyBase || (enemyBase.life<=enemyBase.hitPoints/2));
    },
    "action":function(){
        singleplayer.endLevel(true);
    }
},
```

In the first timed trigger, we release a friendly chopper on `hunt` mode, ten minutes after the game starts. The next two conditional triggers set the conditions for successfully completing or failing the mission.

If we run the game now, we should see the chopper pilot coming in to give us a helping hand after some time, as shown in Figure 10-5.

Figure 10-5. *Pilot flying in chopper for air support*

If you have trouble completing the mission, the extra air support should help. Again, we introduce a new character, the pilot, who will stay with us for the next mission. With the assistance of the chopper, we can capture the enemy base and complete the mission so we can go on to the final mission.

Now it's time to build our final mission: Under Siege.

Under Siege

The final level in our game will be the most challenging. The player will need to constantly build units to fight off several waves of enemy units.

This time, the player will take over the enemy base captured after the last mission. The player will be provided some initial supplies to help get started. After that, the player will need to fend off several waves of units and protect the transport vehicles filled with refugees until the colony reinforcements arrive to help them.

We will create a new level object inside the `singleplayer` array of the `map` object, as shown in Listing 10-15. This new level will automatically be loaded once the second mission has been completed.

Listing 10-15. Creating the Third Level (maps.js)

```
{
    "name":"Under Siege",
    "briefing": "Thanks to the attack led by you, we now have control of the rebel base. We can
expect the rebels to try to retaliate.\n The colony is sending in aircraft to help us evacuate back
to the main camp. All we need to do is hang tight until the choppers get here. \n Luckily, we have
```

```
some supplies and ammunition to defend ourselves with until they get here. \n Protect the transports
at all costs.",

    /* Map Details */
    "mapImage":"images/maps/level-one.png",
    "startX":0,
    "startY":20,

    /* Map coordinates that are obstructed by terrain*/
    "mapGridWidth":60,
    "mapGridHeight":40,
    "mapObstructedTerrain":[
        [49,8], [50,8], [51,8], [51,9], [52,9], [53,9], [53,10], [53,11], [53,12], [53,13],
[53,14], [53,15], [53,16], [52,16], [52,17], [52,18], [52,19], [51,19], [50,19], [50,18], [50,17],
[49,17], [49,18], [48,18], [47,18], [47,17], [47,16], [48,16], [49,16], [49,15], [49,14], [48,14],
[48,13], [48,12], [49,12], [49,11], [50,11], [50,10], [49,10], [49,9], [44,0], [45,0], [45,1],
[45,2], [46,2], [47,2], [47,3], [48,3], [48,4], [48,5], [49,5], [49,6], [49,7], [50,7], [51,7],
[51,6], [51,5], [51,4], [52,4], [53,4], [53,3], [54,3], [55,3], [55,2], [56,2], [56,1], [56,0],
[55,0], [43,19], [44,19], [45,19], [46,19], [47,19], [48,19], [48,20], [48,21], [47,21], [46,21],
[45,21], [44,21], [43,21], [43,20], [41,22], [42,22], [43,22], [44,22], [45,22], [46,22], [47,22],
[48,22], [49,22], [50,22], [50,23], [50,24], [49,24], [48,24], [47,24], [47,25], [47,26], [47,27],
[47,28], [47,29], [47,30], [46,30], [45,30], [44,30], [43,30], [43,29], [43,28], [43,27], [43,26],
[43,25], [43,24], [42,24], [41,24], [41,23], [48,39], [49,39], [50,39], [51,39], [52,39], [53,39],
[54,39], [55,39], [56,39], [57,39], [58,39], [59,39], [59,38], [59,37], [59,36], [59,35], [59,34],
[59,33], [59,32], [59,31], [59,30], [59,29], [0,0], [1,0], [2,0], [1,1], [2,1], [10,3], [11,3],
[12,3], [12,2], [13,2], [14,2], [14,3], [14,4], [15,4], [15,5], [15,6], [14,6], [13,6], [13,5],
[12,5], [11,5], [10,5], [10,4], [3,9], [4,9], [5,9], [5,10], [6,10], [7,10], [8,10], [9,10], [9,11],
[10,11], [11,11], [11,10], [12,10], [13,10], [13,11], [13,12], [12,12], [11,12], [10,12], [9,12],
[8,12], [7,12], [7,13], [7,14], [6,14], [5,14], [5,13], [5,12], [5,11], [4,11], [3,11], [3,10],
[33,33], [34,33], [35,33], [35,34], [35,35], [34,35], [33,35], [33,34], [27,39], [27,38], [27,37],
[28,37], [28,36], [28,35], [28,34], [28,33], [28,32], [28,31], [28,30], [28,29], [29,29], [29,28],
[29,27], [29,26], [29,25], [29,24], [29,23], [30,23], [31,23], [32,23], [32,22], [32,21], [31,21],
[30,21], [30,22], [29,22], [28,22], [27,22], [26,22], [26,21], [25,21], [24,21], [24,22], [24,23],
[25,23], [26,23], [26,24], [25,24], [25,25], [24,25], [24,26], [24,27], [25,27], [25,28], [25,29],
[24,29], [23,29], [23,30], [23,31], [24,31], [25,31], [25,32], [25,33], [24,33], [23,33], [23,34],
[23,35], [24,35], [24,36], [24,37], [23,37], [22,37], [22,38], [22,39], [23,39], [24,39], [25,39],
[26,0], [26,1], [25,1], [25,2], [25,3], [26,3], [27,3], [27,2], [28,2], [29,2], [29,3], [30,3],
[31,3], [31,2], [31,1], [32,1], [32,0], [33,0], [32,8], [33,8], [34,8], [34,9], [34,10], [33,10],
[32,10], [32,9], [8,29], [9,29], [9,30], [17,32], [18,32], [19,32], [19,33], [18,33], [17,33],
[18,34], [19,34], [3,27], [4,27], [4,26], [3,26], [2,26], [3,25], [4,25], [9,20], [10,20], [11,20],
[11,21], [10,21], [10,19], [19,7], [15,7], [29,12], [30,13], [20,14], [21,14], [34,13], [35,13],
[36,13], [36,14], [35,14], [34,14], [35,15], [36,15], [16,18], [17,18], [18,18], [16,19], [17,19],
[18,19], [17,20], [18,20], [11,19], [58,0], [59,0], [58,1], [59,1], [59,2], [58,3], [59,3], [58,4],
[59,4], [59,5], [58,6], [59,6], [58,7], [59,7], [59,8], [58,9], [59,9], [58,10], [59,10], [59,11],
[52,6], [53,6], [54,6], [52,7], [53,7], [54,7], [53,8], [54,8], [44,17], [46,32], [55,32], [54,28],
[26,34], [34,34], [4,10], [6,11], [6,12], [6,13], [7,11], [8,11], [12,11], [27,0], [27,1], [26,2],
[28,1], [28,0], [29,0], [29,1], [30,2], [30,1], [30,0], [31,0], [33,9], [46,0], [47,0], [48,0],
[49,0], [50,0], [51,0], [52,0], [53,0], [54,0], [55,1], [54,1], [53,1], [52,1], [51,1], [50,1],
[49,1], [48,1], [47,1], [46,1], [48,2], [49,2], [50,2], [51,2], [52,2], [53,2], [54,2], [52,3],
[51,3], [50,3], [49,3], [49,4], [50,4], [50,5], [50,6], [50,9], [51,10], [52,10], [51,11], [52,11],
[50,12], [51,12], [52,12], [49,13], [50,13], [51,13], [52,13], [50,14], [51,14], [52,14], [50,15],
```

```
[51,15], [52,15], [50,16], [51,16], [51,17], [48,17], [51,18], [44,20], [45,20], [46,20], [47,20],
[42,23], [43,23], [44,23], [45,23], [46,23], [47,23], [48,23], [49,23], [44,24], [45,24], [46,24],
[44,25], [45,25], [46,25], [44,26], [45,26], [46,26], [44,27], [45,27], [46,27], [44,28], [45,28],
[46,28], [44,29], [45,29], [46,29], [11,4], [12,4], [13,4], [13,3], [14,5], [25,22], [31,22],
[27,23], [28,23], [27,24], [28,24], [26,25], [27,25], [28,25], [25,26], [26,26], [27,26], [28,26],
[26,27], [27,27], [28,27], [26,28], [27,28], [28,28], [26,29], [27,29], [24,30], [25,30], [26,30],
[27,30], [26,31], [27,31], [26,32], [27,32], [26,33], [27,33], [24,34], [25,34], [27,34], [25,35],
[26,35], [27,35], [25,36], [26,36], [27,36], [25,37], [26,37], [23,38], [24,38], [25,38], [26,38],
[26,39], [2,25], [9,19], [36,31]
    ],

    /* Entities to be loaded */
    "requirements":{
        "buildings":["base","ground-turret","starport","harvester"],
        "vehicles":["transport","scout-tank","heavy-tank"],
        "aircraft":["chopper","wraith"],
        "terrain":[]
    },

    /* Economy Related*/
    "cash":{
        "blue":0,
        "green":0
    },

    /* Entities to be added */
    "items":[
        /* The Rebel Base now in our hands */
        {"type":"buildings","name":"base","x":5,"y":36,"team":"blue","uid":-11},
        {"type":"buildings","name":"starport","x":1,"y":28,"team":"blue","uid":-12},
        {"type":"buildings","name":"starport","x":4,"y":32,"team":"blue","uid":-13},
        {"type":"buildings","name":"harvester","x":1,"y":38,"team":"blue","action":"deploy"},
        {"type":"buildings","name":"ground-turret","x":7,"y":28,"team":"blue"},
        {"type":"buildings","name":"ground-turret","x":8,"y":32,"team":"blue"},
        {"type":"buildings","name":"ground-turret","x":11,"y":37,"team":"blue"},

        /* The transports that need to be protected*/
        {"type":"vehicles","name":"transport","x":2,"y":33,"team":"blue","direction":2,
"selectable":false, "uid":-1},
        {"type":"vehicles","name":"transport","x":1,"y":34,"team":"blue","direction":2,
"selectable":false,"uid":-2},
        {"type":"vehicles","name":"transport","x":2,"y":35,"team":"blue","direction":2,
"selectable":false,"uid":-3},
        {"type":"vehicles","name":"transport","x":1,"y":36,"team":"blue","direction":2,
"selectable":false,"uid":-4},

        /* The chopper pilot from the last mission */

{"type":"aircraft","name":"chopper","x":15,"y":40,"team":"blue","selectable":false,
"uid":-5,"orders":{"type":"patrol","from":{"x":15,"y":40},"to":{"x":0,"y":25}}},
```

```
        /* The first wave of attacks*/
        {"type":"vehicles","name":"scout-tank","x":15,"y":16,"direction":4,"team":"green",
"orders":{"type":"hunt"}},
        {"type":"vehicles","name":"scout-tank","x":17,"y":16,"direction":4,"team":"green",
"orders":{"type":"hunt"}},

        /* Secret Rebel bases*/

        {"type":"buildings","name":"starport","x":35,"y":37,"team":"green","uid":-23},
        {"type":"buildings","name":"starport","x":33,"y":37,"team":"green","uid":-24},
        {"type":"buildings","name":"harvester","x":28,"y":39,"team":"green","action":"deploy"},
        {"type":"buildings","name":"harvester","x":30,"y":39,"team":"green","action":"deploy"},

        {"type":"buildings","name":"starport","x":3,"y":0,"team":"green","uid":-21},
        {"type":"buildings","name":"starport","x":6,"y":0,"team":"green","uid":-22},
        {"type":"buildings","name":"harvester","x":0,"y":2,"team":"green","action":"deploy"},
        {"type":"buildings","name":"harvester","x":0,"y":4,"team":"green","action":"deploy"},

    ],

    /* Conditional and Timed Trigger Events */
    "triggers":[
        /* Lose if even one transport gets destroyed  */
        {"type":"conditional",
            "condition":function(){
                return isItemDead(-1)||isItemDead(-2)||isItemDead(-3)||isItemDead(-4);
            },
            "action":function(){
                singleplayer.endLevel(false);
            }
        },
        {"type":"timed","time":5000,
            "action":function(){
                game.showMessage("op", "Commander!! The rebels have started attacking. We need to
protect the base at any cost.");
            }
        },
    ],
}
```

Again, we are reusing the map from level 1. The first portion of the level has nearly the same metadata as the previous levels. The only thing that changes is the mission briefing.

Next we load all the essential items in the `requirements` array and set the starting cash balance for both players to 0.

This time, we add a lot more items to the map. First, we re-create the entire enemy base, but for the player team. Next, we add several transport vehicles that we will be protecting in this mission and add the chopper from the last mission on patrol mode to protect the base. Next, we add a few enemy units for a first wave of attacks. Finally, we define several starports and refineries for two secret enemy bases that will be attacking the player.

Within the triggers, we define one conditional trigger that ends the mission in case even one of the transports dies. The second timed trigger just displays a message from the operator.

If we run the game and start the third level, we should see the rebels attacking the base and the patrolling chopper fending them off, as shown in Figure 10-6.

Figure 10-6. *Patrolling chopper defending the base*

Now that the first wave of attacks has been fended off, we will build a little drama with a small cinematic story line within the mission by adding a few creative triggers to the map, as shown in Listing 10-16.

Listing 10-16. Adding a Little Drama to the Level (maps.js)

```
{"type":"timed","time":20000,
    "action":function(){
        game.add({"type":"vehicles","name":"transport","x":57,"y":3,"team":"blue","direction":4,
"selectable":false,"uid":-6});
        game.sendCommand([-5],{"type":"guard","toUid":-6})
        game.showMessage("driver", "Commander!! The colony has sent some extra supplies. We are
coming in from the North East sector through rebel territory. We could use a little protection.");
    }
},
//Have the pilot offer to assist and get some villains in to make it interesting
{"type":"timed","time":28000,
    "action":function(){
        game.showMessage("pilot", "I'm on my way.");
        game.add({"type":"vehicles","name":"scout-tank","x":57,"y":28,"team":"green","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"wraith","x":55,"y":33,"team":"green",
"orders":{"type":"sentry"}});
        game.add({"type":"aircraft","name":"wraith","x":53,"y":33,"team":"green",
"orders":{"type":"sentry"}});
```

```
        game.add({"type":"vehicles","name":"scout-tank","x":35,"y":25,"life":20,"direction":4,"team"
:"green", "orders":{"type":"patrol","from":{"x":35,"y":25},"to":{"x":35,"y":30}}});
    }
},
// Start moving the transport,
{"type":"timed","time":48000,
    "action":function(){
        game.showMessage("driver", "Thanks! Appreciate the backup. All right. Off we go.");
        game.sendCommand([-6],{"type":"move","to":{"x":0,"y":39,}});
    }
},
// Pilot asks for help when attacked
{"type":"conditional",
    "condition":function(){
        var pilot = game.getItemByUid(-5);
        return pilot.life<pilot.hitPoints;
    },
    "action":function(){
        game.showMessage("pilot", "We are under attack! Need assistance. This doesn't look good.");
    }
},
// Extra supplies from new transport
{"type":"conditional",
    "condition":function(){
        var driver = game.getItemByUid(-6);
        return driver && driver.x < 2 && driver.y>37;
    },
    "action":function(){
        game.showMessage("driver", "The rebels came out of nowhere. There was nothing we could do.
She saved our lives. Hope these supplies were worth it.");
        game.cash["blue"] += 1200;
    }
},
```

In the first timed trigger, the driver from the first mission asks for assistance while standing at the top-right corner of the map. We then command the pilot to guard the transport.

In the second timed trigger, the pilot announces she is on her way. We also add several enemy units to the map.

In the third timed trigger, which will be set off around the time the pilot arrives at the transport's location, we command the transport to start moving toward the base.

In the fourth conditional trigger—which is set off if the pilot's chopper gets attacked—the pilot messages asking for help.

In the final conditional trigger, which is set off if the transport reaches its destination, the driver talks about his experience, and the player's cash resources are increased.

If you run the game now, you should see a fairly interesting scene play out. You will see the pilot going out to help the driver when he calls for help. The pilot then protects the transport before getting ambushed by several enemy aircraft.

The transport continues to drive toward the base while under enemy fire. Once the driver reaches the base, the driver provides the player with some supplies and describes the experience, as shown in Figure 10-7.

Figure 10-7. *Driver describes the ordeal after getting back*

At the end of the whole experience, the player now has some extra cash to start building units.

Even though the story in our game is a little rushed, as you can see, this trigger mechanism can be used to tell a fairly interesting story. Obviously, the game framework can be extended to use a combination of video, audio, or animated GIFs to make the experience even more immersive.

Now let's add a trigger to set up the waves of enemy units, as shown in Listing 10-17.

Listing 10-17. Adding Enemy Waves (maps.js)

```
// Send in waves of enemies every 150 seconds
{"type":"timed","time":150000,"repeat":true,
    "action":function(){
        // Count aircraft and tanks already available to bad guys
        var wraithCount = 0;
        var chopperCount = 0;
        var scoutTankCount = 0;
        var heavyTankCount = 0;
        for (var i = game.items.length - 1; i >= 0; i--){
            var item = game.items[i];
            if(item.team=="green"){
                switch(item.name){
                    case "chopper":
                        chopperCount++;
                        break;
                    case "wraith":
```

```
                        wraithCount++;
                        break;
                    case "scout-tank":
                        scoutTankCount++;
                        break;
                    case "heavy-tank":
                        heavyTankCount++;
                        break;
                }
            }
        };

        // Make sure enemy has atleast two wraiths and two heavy tanks, and use the remaining
starports to build choppers and scouts
        if(wraithCount==0){
            // No wraiths alive. Ask both starports to make wraiths
            game.sendCommand([-23,-24],{type:"construct-unit",details:{type:"aircraft",name:
"wraith","orders":{"type":"hunt"}}});
        } else if (wraithCount==1){
            // One wraith alive. Ask starports to make one wraith and one chopper
            game.sendCommand([-23],{type:"construct-unit",details:{type:"aircraft",name:"wraith",
"orders":{"type":"hunt"}}});
            game.sendCommand([-24],{type:"construct-unit",details:{type:"aircraft",name:"chopper",
"orders":{"type":"hunt"}}});
        } else {
            // Two wraiths alive. Ask both starports to make choppers
            game.sendCommand([-23,-24],{type:"construct-unit",details:{type:"aircraft",name:
"chopper","orders":{"type":"hunt"}}});
        }

        if(heavyTankCount==0){
            // No heavy-tanks alive. Ask both starports to make heavy-tanks
            game.sendCommand([-21,-22],{type:"construct-unit",details:{type:"vehicles",name:
"heavy-tank","orders":{"type":"hunt"}}});
        } else if (heavyTankCount==1){
        // One heavy-tank alive. Ask starports to make one heavy-tank and one scout-tank
            game.sendCommand([-21],{type:"construct-unit",details:{type:"vehicles",name:
"heavy-tank","orders":{"type":"hunt"}}});
            game.sendCommand([-22],{type:"construct-unit",details:{type:"vehicles",name:
"scout-tank","orders":{"type":"hunt"}}});
        } else {
            // Two heavy-tanks alive. Ask both starports to make scout-tanks
            game.sendCommand([-21,-22],{type:"construct-unit",details:{type:"vehicles",name:
"scout-tank","orders":{"type":"hunt"}}});
        }
        // Ask any units on the field to attack
        var uids = [];
        for (var i=0; i < game.items.length; i++) {
            var item = game.items[i];
```

```
            if(item.team == "green" && item.canAttack){
                uids.push(item.uid);
            }
        };
        game.sendCommand(uids,{"type":"hunt"});
    }
},
```

Unlike the previous level, the enemy waves trigger is a little more intelligent. The timed trigger runs every 150 seconds. It first counts the number of enemy units of each type. It then decides which units to build based on what units are available. In this simple example, we first make sure that the green team has at least two wraiths to control the sky and two heavy tanks to control the ground, and if not, we build them at the starports. We build choppers and scout tanks on any remaining starports. Finally, we command all green team units that can attack to go on hunt mode.

If you run the game now, the enemy will send out waves every few minutes. This time, the composition of the enemy units will vary with each attack. If you don't plan your defense properly, you can expect to get overwhelmed by the enemy.

Obviously, this AI can be improved further by tweaking the enemy composition based on the player's composition or selecting the targets to attack intelligently. However, as you can see, even this simple set of instructions provides us with a fairly challenging enemy.

Now that we have a challenging enemy in the level, we will implement triggers for ending the mission, as shown in Listing 10-18.

Listing 10-18. Implementing the Ending (maps.js)

```
//After 8 minutes, start waiting for the end
{"type":"timed","time":480000,
    "action":function(){
        game.showMessage("op", "Commander! The colony air fleet is just a few minutes away.");
    }
},
//After 10 minutes send in reinforcements
{"type":"timed","time":600000,
    "action":function(){
        game.showMessage("op", "Commander! The colony air fleet is approaching");
        game.add({"type":"aircraft","name":"wraith","x":-1,"y":30, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"chopper","x":-1,"y":31, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"wraith","x":-1,"y":32, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"chopper","x":-1,"y":33, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"wraith","x":-1,"y":34, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"chopper","x":-1,"y":35, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"wraith","x":-1,"y":36, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"chopper","x":-1,"y":37, "team":"blue","orders":
{"type":"hunt"}});
```

```
        game.add({"type":"aircraft","name":"wraith","x":-1,"y":38, "team":"blue","orders":
{"type":"hunt"}});
        game.add({"type":"aircraft","name":"chopper","x":-1,"y":39, "team":"blue","orders":
{"type":"hunt"}});
    }
},
// And a minute after, end the level
{"type":"timed","time":660000,
    "action":function(){
        singleplayer.endLevel(true);

    }
},
```

The first trigger, eight minutes into the game, is the operator announcing that the colony air fleet has almost arrived. In the second trigger, two minutes later, we add an entire fleet of friendly aircraft in hunt mode. Finally, one minute after the fleet arrival we end the level.

Obviously, the only goal of this mission is to survive and protect the transport until the fleet arrives. If you play the mission now and survive that long, you should see the large fleet flying in and destroying the enemy, as shown in Figure 10-8.

Figure 10-8. *The colony air fleet flying in to save the day*

Once the fleet flies in, we have an extra minute to enjoy watching them attack and destroy the enemy before completing the last level in our single-player campaign.

Summary

In this chapter, we finally completed the entire single-player campaign of our RTS game. We started by adding sound to the game. We then used the framework that we built over the past few chapters to develop several levels for the campaign. We looked at ways to make the levels challenging and interesting by creatively using triggered events. We also learned how to weave a complete story into the game.

At this point, you have a complete, working, single-player RTS game that you can either extend or use for your own ideas. A good way to go forward from here is to try developing your own interesting levels for this campaign.

After that, if you are ready for something more challenging, you should try building your own game. You can use this code to prototype new game ideas fairly quickly by just modifying the artwork and adjusting the settings. If the feedback on your prototype is encouraging, you can then invest more time and effort to build a complete game.

Of course, while playing against the computer is fun, it can be a lot more fun challenging your friends. In the next two chapters, we will look at the HTML5 WebSocket API and how we can use it to build a multiplayer game so you can play against your friends over the network. So, once you have spent some time enjoying the single-player game, proceed on to the next chapter so that we can get started with multiplayer.