PHOTOSHOP FOR GAMES

Creating Art for **Console**, **Mobile**, and **Social Games**

SHAWN NELSON

PHOTOSHOP FOR GAMES

Creating Art for **Console**, **Mobile**, and **Social Games**

SHAWN NELSON

PHOTOSHOP FOR GAMES

Creating Art for Console, Mobile, and Social Games

Shawn Nelson

New Riders
www.newriders.com

To report errors, please send a note to errata@peachpit.com
New Riders is an imprint of Peachpit, a division of Pearson Education.

Acquisitions Editor: Karyn Johnson
Project Editor: Valerie Witte
Senior Production Editor: Lisa Brazieal
Developmental Editor: Bob Lindstrom
Copyeditor: Patricia Pane
Technical Editor: Jeff Cooperman
Composition: Kim Scott/Bumpy Design
Indexer: Karin Arrigoni
Cover Design: Aren Straiger
Interior Design: Kim Scott/Bumpy Design
Cover Images: Shawn Nelson

ISBN-13: 978-0-321-99020-4
ISBN-10: 0-321-99020-X

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

This book is dedicated to my darling wife, Katie,
whose constant support pushes me ever forward;
and to my two crazy kids Blake and Maggie,
who are often confused with eccentric monkeys,
especially in dining situations.

ACKNOWLEDGMENTS

I've been working in games since 1997 and teaching game-asset creation since 2004. I used the very first version of Photoshop and have worked in every update since. That being said, I find that every time I work with a new person or read a new book, I seem to find something new to add to my bag of Photoshop tricks. My hope is that this book does the same for you.

A bunch of people need to be thanked for helping me write this book: Karyn Johnson, the senior editor at Peachpit who helped me present the book and get it going; the fantastic development team who consistently made me look and sound smart: Valerie Witte, Bob Lindstrom, Patricia Pane, Lisa Brazieal, and Kim Scott. Thanks to Jeff Cooperman for making sure I was on the level, and to Lisa Milosevich Matthews for introducing me to the wonderful world of Peachpit.

Of course I need to thank my wonderful family, who gave me the space to put down some words and noodle some images—Katie, Blake, Maggie, Pooka, and Hoola—they are my inspiration and taskmasters, all in the same package. I would also like to thank all the wonderful friends I have worked with over the years who were kind enough to share their secrets of digital art.

For me, being an artist means you are on a never-ending quest to try new things and create new work; and trust me, it is a far more enjoyable journey when you have good people around you.

For more information about me (Shawn Nelson), please visit one of my websites: my work site at http://shaw04.wix.com/snelsondigart or my tiki-themed cartoon site at http://shaw04.wix.com/tikiislandprincess.

CONTENTS

INTRODUCTION

Digital games have been around for years, and although their look has continued to morph with the tide of technology, almost all of them have something in common: They all were built in part using Photoshop. Photoshop has proven to be a game-art juggernaut. Its use is so prevalent that you would be hard-pressed to find a game artist job description that does not include the words “Must know Photoshop.” With that in mind, this book features lessons specifically geared to introduce you to some of the most common uses of Photoshop in game development. It’s intended to give you the understanding you need to improve your skillset and turn your portfolio into a game-ready package.

ABOUT THIS BOOK

Photoshop is an essential tool used by artists who make games. This book will give you insight into some of those uses and help you enhance the skills you need to create game assets at a professional level.

Who This Book Is For

This book is for anyone interested in using Photoshop to make digital games. Whether you’re an experienced artist familiar with all things Photoshop, or you have only a basic understanding of digital art, this book will guide you through the necessary theory and practice you need to use Photoshop when making games.

How This Book Is Organized

This book features a series of lessons, each focusing on a part of the Photoshop interface, a tool, or a technique. Chapters 1 and 2 discuss general concepts used in making games for any format. Chapters 3, 4, and 5 delve into creating assets for specific platforms: mobile, social media, and console. Chapters 6 and 7 explore some advanced techniques that are not often used but are very powerful.

What This Book Covers

This book teaches students how to use Photoshop from its very basic interface to its most advanced tool sets. It also discusses topics such as 3D asset creation, lighting, and batch processing. Some of the theory discussed in this book may be new to you, but the intention is to advance your understanding of how to use Photoshop to create games.

HOW TO USE THIS BOOK

Whether using this book as a guide to self-study or in a classroom context, it is best to work through the lessons of this book in order, because each lesson builds on knowledge you acquire in the earlier lessons.

Each lesson consists of explanatory text and numbered steps designed to introduce you to new concepts and techniques. You should follow these steps precisely to gain the most useful learning experience. Performing steps out of order can produce unintended results and lead to a frustrating experience.

Using This Book for Self-Study

Progress at your own pace. Because you are not limited by time constraints, you may want to reinforce your learning by going through each lesson a second time, trying to do as much of the coursework as possible without specifically following the steps. In fact, as you go through the book, you are encouraged to pursue further experimentation on your own. Use the tools and techniques you've learned to take your art even further the second time around.

Using This Book in a Classroom Setting

This book offers an effective structure for teaching game creation in the classroom. We suggest that the trainer teach the lessons in book order with the students listening and writing down notes. Then the trainer can explain the steps while answering student questions and expanding on the text as necessary.

After a lesson is completed, students may need time to review the same chapter on their own in the classroom, under trainer supervision.

INSTALLING PHOTOSHOP

Although this book was originally written for Photoshop CC, the lessons can be followed successfully using more recent updates or legacy versions of Photoshop. Minor interface and behavior updates may account for slight differences between what the student sees on her screen and in the screenshots in the book.

DOWNLOADING PHOTOSHOP

To download Photoshop, follow these steps.

1. Go to www.adobe.com.
2. Navigate to Downloads.
3. From the product list, choose Photoshop.
4. Click the "Download trial" button.
5. Follow the instructions describing installation for your specific operating system.

ABOUT THE MEDIA FILES

As an added bonus, you can download some lessons that are available in video form. These videos feature step-by-step instructions for creating some of the game assets covered in the text.

Along with the videos, you will also find downloadable image files. These files are the images that I created while writing this book, and many still have the layers intact so you can reverse engineer the file if you need to.

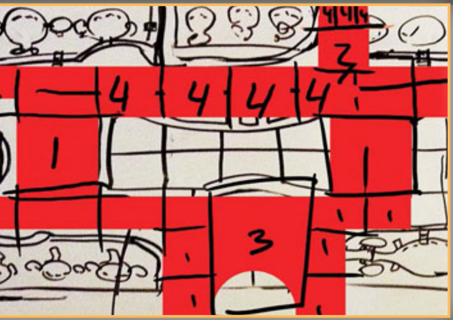
To access the files and install them on your computer, follow these steps:

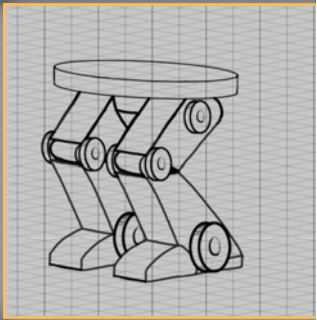
1. On a Mac or PC, go to www.peachpit.com.
2. If you don't yet have a Peachpit.com account, you will need to create one.
3. Enter the book's ISBN or go directly to the book's product page to register. Once on the book's page, click the Register Your Product link. The book will show up in your list of registered products along with a link to the book's bonus content.
4. Click the link to access the media files for the book. When the download is complete, double-click the archive to decompress it.

4

CREATING A MOBILE GAME

In this chapter, you will examine the process of making a mobile game. Mobile and social games are closely related in content size, and somewhat in style, but they have each evolved into their own separate spaces.





IS THAT A GAME ON MY PHONE?

NOTE To access the resource files and videos, just log in or join peachpit.com, and enter the book's ISBN. After you register the book, links to the files will be listed on your Account page under Registered Products.

All jokes aside, mobile games have by far the most potential of all three of the game types discussed in this book. Console and social games will continue to look better as screen sizes increase and hardware gets faster and smarter; but the largest strides in the future will be on your phone and for just one dominating reason: Everyone has one.

That built-in audience is a prime target for companies that profit by reaching out to as many people as possible. So where do you think all their technology development is going to be aimed? Right at your pocket.

Fortunately for game makers, the returns from game and game-related purchases are high enough to justify continued investment in mobile game companies. The more companies out there, the better chance you have of getting work. So when it is reported that mobile gamers spend over 30 minutes per day playing games, you should smile a little because that is good news for the industry.

THE PIPELINE FOR MAKING MOBILE GAMES

The mobile world as it stands today has been playing catch-up. The computer and console memory specs and computing power are miles ahead of mobile devices. However, it is not an even race. Mobile hardware, spec-wise, is at about the level of a good PC from the year 1994. But considering that the smart mobile phone has not been around as long as the PC, it's doing pretty well.

So when you look at the graphics of mobile games available now and are wondering where the future of mobile gaming is headed, look no further than the current PC/Mac games market for a glimpse of the future (**FIGURE 4.1**).

FIGURE 4.1 Modern mobile games.

Mobile Game Specifics

Knowing the limitations of a mobile device and how to push its performance as far as possible has become a sought-after skill in the game industry. Those who know how to squeeze the most out of every little bit of bandwidth and memory are integral to creating the best-looking games. Right now, console games are at the techno-forefront, followed by social games, with mobile showing up in third place. Obviously, the key to success for the mobile gaming world is being able to replicate the console experience on a phone. Developers are on their way toward that goal, but they are just not there yet.

Levels

The levels in a mobile game are a bit different than you might expect. Take your average games featuring a robot on an alien planet with hostile locals, for example. A normal level in a console game would probably provide about 2 hours of play. That 2 hours x 10 levels costs you 60 bucks, or around 20 hours of average game play.

A mobile game works a little differently. The levels are far shorter, lasting only several minutes at the most. This is because most mobile game players only play while waiting for the bus or standing in line at the bank. Getting involved in a long campaign in which you have to start and stop all the time just does not go over very well. Also, implementing a "save anywhere" feature gets a little memory heavy and is usually not possible in the world of mobile hardware.

Characters

The characters in a mobile game follow this same trend. In a console game, the characters might have millions of polygons. (Polygons are the 3D version of pixels, the basic unit of 3D graphics. They will be further explained in Chapter 5, "Creating Console Game Assets.") Mobile hardware is challenged to animate a character with only thousands of polygons. That makes quite a visual difference. Why is this? Phones just don't have enough computing power yet. Moving the characters, calculating the polygons, and mapping the textures is very memory and processor intensive.

The Bank

You might think of the constraints for making a mobile game as similar to dealing with a bank. You have only so much money in the bank that you can draw on. Although you can take money out, the bank is not going to give you more money than you start with. Mobile games are like that. The phone can only process so much data at a time. If you factor all the things in a game happening at once—the audio, the effects, the playable character, the non-playable characters, the environment, the HUD, all the code, tracking your play history—you can see how you quickly you can become technologically overdrawn in a mobile game.

How do you work with this? You budget your game, processing only those things you need at the moment you need them. Those Level Three aliens do not need to be loaded in Level One so they are stored (probably in a compressed file format) until they are needed. When their time comes, the Level One aliens are dumped from memory so all available storage and computing power can go to running their Level Three companions. Similar resource budgeting and asset swapping is applied to audio and special effects and levels. This way you are always using the most amount of game budget on precisely what you need to be showing at the time.

Your creative choices can also contribute to optimizing your use of hardware resources. If you need more aliens onscreen in Level Four, then Level Four is designed as a graphically simple environment with snow and no trees to calculate. With mobile games, you borrow from here to enhance there. If you find you need trees in the level, then you reduce the aliens' graphic detail. Same fixed techno-budget, you just spend it differently.

NOTE A frame rate is the number of full-screen images per second that a device can display. Higher frame rates require increased processing power. The lower the frame rate, the choppier the onscreen movement. Unfortunately (that budget thing again), increasing graphic detail lowers frame rate.

Digital game makers have worked like this since games began. Working out a realistic technology "budget" is usually done at the beginning of a project, and then revised on the fly as the project progresses. The keepers of this techno-bank are the engineers. They are ultimately responsible for ensuring that the game remains within a certain memory footprint, and able to run at a reasonable frame rate.

Aspect Ratios and Screen Resolutions

Mobile phones, smartphones, cells, and whatever else you want to call them, mobile devices come in a variety of sizes, and as with the social game scene, they represent an ever-moving target to try and find an aspect ratio and resolution to build to (**FIGURE 4.2**). As the mobile device window size remains in constant flux, the resolution gets better and better, and the amount of pixels one can use at a time also shifts. Fortunately, it always seems to lean toward more pixels not less, so that is good news.

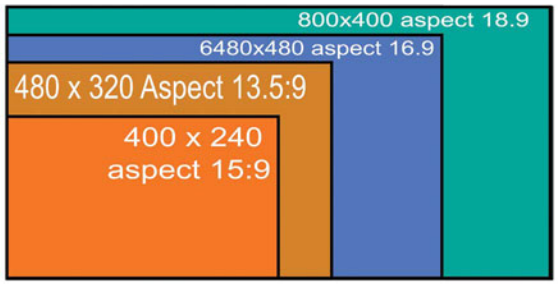


FIGURE 4.2 Mobile pixel ratios.

Generally speaking, before you start a mobile game you do some research to see what the current trend in screen specs is. Just like you did with the social game, you pick one that will allow you to stretch or reduce the screen to fit other screens that might be just a little off your numbers. This allows you to make one game that will be playable on multiple phone types.

EXERCISE 7: CREATING ASSETS FOR A MOBILE ROBOT SHOOTER

With the standard graphic tools you've already used, as well as some new ones, you will create assets for a 3D robot shooter game. You will create characters, several level maps, and some asset textures. You will also animate a 2D explosion effect using a sprite sheet.

The Story

Your game this time is called *Nebula Pirate Robots*. In this game, you don your robot armor, and with two robot friends, land in an alien environment where you fight your way through bloodthirsty aliens who want to steal all the energy pods in the universes. You and your friends are also collecting the pods to power up your super robots and defeat the alien queen, shown in **FIGURE 4.3**.

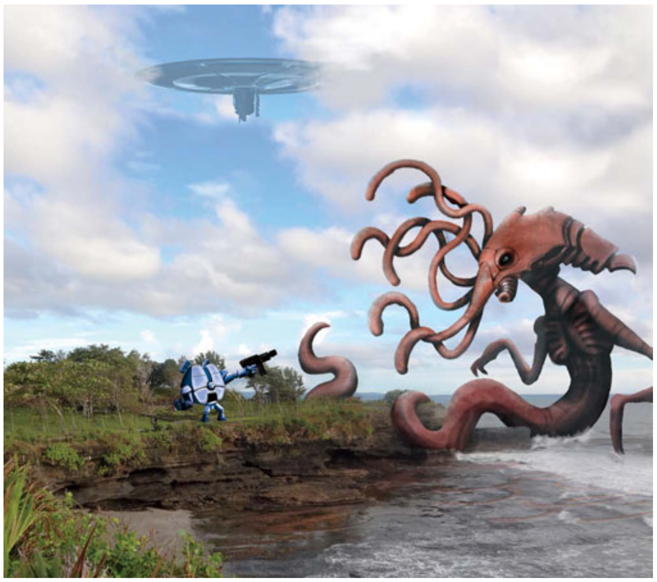


FIGURE 4.3 Fighting the alien queen.

How to Begin

By now you should be familiar with the art process for starting a game.

- Create a theme or a story
- Collect images in mood boards
- Sketch conceptual drawings
- Initiate product development
- Build the game
- Make loads of money and become famous

For this game, the art director had some mood pieces created by a company that specializes in doing conceptual illustrations (**FIGURE 4.4**). This is a common industry practice when the art director knows what look he is going for and no one on his team can satisfactorily meet those needs. These images will be used for inspiration instead of mood boards. From these images, you will start work on some character variation sketches.

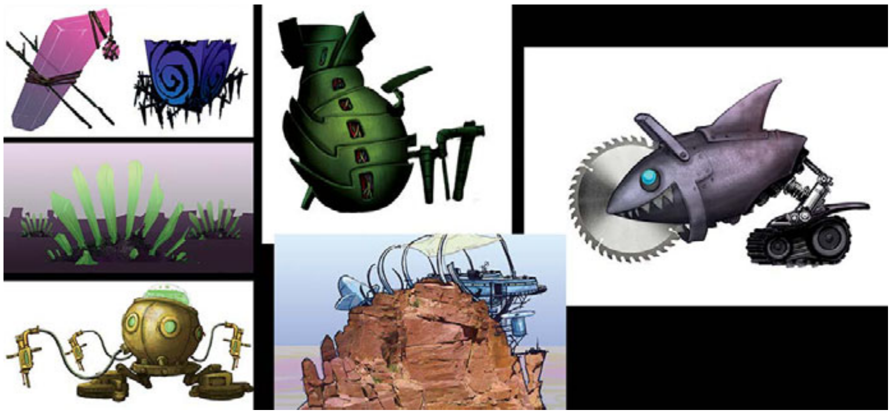


FIGURE 4.4 Conceptual illustrations for a robot game.

Character Variations

You draw character variation sketches when you have a pretty good idea of what your character is going to look like but want to explore subtle variations in its look, or when you need multiple versions of a single character for a team or a squad. For this exercise, you will create a three-robot squad, all about the same size and built using the same material but with different-looking shells.

In a mobile game—where you need to cut any corner you can to save on memory—you should determine in advance the assets you will need for the game and how to most efficiently implement them. Remember, everything you put into a mobile game has a cost, so the more times you can multipurpose the same assets, the better it is.

TIP One function in Photoshop will come in very handy here. First, select the Pen tool and the size and style of brush you would like. Shift-click anywhere on the page. Then while still pressing the Shift key, move your mouse pointer somewhere else and click again. Photoshop will draw a line between the two clicked points. This is not a vector line, as is created with the Shape tool, but it is a good raster facsimile.

Creating the lower half of the robots

To save on memory, all three robotic units will have the same legs. This means you only need to make one set of 3D legs, with one set of animations, which only have to be coded once. Multipurpose. You will then create different chassis for the legs and change their color to match each new chassis. Changing the color is the easiest (and techno-cheapest) way to create a visual difference.

1. Open a new file, 15 inches wide by 5 inches high at 300 dpi.
2. Open the diametric perspective grid provided in your Resources folder (**FIGURE 4.5**). Scale the grid to fit in the window, which should give you the proper perspective depth on the grid.

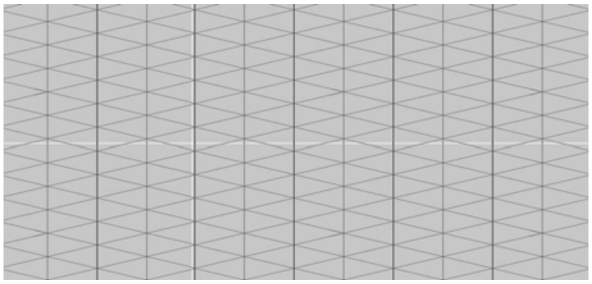


FIGURE 4.5 A diametric perspective grid.

NOTE A diametric perspective grid allows you to view a grid in a *fake* 3D environment. This will be helpful here because you are designing robots to later be constructed in 3D modeling software.

3. Begin sketching the lower half of the robot from about the waist down. Use the grid lines as the outline for the shape of the left leg. The guidelines that you used in drawing this foot are highlighted in red (**FIGURE 4.6**).

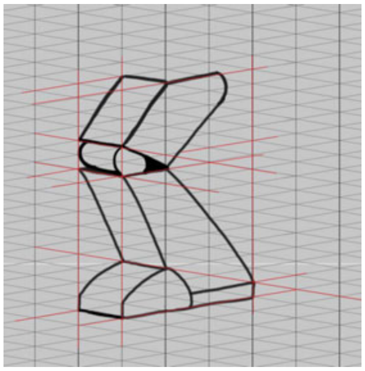


FIGURE 4.6 Use volume boxes to help draw shapes.

4. Duplicate the layer and translate it 1 grid space to the right while remaining on the front line that the toes touch (**FIGURE 4.7**). Use the corners of the toes as a guide. As you use the grid more, it becomes easier to see the lines that should not be there. Erase any lines that would not show on a solid object (**FIGURE 4.8**).

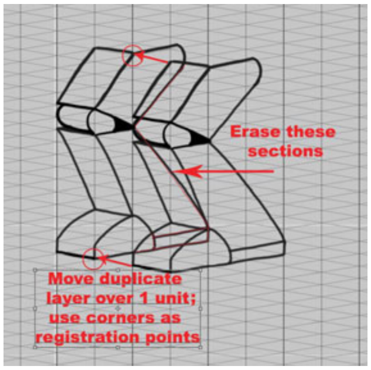

Erase these sections
Move duplicate layer over 1 unit; use corners as registration points

FIGURE 4.7 Erase any lines that would not show on a solid object.

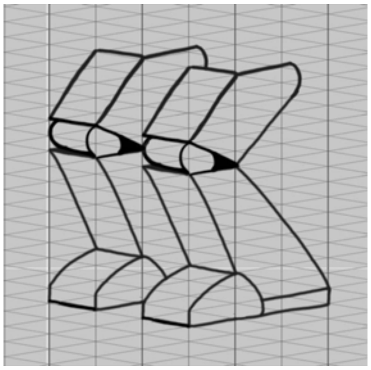


FIGURE 4.8 The robot feet.

5. When the second leg is in place, begin work on the waist disk. This is where the two halves of each robot will connect.

   Because a disk is a hard shape to draw in perspective, you will need to create a volume box using the Line tool (shown in **FIGURE 4.9** in red). You then create an oval shape using the Shape tool with no fill and a 1-pt. stroke. The oval shape is then fitted into the bottom square of the red volume box by choosing the Edit > Transform > Distort tool.

6. Duplicate the oval layer and translate it up to the top section of the red volume box. You now have a top and a bottom oval. Connect the two with a line drawn from one to the other at the far edges of the ovals, creating a volume (**FIGURE 4.10**). Readjust the height of the hip disk, if necessary, and then erase any lines that may have strayed from the design or bled over.

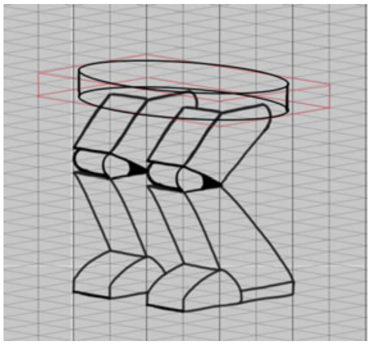


FIGURE 4.9 **Create the torso base using an oval shape for the hip disk.**

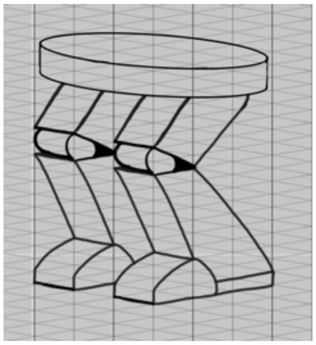


FIGURE 4.10 **Fine-tune the torso base.**

Now you will create the leg joints, using the same technique to create a cylinder shape on the left ankle.

7. Using the Shape or Brush tool, create a center bolt.

8. Copy this shape to the right ankle, the right knee, the left knee, and the left hip. Then erase any extra lines that need to be removed to make this look like a solid volume (**FIGURE 4.11**).

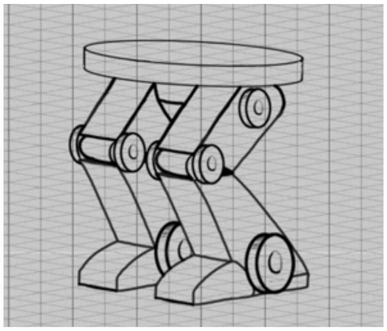


FIGURE 4.11 **The completed lower portion of the robot.**

Collect all the lower-half layers into a group, and name it *legs*. Scale down the group 30 percent, and move it toward the bottom of the page. Make sure you realign it with the background grid.

Creating a robot torso

Now, let's move on to the body (**FIGURE 4.12**).

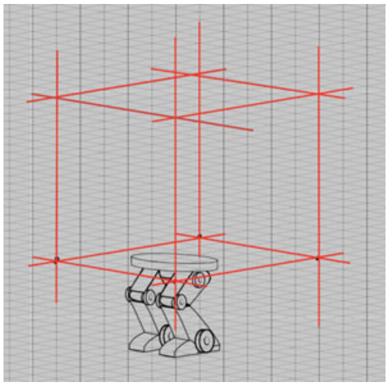


FIGURE 4.12 Volume box for the robot torso.

1. Start as you did when drawing the hip disk. Define a volume of a size appropriate for the torso. Use the Line tool with a **red** color so it is easy to see. If you're having trouble, try counting the spaces on the background grid. Start at the hip disk and find the line to the right that is one big square away. Stripe a big line. Now do the same on the left, followed by the front. In the back, go two spaces. Your robot is going to have stuff back there and you need a bit of extra room.
2. Using the volume (or bounding box, as it is called in 3D), create your robot shape (**FIGURE 4.13**). You will find that the first line is the hardest. Once it is set, the rest of the shape flows much more easily.

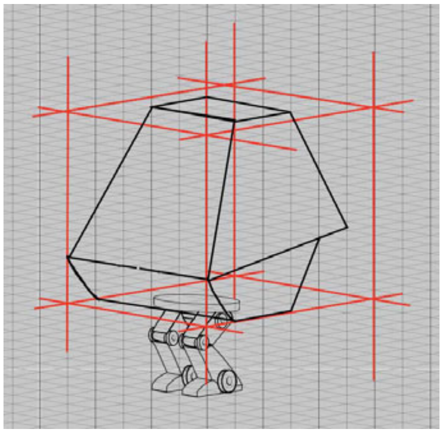


FIGURE 4.13 Begin sketching the torso.

3. When your shape is defined, select the red volume and the black outline layers, and collect them into a new layer group, and name it *Torso*. You can hide the red line for now by turning off its visibility.
4. Now create the head in a skull shape (**FIGURE 4.14**).
5. To build a volume box for the left arm, use a red Line tool, and guided by the grid, lay out a rough arm shape off to the side of the robot (**FIGURE 4.15**). Don't worry about the detail right now, just make sure it is roughly arm shaped. The appropriate volume is what you are after.

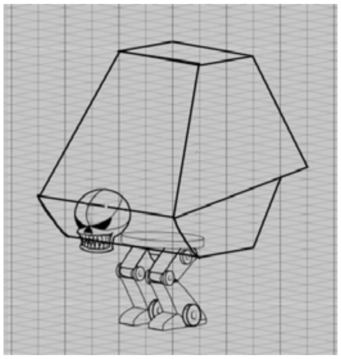


FIGURE 4.14 Position the skull within the main shape of the torso.

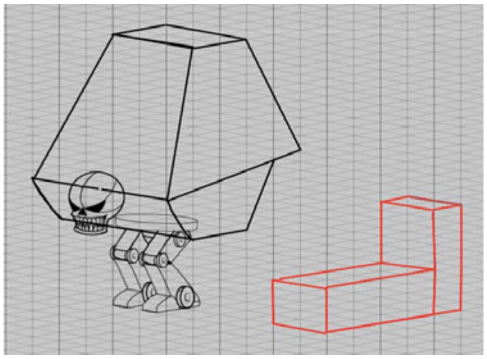


FIGURE 4.15 Create the volume box for the arm.

6. Once the volume box is completed, start creating your arm (**FIGURE 4.16**). Keep in mind that this will be pretty small onscreen and a lot of detail might not show up. It is better to try and make the larger shapes work for you. Also, remember that the fewer pieces you have to build in 3D, the better.

   At this point things will start to get confusing, with lines intersecting all over the place. It is helpful to create a backfill layer for all of the elements. This will give you the option to eliminate the transparency if you need to, and aid in the final render (**FIGURE 4.17**).

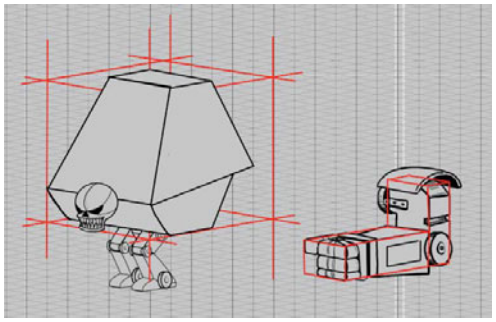


FIGURE 4.16 Create the arm.

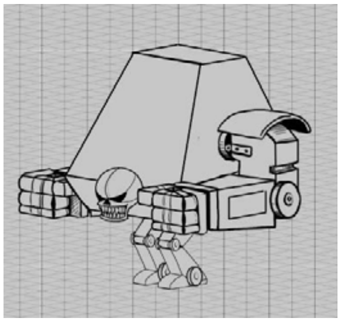


FIGURE 4.17 Create a background shape to control opacity.

7. Select and duplicate a layer group such as "legs." Flatten the new layers, (not the originals) and with the Magic Wand tool, select the flattened shape. From the grid, select the **lighter gray** color, and choose Edit > Fill. Fill the shape with the foreground color you just selected. Rename this layer *backfill*, and place it on the bottom of the original stack in the "legs" group. Repeat this process for all the groups.

8. Move the arm group into place. You will also need to draw the small bit of hand that is showing on the right side. Make sure that layer is below the torso layer.

   Now it's time to decorate. You might have design requests from the designer or feel inspired by a favorite missile launcher, but now it's time to design some attached weapons.

9. Create a new layer at the top of the layer stack, and draw some weapons in place (**FIGURE 4.18**). You might need to erase some of the torso or some of the other sets, and that is fine. Draw your attachments first, and later you can go back into the black line sections and erase all you want.

   If you find that you are still having a hard time seeing what is going on, you can add tone to the shape (Figure 4.18).

TIP A good way to create a non-raster oval is to use the Ellipse selection tool and choose Stroke from the Edit > Stroke menu. Make it black at 4 pts. Make sure it is on its own layer. (You can add a stroke to any selection shape, by the way.) Keeping the shape on its own layer allows you to move it around and erase it if you need to.

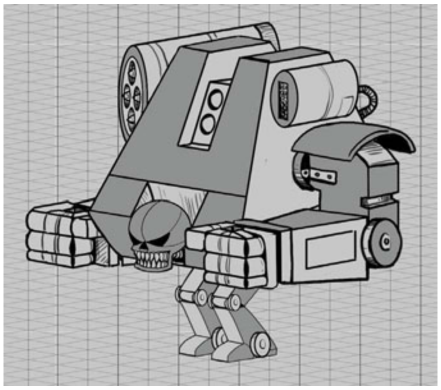


FIGURE 4.18 Add shadow to enhance the image.

10. Select and duplicate all the layers and layer groups. Merge all the duplicate layers, and use the Fill tool to select a darker or lighter **gray**. Do not add color yet.

11. Next, duplicate the "legs" group, and name the new layer group *M2_legs*. Move the volume boxes for the torso and the arm into place, and you are ready to make the next robot in the series (**FIGURE 4.19**).

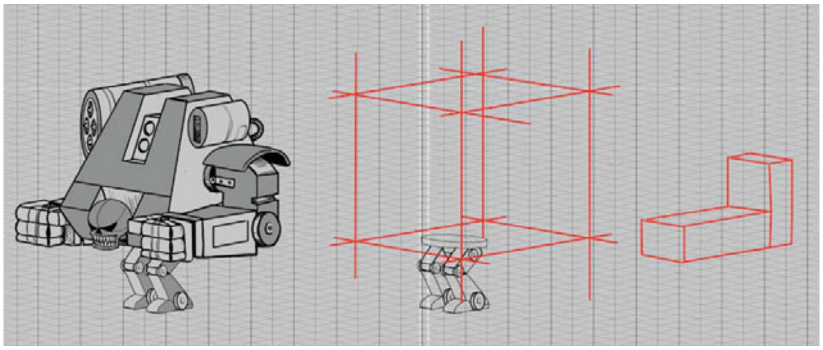


FIGURE 4.19 Begin creating the second robot.

To create a second robot, continue your robot assembly line by designing a new model to establish some visual contrast.

1. Begin as you did previously by defining the torso above the leg group.

   Try to make this design unlike the silhouette of your previous robot to create a solid visual difference between the two. If it helps, you can give them classifications like "water robot" and "air robot." Knowing what a robot will be used for may help you think about what attributes to use in your design.

2. When you've finished your torso, and repeated the backfill layer creation and the layer grouping, you can move on to creating the head. As before, when making a new torso, your emphasis should be on designing a very different-looking head. If you created a round head previously, then create a square one this time. When you are happy with the design, create the backfill layer, and collect the layers in a layer group called *m2_head*.

3. You have the torso and the head, now let's finish the arm. Using the arm volume box again, begin to create the arm. As with the head and the torso, strive to make this arm a bit different from the previous robot's arms. Sometimes finding an interesting way to bend the elbow or an unusual hand shape will dictate the look of the arm. Those are both great places to start. Think also how this arm will attach to the robot body and what its range of movement might be. When you get something you like, create the backfill layer, and collect the arm and the backfill into a layer group named *m3_leftarm*. You will also need to create the right arm and do the whole backfill/layer group thing.

4. Following the same process, create any weapons or torso details you might want, if you have not done so in the torso layer. You will need to do the whole backfill/layer group procedure here as well.

   Everything is good to go. So, collect all of the m2_ layers in a layer group called *Robot_02* and move them into the layer stack just above the Robot_01 layer.

Creating robot number 3

Because you've already done this twice, you should be able to create the third robot on your own. If you get lost, just refer to the previous sections for help.

When your three robots are finished, you can add color to them. Either add the same color to all three to show the variation on one robot, or make them three different colors to show one robot as three different characters.

The same process you used to make a drawing of a robot could eventually be used to make the 3D version. The legs would be modeled separately from the torso. The arms would get the same treatment. Each would get its own animations to match the timing of the others.

This 3D modeling process can get quite complicated, but in the end, it is the way to go. Why is this the way to go? It will buy you loads of different robots if you want them. In the engine, any arm can be combined with any torso, as long as they are rigged correctly and use the same animations. So with just one set of legs, three torsos, and three arms, you get tons of variations, and more importantly, upgrade items that can be sold to the user (**FIGURE 4.20**).

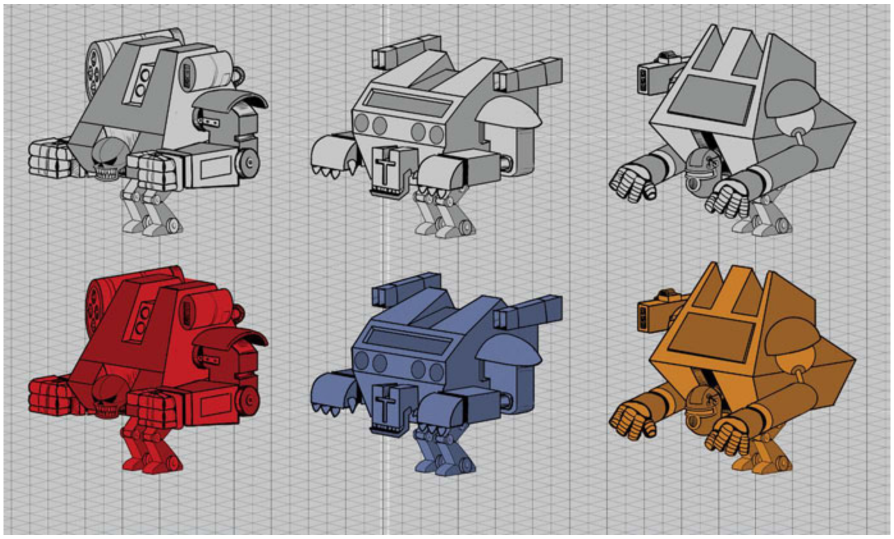


FIGURE 4.20 Robot variations.

A single arm can be mirrored in the engine along with its animation, so you really only need to make either a left or a right arm.

CREATING LEVELS

In the beginning, there were robots, and the designer saw these robots and said, “Now where can they fight swarms of aliens for somewhere between one to two minutes?”

Creating levels is a job that takes a village. Many things have to come together to make a level work correctly, and if you plan on churning out hundreds of levels that only last a minute or two each, you’d better make sure that everyone knows what they are doing from the start. If not, you will be redoing a lot of stuff in your future.

What is this stuff, you ask? Well, it all starts with—guess what—an idea that is usually handed down from the designer, or in a larger studio, a writer who hands it off to the designer. In your sample game world, the designer and the art director hired a concept company to flesh out some images to be used as examples or inspiration for levels. The look of the level you need to create is shown in **FIGURE 4.21**.

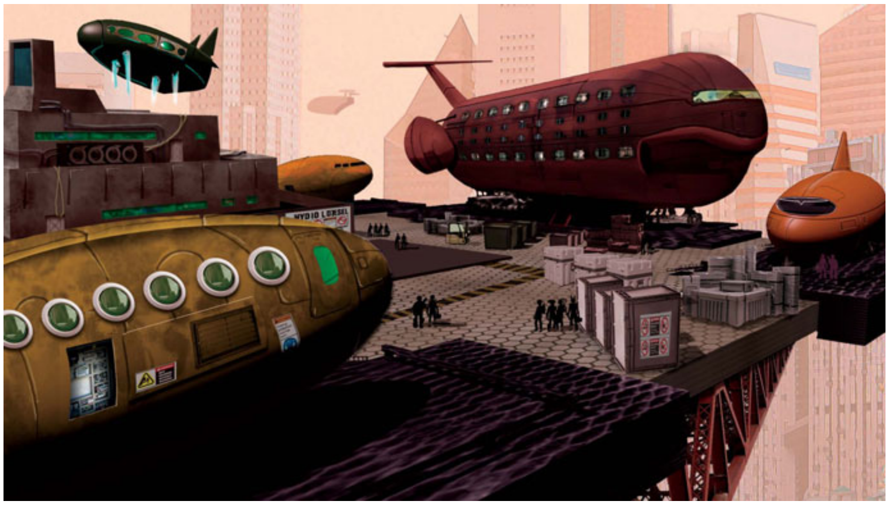


FIGURE 4.21 Spaceport city.

The designer’s instruction to the concept house for this image was:

Create a spaceport with ships coming and going, and cargo stacked around. It should be industrial, and have a dirty modern feel to it. This is the first level of the game.

The concept house did its job, and it is enough for you to start laying out some of the groundwork that will turn into this level.

The first thing you need to do is identify all of the elements that need to be in the level to make it work. These might be a road, or a special rock, or a bridge the player has to cross. Your level design must include anything that is a critical part of the gameplay.

You can get this list from the designer, who has thought this through. Even so, it is also a good idea to sit down with the designer and have her walk you through her design process (**FIGURE 4.22**). This is the kind of meeting that dry erase boards were invented for.

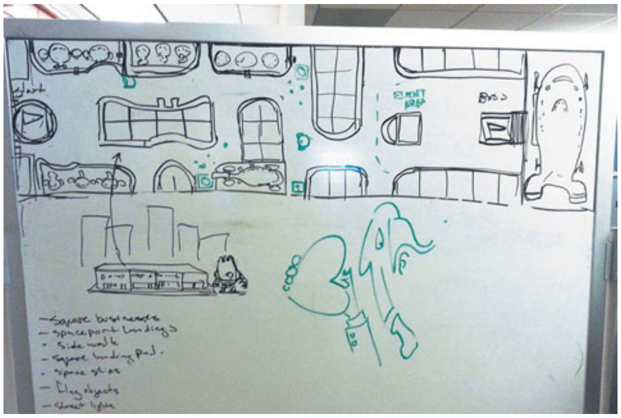


FIGURE 4.22 **Doodles on the whiteboard.**

Here are the mission-critical items that the designer wanted addressed. Keep in mind these might change, but you need to start somewhere.

- Square buildings
- Spaceport landing pads
- Sidewalks
- Square landing pad
- Spaceships
- Flag objects
- Streetlights
- A floor is implied
- A height elevation drawing showing how tall the robots can be

At this point, you should be able to draw a rough layout. It has a beginning, an end, and a variety of stuff in the middle.

Creating a Basic Level Map

The level map is created in multiple sections that can be recolored and moved around as the level is dialed in. The sections are the ground plane, the road, the buildings, and all the movable objects such as crates and trees. When you are done, you will have a pretty good idea of what the level will look like from a top-down view. By conceiving the level in this way, you can make changes before time is spent in 3D modeling.

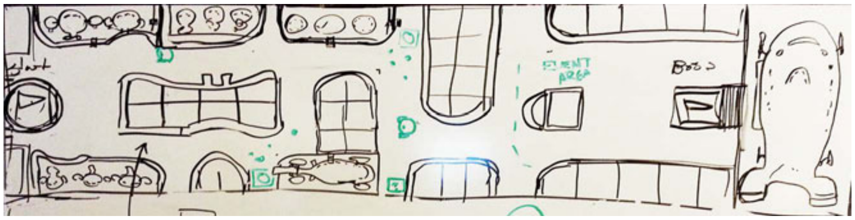
A good way to start is to take a picture of the whiteboard and the work you did with the designer. You can then bring that right into Photoshop and draw on top of it, after some careful cropping and stretching. Keep in mind that this is just the rough sketch, not the final. The map could change quite a bit before you are done.

Creating the Navmesh

The first step is creating the ground that your characters will walk upon.

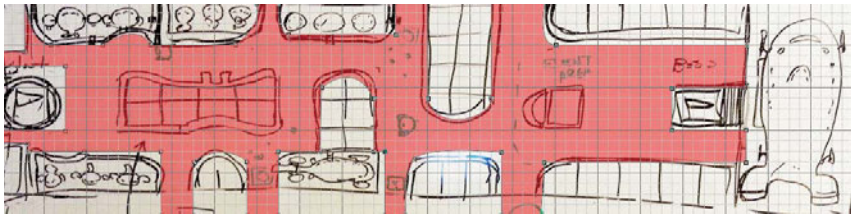
1. Once you have your image ready to go, create a new file in Photoshop. Name it *LV1_Spaceport*, and make it 20 inches wide by 5 inches high at 300 dpi. In the filename, LV stands for level, and you add "1" to indicate the first level, followed by a description of what is in that level.
2. From the rough map file, which you should have open, select the layer that has the rough map on it, and drag it into your new LV1_Spaceport file. Save the file.
3. Create a new layer and call it *Navmesh* (**FIGURE 4.23**). This word is an abbreviation of *navigation mesh*, and it will be the ground on which the hero and the bad guys will travel. By constraining the characters' movements to a particular piece of geometry, you ensure that they will not clip (penetrate) through larger pieces of the world. You also only have to create the physics code once, and then just apply it to the path. That way any new characters you add will follow the rules of the code you set as long as they are walking on the path. Also, physics code is expensive to run so you do not want it applied to the whole world, especially to the places you can't go.

FIGURE 4.23 Navmesh sketch.



4. Set the navmesh layer Opacity to **60%**. Using the Pen tool, choose a **red** color (**FIGURE 4.24**). Trace the outline of the floor area as best you can. Do not worry about the details too much, you will clean it up later. In fact, the fewer points you can use, the better.

FIGURE 4.24 Highlight areas of possible travel.



5. Now choose View > Show > Grid to turn on the grid. This will help you clean up your lines.
6. Choose the Direct Selection tool, a few tools down from the Pen tool. Try to align the points with the nearest grid point, so when your 3D modeling begins, the shapes can be built with a minimal number of polygons (**FIGURE 4.25**). Everything you do at this point should be geared to saving polys.

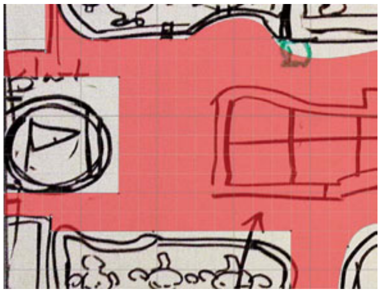


FIGURE 4.25 Create a vector image to get right angles in your mesh.

7. When you get your shape to a respectable state, you can create a new layer just below the navmesh layer, and merge the shape down. Make sure that you raise the layer opacity before you do so. This merge will convert the navmesh from a vector shape to a raster image. You did not ultimately want this shape as a vector shape, but it is easier to align it in that state. You could do this whole part with the Brush tool, if you wanted, but you would need a very steady hand.

TIP Press the number keys to quickly set visibility (1 is 10% and so on).

8. With the shape now editable in raster mode, you can clean up any stray lines and make any cutout corrections you might require. This navmesh will be used as a template to make the polygon mesh (**FIGURE 4.26**). For now, you will not apply any colors.

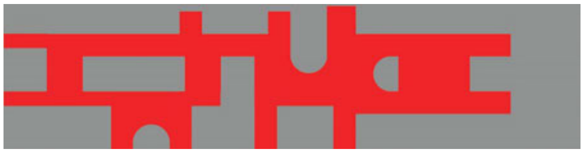


FIGURE 4.26 Vectored navmesh.

Creating the ground plane

Creating the ground plane is very easy because it is, very simply, the size of the whole file. In this case, it's 6000x1500 pixels at 300 dpi. The ground plane is the very bottom of the stack, which means it is the most obscured of the elements. This is great because in 3D, the ground plane usually receives a tiled texture, which is a 2D image that is applied to the polygon multiple times to make the ground plane look like a single image. It is usually a lot smaller than 6000x1500, more like 512x512. The repeating texture then fills in the 6000x1500 area. 512? Why this odd number? It's all about the Power of Two rule.

THE POWER OF TWO RULE

The Power of Two refers mostly to a system that is used in making textures for 3D models. Computers process data in chunks that must be in a certain format and size or the computer that is processing them will have to work harder, which can limit game performance. Textures in sizes that conform to a power of two—such as 8, 16, 32, 64, 128, 256, 512, 1024, and 2048—get through the processing pipe faster. It becomes a bit of a limitation on the art side to work this way, but if it means getting more textures into the game, it's worth it. Also, it has been the established method for years, so everyone works this way.

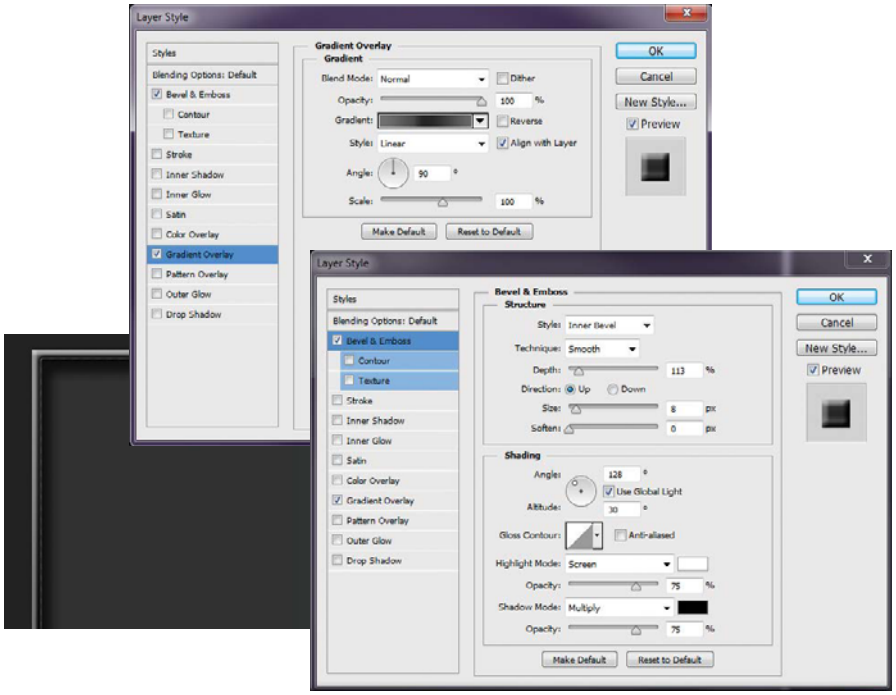
You are going to create a 512x512 texture tile that will be applied to the ground plane. It will be a tiling texture that looks like it is not repeating, especially when you are suggesting grass or dirt (**FIGURE 4.27**). For this reason, natural tiles are generally harder to make. Second, a tiling texture must be able to be located next to another tile on any of the four sides of the square, without the seam showing.

FIGURE 4.27 Ground textures.

Because this level is a spaceport, you will be making a metal grid tile.

FIGURE 4.28 Create a metal tile for the outer frame.

1. Begin by creating a file that is 1024x1024 at 72 dpi. (Most final images going into an engine are at 72 dpi.) You will reduce this to 512x512 later, but if you need a higher-resolution version in the future, you will have it. Fill it with a **dark gray** color (**FIGURE 4.28**).
2. Choose Image > Canvas Size, and set it to 1050x1050, which will create a perfect stroke space around the outside edge. Fill the new empty space with a **lighter gray**. Choose Image > Size, and change the size back to 1024x1024.
3. Using the Magic Wand tool, select the inner square and cut it from the file. Then choose Edit > Paste Special > Past in Place. Name the lower layer *trim*, and the upper layer *base*.
4. Select the trim layer, and in the blending options, select Bevel & Emboss and Gradient Overlay (**FIGURE 4.29**). Set the Depth to **113** and the Size to **8**.


Layer Style

Styles
Blending Options: Default
☑ Bevel & Emboss
☐ Contour
☐ Texture
☐ Stroke
☐ Inner Shadow
☐ Inner Glow
☐ Satin
☐ Color Overlay
☑ Gradient Overlay
☐ Pattern Overlay
☐ Outer Glow
☐ Drop Shadow

Gradient Overlay
Gradient
Blend Mode: Normal ☐ Dither
Opacity: 100 %
Gradient: ☐ Reverse
Style: Linear ☑ Align with Layer
Angle: 90 °
Scale: 100 %
Make Default Reset to Default

OK
Cancel
New Style...
☑ Preview

Layer Style

Styles
Blending Options: Default
☑ Bevel & Emboss
☐ Contour
☐ Texture
☐ Stroke
☐ Inner Shadow
☐ Inner Glow
☐ Satin
☐ Color Overlay
☑ Gradient Overlay
☐ Pattern Overlay
☐ Outer Glow
☐ Drop Shadow

Bevel & Emboss
Structure
Style: Inner Bevel
Technique: Smooth
Depth: 113 %
Direction: ● Up ○ Down
Size: 8 px
Soften: 0 px

Shading
Angle: 128 °
☑ Use Global Light
Altitude: 30 °
Gloss Contour: ☐ Anti-aliased
Highlight Mode: Screen
Opacity: 75 %
Shadow Mode: Multiply
Opacity: 75 %
Make Default Reset to Default

OK
Cancel
New Style...
☑ Preview

FIGURE 4.29 Blending options.

5. Using the Shape tool, pick an industrial-looking pattern from the shape gallery. In this example, the multiple diagonal line shape was used (**FIGURE 4.30**). Position the pattern in the middle of the square so there is a small border. In the blending options, select Bevel & Emboss and Drop Shadow. Make sure Bevel & Emboss is set to **Pillow Emboss**, and that the Depth is set to **164** (**FIGURE 4.31**).

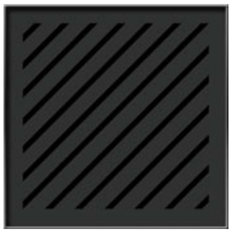
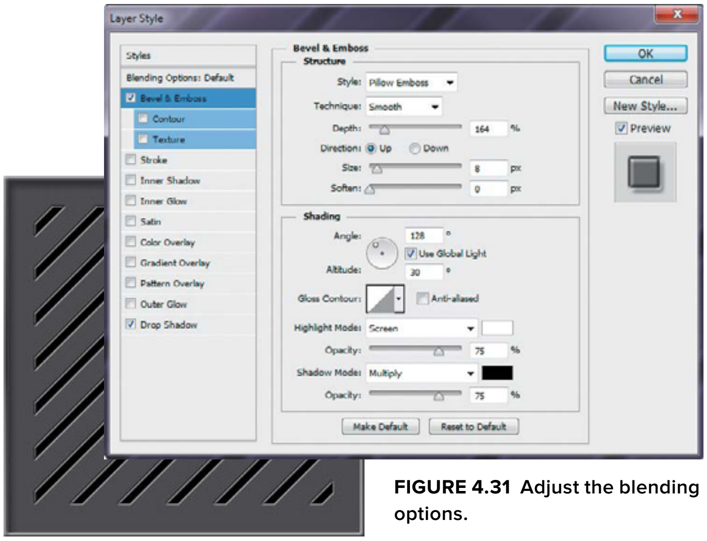


FIGURE 4.30 Place the shape from the gallery.


Layer Style

Styles
Blending Options: Default
☑ Bevel & Emboss
☐ Contour
☐ Texture
☐ Stroke
☐ Inner Shadow
☐ Inner Glow
☐ Satin
☐ Color Overlay
☐ Gradient Overlay
☐ Pattern Overlay
☐ Outer Glow
☑ Drop Shadow

Bevel & Emboss
Structure
Style: Pillow Emboss
Technique: Smooth
Depth: 164 %
Direction: ● Up ○ Down
Size: 8 px
Soften: 0 px

Shading
Angle: 128 °
☑ Use Global Light
Altitude: 30 °
Gloss Contour: ☐ Anti-aliased
Highlight Mode: Screen
Opacity: 75 %
Shadow Mode: Multiply
Opacity: 75 %
Make Default Reset to Default

OK
Cancel
New Style...
☑ Preview

FIGURE 4.31 Adjust the blending options.

6. Create a new layer called *emblem*. Using the Ellipse tool, create a circle in the middle of the page (**FIGURE 4.32**). Fill it with the base color, and once again select Bevel & Emboss and Drop Shadow.
7. From the Custom Shape tool library, select a shape that will work as an emblem, such as the atomic shape. Center it in the circle, and open the blending options. Select Bevel & Emboss and Drop Shadow. Make sure the Style is set to **Emboss** and the Direction is set to **Down** (**FIGURE 4.33**). Save your file.

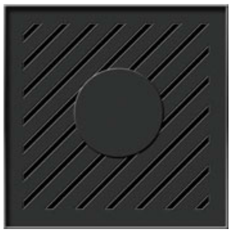
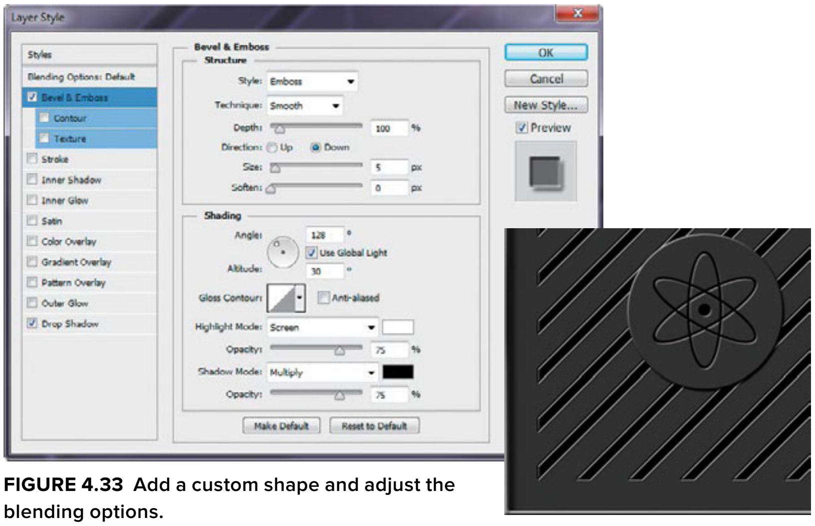


FIGURE 4.32 Add a circle shape.


Layer Style

Styles
Blending Options: Default
[✓] Bevel & Emboss
[ ] Contour
[ ] Texture
[ ] Stroke
[ ] Inner Shadow
[ ] Inner Glow
[ ] Satin
[ ] Color Overlay
[ ] Gradient Overlay
[ ] Pattern Overlay
[ ] Outer Glow
[✓] Drop Shadow

Bevel & Emboss
Structure
Style: Emboss
Technique: Smooth
Depth: 100 %
Direction: Up [✓ Down]
Size: 5 px
Soften: 0 px

Shading
Angle: 128 °
[✓] Use Global Light
Altitude: 30 °
Gloss Contour: [ ] Anti-aliased
Highlight Mode: Screen
Opacity: 75 %
Shadow Mode: Multiply
Opacity: 75 %

FIGURE 4.33 Add a custom shape and adjust the blending options.

8. Create another layer called *dirt*. Select one of the scatter brushes, such as number 24, and then select **black** as your fill color. Dot the pattern around the layer, changing the size and position (**FIGURE 4.34**).
9. Create a new layer and call it *dirt2*. Select an **orange** color, and switch to a different scatter brush (**FIGURE 4.35**). Repeat the painting process, but do not paint as much of the area as you did previously.
10. Create a new layer, and call it *dirt3*. Using a **blue** color and a new scatter brush, once again apply small doses of paint, and add more shapes to the mix (**FIGURE 4.36**).

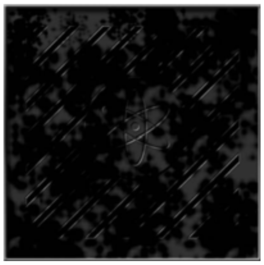


FIGURE 4.34 Begin creating a "dirt" layer.

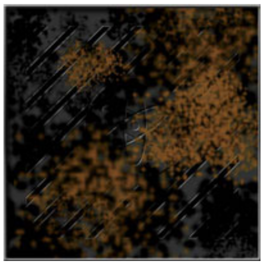


FIGURE 4.35 Add color to the dirt layer.

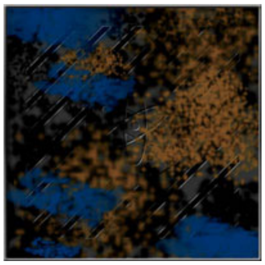


FIGURE 4.36 Add more color and shapes to the dirt layer.

11. Adjust the opacity of the three layers until you are satisfied with the look. When you are satisfied, merge the three layers together, return to the diagonal shape layer, and select the inside of the grooves using the Magic Wand. Then on the dirt layer, erase the dirt in the grooves but not the dirt on the emblem section.
12. If you feel the need, you can add another dirt map. In the example, the shadows were accentuated a bit with a layer inserted under the emblem layer.
13. Now it is time to test your tile. Save the file as a TIFF or PSD. Collapse all the layers and save it as a PNG. Choose Image > Canvas Size, and change the file size to 2048x2048. Locate your image in the upper-left corner. Duplicate the layer three times and move each layer to a corner of the square (**FIGURE 4.37**). Check for obvious mistakes and verify the tile blends with other tiles.

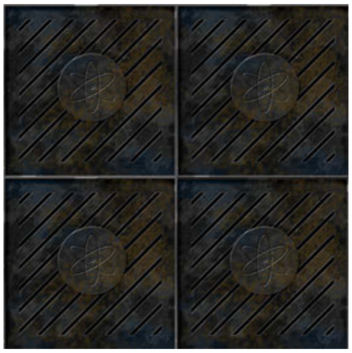


FIGURE 4.37 Duplicate the tile and position one in each corner.

When a tile like this is used in a game, a shader and lighting are often applied. So even if the lighting in the drawing is not perfect, the engine will bring it together. That being said, it is a good idea to do the best you can with the illustration because the game engine can do only so much to make a tile work.

FIGURE 4.38 shows what the tile looks like rendered in 3ds Max with simple lighting and an object added to show what a shadow looks like on the tile. The tile in the image is set to repeat five times over the surface of the ground plane. A simple shader has also been added.

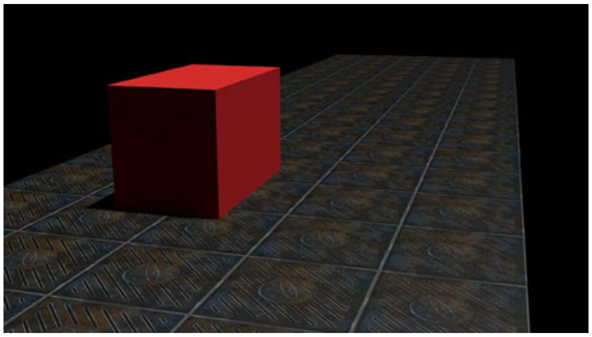


FIGURE 4.38 The tiles applied to a ground plane.

14. Now that you know the ground plane will work, you can apply it to your level map. Save a PNG version of your tile that has only one tile. You may need to crop it to fit. Make sure it is 1024x1024.
15. Open your level map file again and import the tile image. Tile the image across the entire ground plane without going beyond the borders (**FIGURE 4.39**). This is best done by placing the tile down, and then selecting and duplicating the two tiles. Then select the four new tile layers and duplicate them.
16. When you have the area filled, collapse all the layers together into one layer called *ground tile*, and scale it the little extra bit to fit the area. You just need an idea of how it works; you have the tile for the actual game asset.

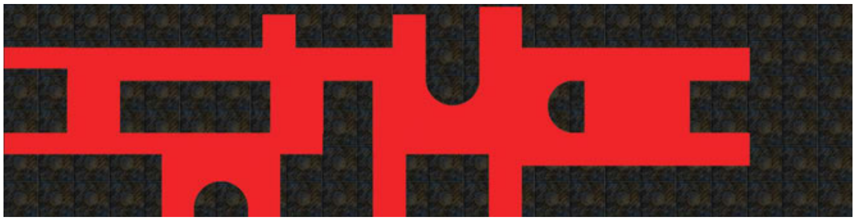


FIGURE 4.39 The navmesh on top of the ground plane.

Creating Stand-in Buildings

Designing a building is a bit of a trip down architecture lane. Just as in real architecture, the design must fit into your environment. You can't have a Victorian house at your spaceport; it just won't do. So you need to design a building that will look right for the location. It should have all the trappings of a spaceport café, or storage unit, or customs office. Think space station meets shipping yard. Actually, any of those would be fine, but because the concept illustration had a café in it, that is where you will start.

Because you are only making a level map, you have no real need to flesh out a fully realized building design. Players will see it only from the top. But in the interest of learning, you will flesh out an isometric view of the café, and then generate a top-down view based on that design.

You can go about designing a structure for a 3D environment in two ways. One is to just design the building as best you can, and then let the 3D modeler figure out how to bring it in within the polygon budget. The other is to design the building with the poly budget in mind, making sure that everything you place on the page has been considered from a modeling point of view. You are going to use the second method.

1. Begin by opening up the diametric grid file you used on the robots. Scale the grid down in height by 50% to get a more front-on perspective. Make sure your file is 6 inches high by 10 inches wide at 300 dpi (**FIGURE 4.40**).

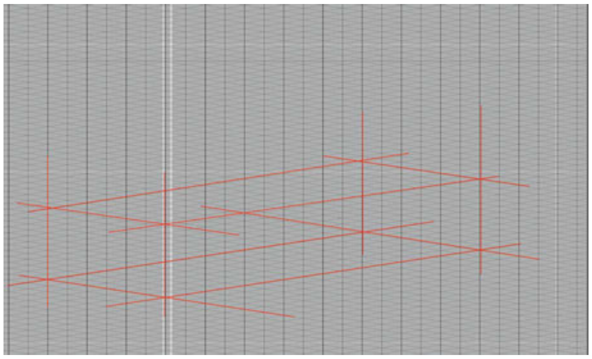


FIGURE 4.40 Create the volume box for the first part of the building.

2. Using the Line tool with a **red** color at **4 pts**, begin to lay out the foundation of your space café. It will be three polygonal shapes: the main floor, the smaller second story, and the roof sign. It has purposefully been kept simple to keep the poly count low. But this means that you'll have to do quite a bit with texture.
3. Once you have the main floor in place, create a new layer called *2ndstory*, and repeat what you did with the first layer, except leave a bit of a border. When you are dealing with low poly models, any surface area that you can create to reflect light is a good one.
4. When you have finished the 2ndstory layer, create a new layer called *roof sign*. Using the Line tool, finish creating the layout for the shape (**FIGURE 4.41**). It will look a little like a bunch of crazy toothpicks. Do not worry.

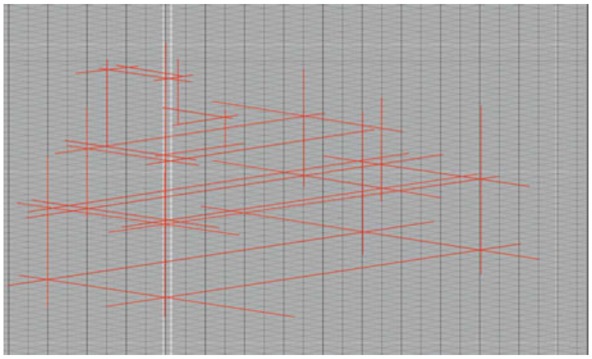


FIGURE 4.41 Create the volume box for the second part of the building.

5. Select the Line tool and choose a **black** color, and keep the size at **4 pts**. Create a new layer called *blackline*, and begin to outline your shape (**FIGURE 4.42**). If it helps, you can reduce the opacity of the red line layers just a bit.

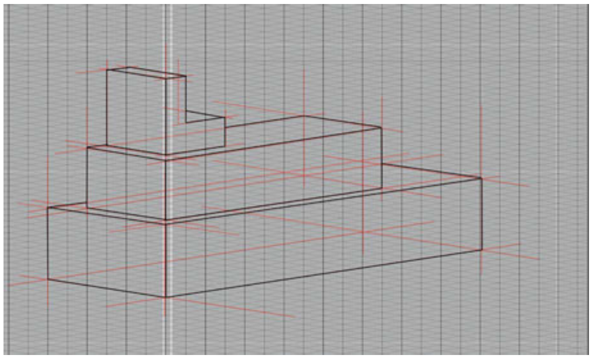


FIGURE 4.42 Begin creating the building.

With the main structure in place, you can start working on the decorations, which include windows, doors, vents, signs, and pipes. In 3D, most of these will just become part of the texture, but that would depend on how many polys you've spent in other places. However, these elements always look better in 3D if you actually build them.

Begin with the doors (**FIGURE 4.43**). They are good to nail down because they give the structure a sense of scale. Your average industrial external door is about 7 feet high by 34 inches wide. Space café doors are just a little bit larger than that. Don't worry too much about fully fleshing out the doors right now. Just use the grid to indicate where they are. If you really can't help yourself, you can indicate a door jamb.

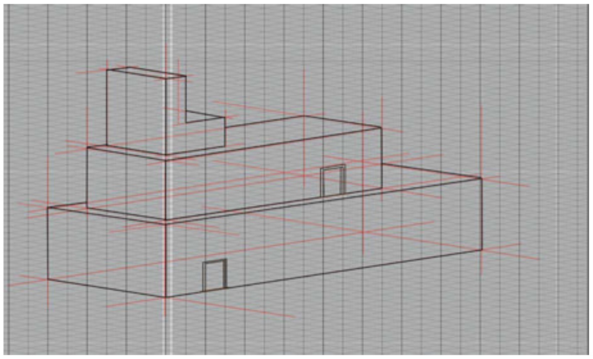


FIGURE 4.43 Flesh out the doors.

6. With the doors in place, the next logical element is the windows (**FIGURE 4.44**). Windows in a space café differ from those in a real diner. Instead of being big and open, they are small, more akin to what you would see on a boat or an airplane. Smaller windows will help sell the idea that the environment is in space. Notice that the windows were also given a bit of dimension.

   Now you will add some things that look terribly important, but are really there just to fill some of the empty space. Repetition on a building is very common. In fact, you almost expect it. So let's not disappoint.

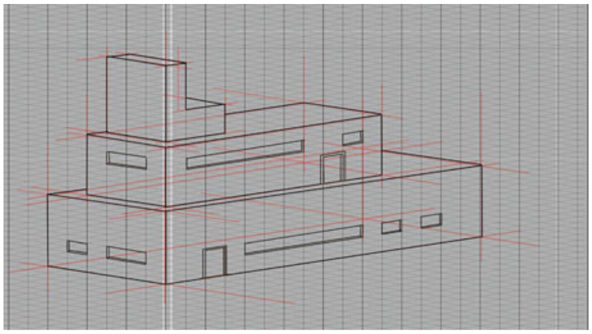


FIGURE 4.44 Create the windows.

Just above the lower door, use the Pen tool to create a trapezoidal shape similar in size to the door (**FIGURE 4.45**). Make sure the bottom line is on a horizontal grid line and that the outer edges also line up with vertical grid lines. You will need these markers in the next step. The look you are going for is a protruding piece of geometry.

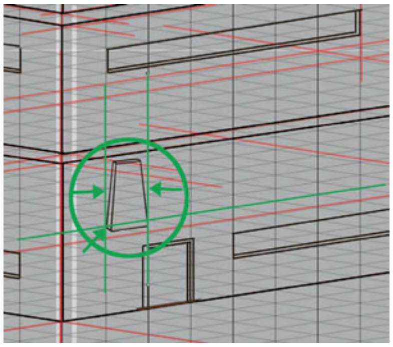


FIGURE 4.45 Create the first part of the wall decoration.

7. Now collapse any shapes to a single layer, and call it *decoration*. Duplicate the decoration layer and slide it 2 grid spaces over to the right. Repeat this until you run out of building (**FIGURE 4.46**). Collect all the layers into a layer group, and name it *decorations lower*.

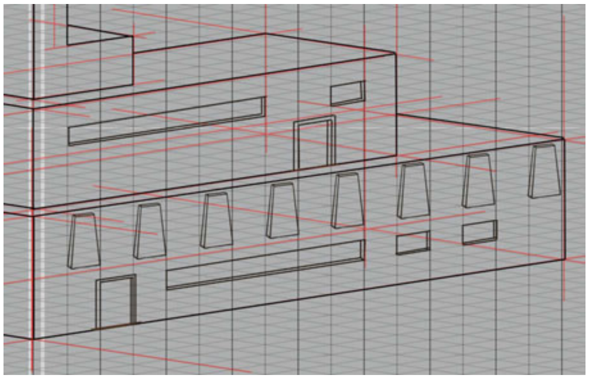


FIGURE 4.46 Duplicate the decoration layer.

8. In the remaining empty spaces on the café, continue to create visual distraction or interest, depending on your view of life. In **FIGURE 4.47**, you can see the space for signage, a large tank of some sort, and a ventilation unit. All of the things a space café cannot do without.

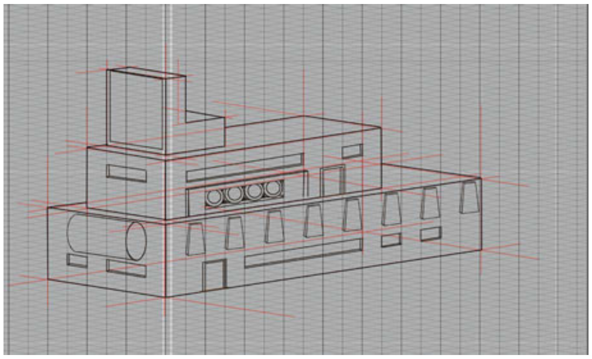


FIGURE 4.47 Continue to create visual interest by adding details.

9. Create a new layer and call it *sidewalk*. Choose the Line tool and create a **1-unit** sidewalk around the base of the building (**FIGURE 4.48**). This will help sell the scale of the building and give you a place later on to put things such as lights and space café vending machines.

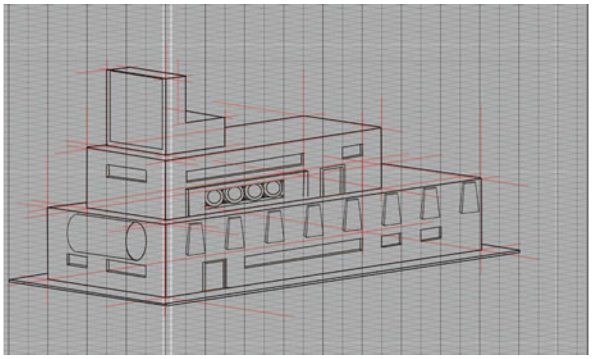


FIGURE 4.48 Create a 1-unit sidewalk.

10. Collect all of the layers and duplicate them. Collapse the new layers into a single layer, and name it *color*. Fill in the shapes you have created using **gray** tones. You can also use tonal ramps if you want. Color the windows in **green** (**FIGURE 4.49**).

    You should have produced more than enough for the 3D artist to start work, and far more than you need for your level map. If you were concepting this, you would also apply colors and finished details such as the signage and perhaps stuff in the windows. You would also do a reverse angle if it were needed.

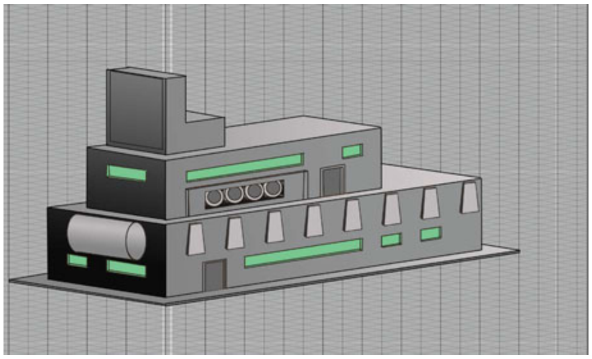


FIGURE 4.49 The finished space café with colors applied.

Creating real stand-in buildings

Creating the building markers for a real level map is far easier than what you just did. The kind of building you actually need to make has the footprint of the building and callouts for any doors, windows, or turrets that might be important. No grids, no shading, very simple. The building images are generally done as in an architectural drawing: a large whitish area with a few hash marks and an indication of which way the door swings (**FIGURE 4.50**). If the building is accessible, you might need some indication of the inside layout, but usually that is a separate map with more detail.

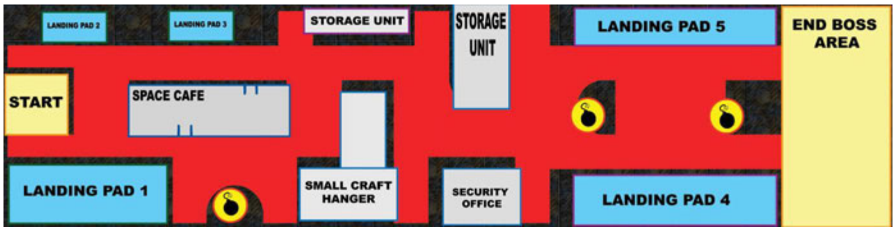


FIGURE 4.50 Create the level layout tags.

1. Open the level map file you were using. With the Shape tool, create a rectangle that fits the area in the middle. Fill the shape with a **white-gray** color and make the Stroke **blue**. With the Text tool, write *SPACE CAFÉ*. Collapse the layers into a single layer called *space café*.
2. Repeat this for all the sections, giving each a name based on what you and the designer discussed. Because these items may change at any time, make sure that they are all on their own layers.

   Most studios have a visual shorthand for calling out repeated items without spelling out the actual name again and again. This accounts for the yellow dots with the red borders and the little pictures of bombs. They stand for a destructible object in this case; that is, an object that can be shot and has the potential for causing damage to bad guys or other players. You might also see shorthand symbols for things like lights, spawning points, save points, and various other common game objects.

3. To make the destructible object icon, choose the Ellipse tool and draw a small **circle**. Make the fill **yellow** and the border **red**. Create a new layer above that layer, and with the Brush tool, create a little image of a bomb. Merge the two layers together, and name the new layer *destructible object*.

   This map is now more or less ready to take to the designer. She will place notations on a new layer to indicate where spawn points may be and what special purposes the various buildings might have, if any. After that, the map is distributed to the team as a guide for what is needed to flesh out the level. A list of assets will be created, and engineering will document the tasks that might be needed for this particular scenario to explain the functionality. The main point of a map like this is to get everyone synchronized on the objectives rather than waste time and energy spinning in the wrong direction.

Creating a Texture Atlas for the Navmesh

Remember that tile you made for the ground plane? You're going to take it a step further. A texture atlas is a group of textures that are all on one sheet, and then applied to one or more objects repeatedly in a game engine to give a more diverse, less tiled look. The reason for putting them on one sheet is that they load faster as a slightly larger file than they would as five little files. You're going to make a series of tiles, collect them on an atlas, and then apply them to the navmesh.

First, you need to identify the types of tiles you will need for the navmesh object. You do this by looking at the layer in Photoshop and seeing how it turns and bends, while trying to come up with some options that will work in multiple locations.

1. Open the level map file. Hide everything but the red navmesh layer.
2. Create a new layer. Identify similar shapes in the path that could use a single tile once or multiple times. With the Brush tool, mark these in rough outline using a black color. Identify as many sections as you need.
3. Number each similarly shaped area.

   As you can see in **FIGURE 4.51**, you have identified four map types. Each of these tiles will need to look as if they are made from the same material, but they do not all need to tile from every direction as did the previous tile.

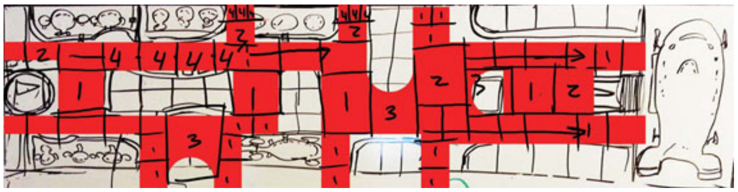


FIGURE 4.51 Number each similarly shaped area.

Now you will break each of the numbered areas out into individual maps. Remember that each map needs to fit into a power of two square, which is 512x512.

4. Open a new file in Photoshop that is 1024x1024 at 72 dpi. Name it *navmesh_t1*. As you can see, this is a larger area tile, which means that the detail will have to be smaller.

   "Why is that?" you ask. Let me explain. The tiles you make will start out the same on the tile atlas, but when applied to the sections of the object in a 3D software package, they may end up displaying at different sizes (**FIGURE 4.52**). So, if you have details like the black border and the 33 in this image, they must start out at different sizes in the 512 maps so that when reduced they are the same scale.

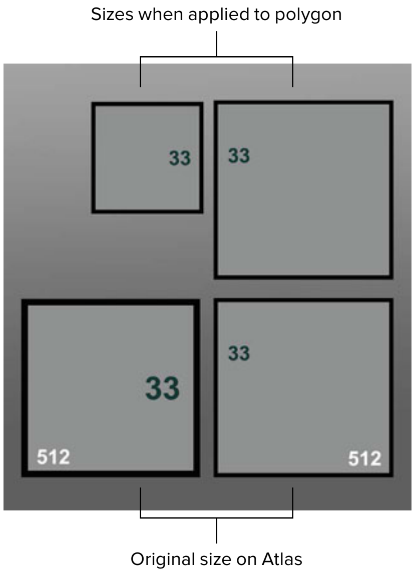


FIGURE 4.52 Tiles may end up displaying at different sizes.

   Notice how large the black trim and the 33 have to be in the original tile to appear to be the same size in the model. It is important that you identify these situations in advance, or you will get way down the road with your project and then have to redo your work.

5. Create a new file that is 1024 inches in height and 4540 inches in width at 72 dpi. Select an example of each of the tile shapes, and import them into the new file, placed next to each other. You do not need to add the numbers here.

   Once you have the pieces, you can begin work on filling them in with space station road-looking art. A good way to start is to create a stroke around each of the shapes (**FIGURE 4.53**). Because these will look like metal floor tiles in the game, a border is a great natural break. It will look like the edge of the tile. If you were drawing dirt, it would be much harder to hide the seams of the tile.

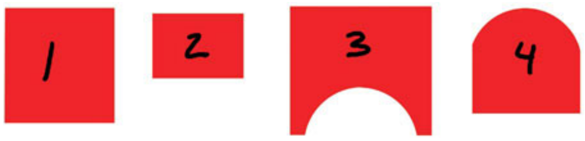


FIGURE 4.53 Lay out your shapes next to each other.

6. Fill the shapes with a **gray-green** color (**FIGURE 4.54**). This is a popular space-station floor color. In the blending options, turn on Bevel & Emboss, and set the Depth to **360**. Then create a new layer and call it *background*. Move it to the bottom of the stack, and fill it with a **gray** color. Once again, this is done so you can see what you are doing.

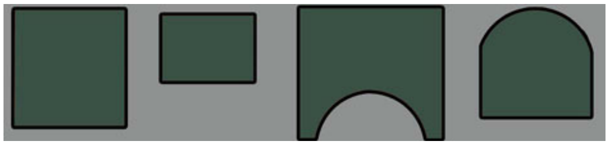


FIGURE 4.54 Fill the shapes with a gray-green color.

7. Select all the green areas, and copy and paste them. This will create a new layer that you can name *lid*. Name the original layer *base*. In the blending option of the lid layer, turn on Drop Shadow and Bevel & Emboss. Marquee select the first shape in the lid layer, and center it over its base layer counterpart (**FIGURE 4.55**). Repeat this for all the shapes. Note that the third arch shape will also need to be scaled a bit to fit.

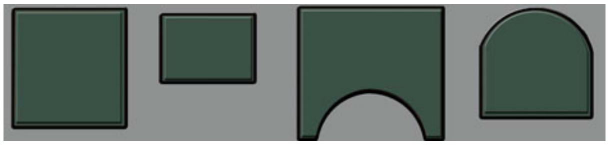
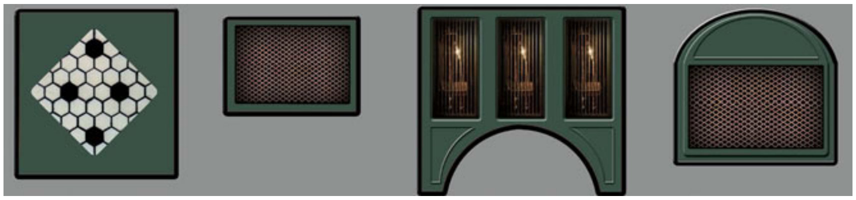


FIGURE 4.55 In the blending option of the lid layer, turn on Drop Shadow and Bevel & Emboss.

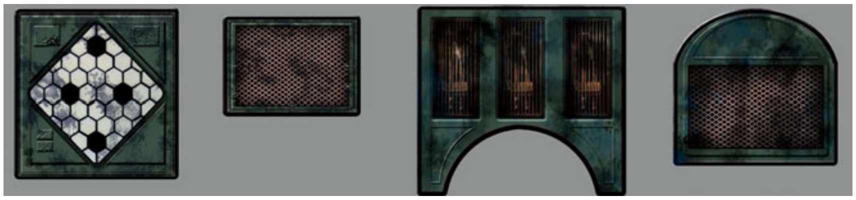
8. From here you can add a central theme to each tile (**FIGURE 4.56**). You have gone with a grate theme because this spaceport has a lot of rain, and drainage is important. Oh, and you added one tile one for visual variety.

FIGURE 4.56 Add a central theme to each tile.



9. After your central theme is in place, you can create a new layer above the others and call it *dirt map1*. You're going to be adding some years to your tiles, just as you did with the ground plane tiles (**FIGURE 4.57**).

FIGURE 4.57 Add some colored dirt to the tiles.



10. Select a black color and a scatter brush, just as you did before, to create some interesting shapes. If you go too heavy with the paint, you can lower the opacity, or select the Eraser tool and a different brush and work back the other way. When you are happy with the mess you have made, create another map, and using a color (such as blue), add some colored dirt.
11. Duplicate all the layers and, with the exception of the gray background, merge the new ones together. Hide the old layers.
12. Create a new file that is 1024x1024 at 72 dpi, and name it *texture atlas*. This will be your texture atlas, as the name suggests. You will stack your files in a square so they will be made easier to read by the game engine.
13. Create a red, square shape that is 512x512. Place it in the upper-left corner, and duplicate it to the lower-right corner (**FIGURE 4.58**). Collapse the layers. These squares will be used to size your texture maps to 512x512.
14. Marquee select tile number 1, and paste it within the upper-left red square. Scale the image until it covers most, if not all, of the red square. As you can see in **FIGURE 4.59**, you have now made your texture 512x512.
15. Repeat this process for the remaining tiles, stretching them to fit the entire space (**FIGURE 4.60**). Don't worry; you will squish them back to their normal sizes later.

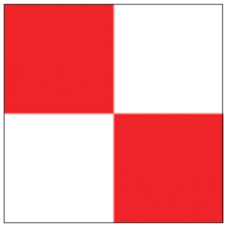


FIGURE 4.58 Create a red, square shape that is 512x512.

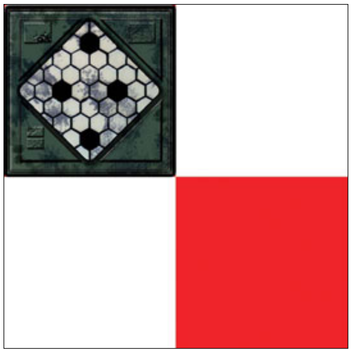


FIGURE 4.59 Marquee select tile number 1.

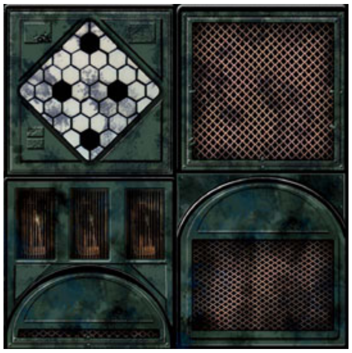


FIGURE 4.60 Repeat the selection/paste tile process for the remaining tiles.

16. Create a layer above the red-checkered level and fill it with a black color. Save the file as a TIFF and a PNG. Obviously, some space on this atlas has not been utilized. Space on a texture atlas is never wasted, and usually you would put a texture for something else into these areas. For now, you will leave it blank, but you should make a mental note that you have some space for a small texture if you need it.

    With your texture atlas, you will now simulate how it would be applied to a mesh object in a 3D software package such as Autodesk Maya or 3ds Max.

Applying the Texture Atlas to the Navmesh

Without the aid of 3D software, applying a texture to a mesh is a bit difficult, to say the least, but you can simulate the process using Photoshop layers and some creative pasting.

In a 3D world, a 2D image is projected onto the surface of a 3D object. The projection can be cut into small sections based on the 3D topology. Each polygon can have its own 2D map, but most of the time the process goes more like this: One 2D image is carefully positioned over multiple polys on an object to give the illusion of it having color, being painted, or wearing pants or dragon scales, or whatever (**FIGURE 4.61**). That is exactly what you're going to do with your navmesh.

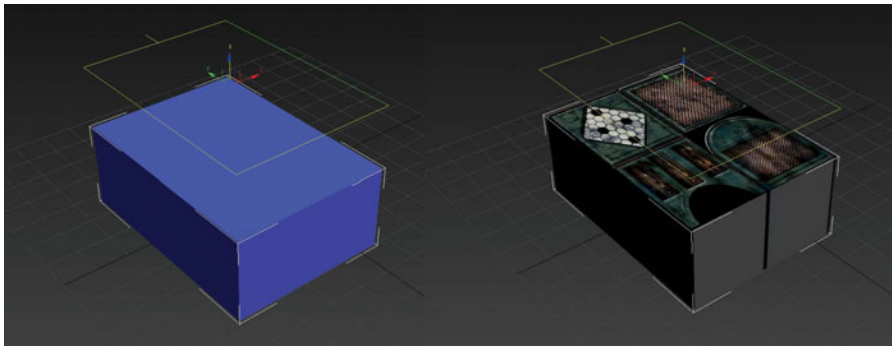


FIGURE 4.61 Apply 2D maps to each polygon.

1. Open the navmesh file you've been using with Level map master.tiff, and hide all the layers except for the red navmesh layer.
2. Also open the texture atlas.png file, and copy the whole file.
3. Return to the navmesh layer, and use the marquee tool to select the red area.
4. Choose Edit > Paste Special > Paste Into. You should now have a version of the texture atlas floating around in the red area. Move the texture atlas layer to the area you identified as a number 1 zone. Move and scale the atlas in area 1 to fit the space (**FIGURE 4.62**). Don't worry about spillover from the rest of the atlas because you will cover it up soon.

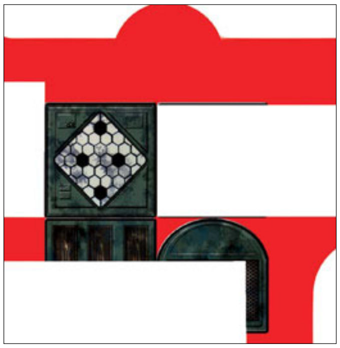


FIGURE 4.62 Move and scale the atlas in area 1 to fit the space.

What you're trying to do here is use the four images on the atlas to fill out portions of the red area. Generally, you would do this without any nonlinear scaling because the stretching is an obvious tell that a texture is being applied.

5. Repeat this technique until the entire red area is filled (**FIGURE 4.63**). Refer to the number map you created earlier, if you need to. Do not worry about strictly adhering to the number map, as it was more of a guide than anything else. Try to introduce variation into the pattern if you can.

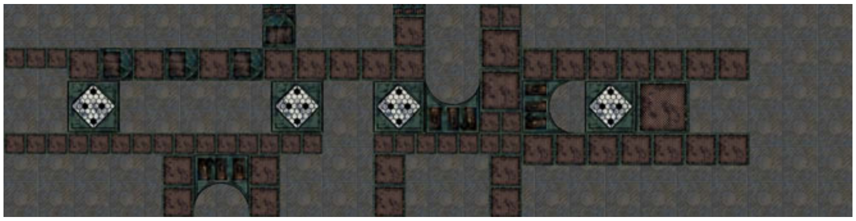


FIGURE 4.63 Fill the entire red area.

A top-down view of the ground map never looks as good as a perspective view. If you were working in 3D software, you would just apply the tile to the meshes and look through the virtual camera to check your work. However, in Photoshop, while it isn't that easy, it isn't impossible. You just need to work some Photoshop magic to see what your ground maps will look like in game view.

6. Collapse your many, many, layers to a single layer.
7. Create a new layer, and fill it with a black color. Move this layer below the ground map layer.
8. Choose Edit > Transform > Distort. Drag the upper-right control handle to the lower-left corner. Move the controller from the lower-right side to the lower-left side. Move the two left-hand side controllers to the upper middle of the screen. This should give you a nice, perspective-style view, as shown in **FIGURE 4.64**. Basically, you are flipping it upside down.

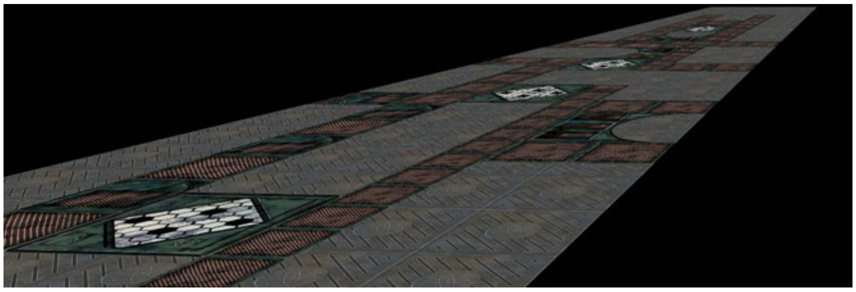


FIGURE 4.64 Games-eye view of the ground map.

You can use this technique to test your building sketches as well. Just adjust the ground plane image with the Distort tool to match the direction of the diametric grid (**FIGURE 4.65**).

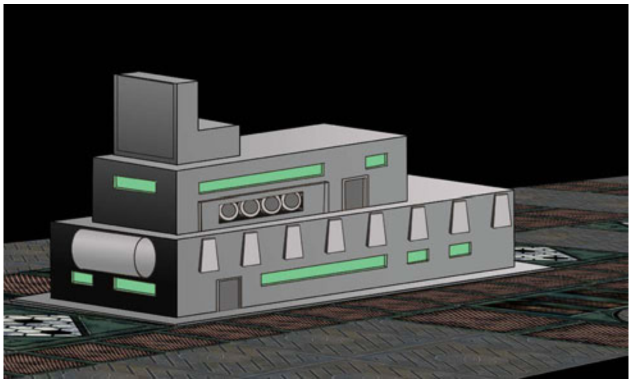


FIGURE 4.65 Building and grid in game view.

MAKING PROPS, PICKUPS, AND OTHER STUFF LYING AROUND

Prop creation is often a misunderstood task at a game company. Because it is not part of the main character set or the main-level layout, the task of creating the props is often given a lower priority. But just like the main characters and the main level, props play a huge part in the game.

A prop in the game world can refer to many kinds of objects. A crate in a warehouse is a very obvious example of a prop, but so is that burning train right behind the crate, and that crowbar on the ground, and that dead guy those zombies are eating. A good definition for a prop is anything that supports the scene but is not part of the level layout or character set. This is not to say that characters can't use props, but they are not generally part of the actual character mesh.

To further understand props, you will organize them into categories. These are Common, Dynamic, Supportive, and Interactive.

- **Common props** are all the stuff you see lying around a level. They generally do not move and you cannot pick them up. They are the pipes, vents, desks, shelves, fallen columns, and skeletons that you pass while playing your game. They are sometimes called static props, and in more than a few engines, they have their own classification.
- **Dynamic props** are props that you can interact with. This includes items such as exploding barrels, levers, switches, buckets, and bowling balls. Dynamic props generally have attributes that cause them to move or react in some way. Sometimes this means that a physics modifier is attached to the prop, which causes said prop to react with a gravity-like behavior when you hit or shoot it. Other times, as with a dynamite barrel, an explosion is triggered that inflicts damage

across a particular radius. Dynamic props can also simply change states, such as a light coming on or a switch going from up to down. Any prop that moves or does something is a dynamic prop.

- **Supportive props** are also known in the game world as *pickups*. These are items to watch for while playing a game. They can change the way your characters perform. They can add or take away health, give ammunition, change your weapon, and do any number of other things. The physical properties of a pickup can vary a great deal. They tend to look like suitcases or magic vials. The pickup is very akin to the doobers we spoke about previously in the book. It is more a quickly recognized icon than a fully rendered version of an object. Bullets, for instance, are pickups. In a game, you do not pick up individual bullets, but you often find a clip or a case of bullets, or find health icons. How would you pick up health, and what would it even look like? In most games, it looks like a first aid kit, of course, or a beaker, or some liquid that glows. When you create a pickup, keep in mind that the player is looking for something that symbolizes their idea of an item or concept.
- **Interactive props** are items that the in-game character can interact with, such as guns, cars, jet bikes, elevators, and zip lines. These props often aid the player in completing a task in the level. They are generally complex and animated, and rival the main character in poly count. For this reason, few interactive props are scattered around, or if they are, they use the same 3D model with different textures.

And Then There Were Crates

One prop reigns supreme as the king of all props. The first and most often thought-of prop in the game world is the crate. Crates are very popular for one good reason: they have, by the very nature of their shape, an extremely low poly count. Mix that with a little visual identification memory for most players, and you get a low-poly object that is easily recognized and extremely believable, and one that you can build with minimal effort. “Ah,” you say, “that is why they are in every game ever made.” Yes, but there are other reasons as well.

Think of crates as the spackle of the game world. They can be used to hide seams in the level layout. They can be used to direct your eye in a certain direction; they can be used to hide ancient hog demons until they are ready to attack. Suffice it to say, they get used a lot.

Crates can also be used to set the mood of a level. A room full of pink crates with a cute bunny on the side will feel much different than a room full of black crates with a toxic waste symbol. All it takes to change that “crate mood” is a new decal on the side or a different color. The best part is that you only have to make one model, and you can just change it as often as you’d like.

EXERCISE 8: CREATING PROP DESIGNS AND TEXTURES

You are going to design four crates, each with its own special feel, but all using the same geometry. Essentially, you are making four textures for one crate object.

Creating the Roughs of Four Crates

The game designer has identified the four crates he wants. He needs a wood crate with the word MUNITIONS on it; a space-aged crate with a window to some glowing matter inside; an old, rusted metal crate with rivets; and for some reason, the aforementioned pink crate with a cute bunny emblem. He would like to see roughs before you go to final, so you should start with those.

1. Open the diametric grid file. This may seem like overkill for a few crates, but it will easily give you perfect perspective, and if you can get that for free, you should take it.
2. Create a new layer above your grid layer, and then create a bounding box using the Line tool with a **red** color and no stroke (**FIGURE 4.66**).
3. Place a new layer above the red layer, and name it *crate*. Change your line color to **black**, and create a shape from the bounding volume (red lines). Collect your shapes into a single shape layer by choosing Merge Shapes (**FIGURE 4.67**).

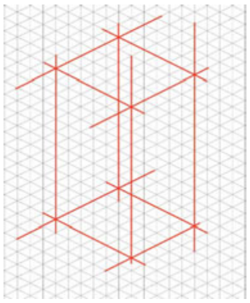


FIGURE 4.66 Create the bounding volume for constructing a crate.

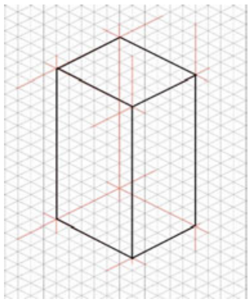


FIGURE 4.67 Create a main shape for the crate.

4. Now you are going to create a relief in the crate shape to give it a bit more depth (**FIGURE 4.68**). Using the grid, go 1 unit in and create a rectangle on both sides and on the top of the crate.
5. Add a bit of thickness to those shapes by creating an internal lip (**FIGURE 4.69**).
6. Finally, collapse all the layers to a single layer. Create a new layer, and merge the shape layer down. Rename it *crate*, if you wish to. Then use the Fill tool to add some shading, lighter on top and darker on the side with the largest face being the neutral tone (**FIGURE 4.70**).

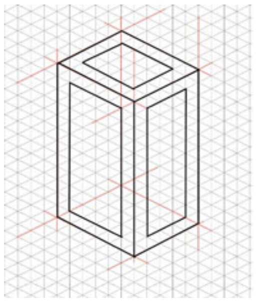


FIGURE 4.68 Create a rectangle on both sides.

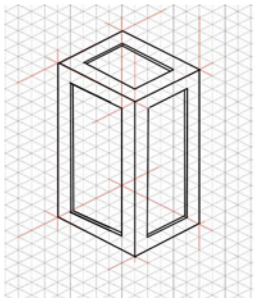


FIGURE 4.69 Add volume by adding an internal lip to the crate.

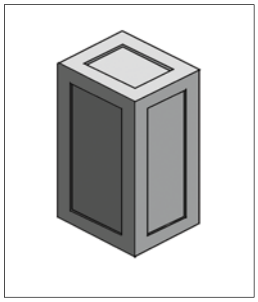


FIGURE 4.70 Add some shading to the crate.

7. Duplicate the new crate layer four times and spread them out. These will be four roughs once you add colors and detail.
8. Using the marquee selection tool, select the first crate interior by selecting the space around the crate, and from the Selection down-down menu, choose Select > Inverse.
9. Create a new layer and fill the marquee selection with a **brownish** ramp. Change the layer style from Normal to **Multiply**. Repeat this technique for the rest of the crates, giving each one its own individual color (**FIGURE 4.71**).

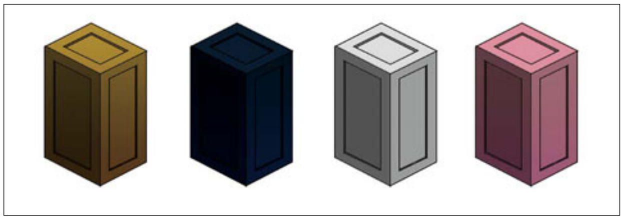


FIGURE 4.71 Give each crate its own individual color.

10. When you have the main colors sorted out, return to the first crate, and create a new layer called *crate_wood*.

Building a Wooden Crate

Put aside the hammer and nails. The only shop you'll be using to hammer out this wooden crate is Photoshop.

1. Choose the brush tool with a **black** color and begin to draw the planks of the crate. Vary the width of the line to simulate uneven placement of the planks. Also draw diagonal lines across the corners to simulate the miter cut of the frame.
2. Create a new layer, and call it *nails*. Turn on Bevel & Emboss and Drop Shadow, then select the brush tool with a hard brush and a **gray** color. In the corners of the crate, draw two circles on either side of the miter cut (**FIGURE 4.72**). Repeat for all corners.

3. Create a new layer under the crate_wood layer, and call it *rings*. Choose a brush tool with a **brown** color just a bit darker than the brown on the crate. Using a small brush, draw the natural lines that occur in woodgrain (**FIGURE 4.73**). Make sure they run lengthwise with the board because most wood is cut that way.
4. Using the Text tool, add the MUNITIONS label, and then rotate and scale it to fit. Create a new layer below the text layer and merge it down to make it editable. Using the Erase tool at **30%** opacity and a Scatter brush, distress the text a bit.
5. Create a new layer at the top of the stack, and name it *shadow*. Choose the brush tool with a soft brush. Change the brush color to **black** and set Opacity to **20%**. Open the shadows around the inner parts of the crate to help sell the 3D dimensionality (**FIGURE 4.74**).

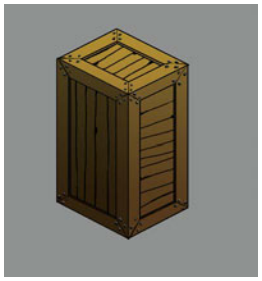


FIGURE 4.72 Turn on Bevel & Emboss and Drop Shadow.

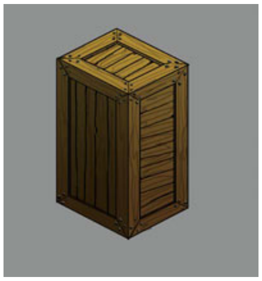


FIGURE 4.73 Draw the natural lines that occur in woodgrain.

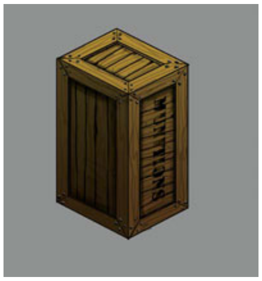


FIGURE 4.74 Add shadow and color to finish the wooden crate.

Building a Space Crate

In space, no one can hear you build a crate.

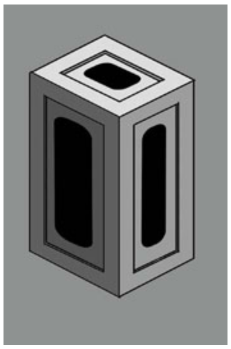


FIGURE 4.75 Create another pill shape.

NOTE The blue color is turned off here so you can better see the shapes.

1. Move over to the second crate, colored blue. Use the Rounded Rectangle tool to create a pill shape (**FIGURE 4.75**).
2. Using the Distort tool, adjust the perspective to match that of the crate's left front panel. Duplicate that layer, and choose Edit > Transform > Flip Horizontal. Move the layer into place on the crate's right side.
3. Create another pill shape, and use the same techniques to place it on the top panel.
4. Create a new layer, and merge all three pill shapes and the blank layer to make the pill shapes editable. Using the Magic Wand, select the inside of the three shapes.
5. Create a new layer named *space matter*. Choose the brush tool with a soft brush and set Opacity to **60%**. Select a **bright green** color and paint in the selected area.

6. Choose a scatter brush in a **darker green** color, and stipple that around to create particle-looking matter. Continue doing this until you have found a space matter that works for you. Then, in blending options, turn on Bevel & Emboss and Outer Glow (**FIGURE 4.76**). Set Bevel & Emboss to go **Down** instead of Up, and change the glow color to match the **lighter green** color.
7. Choose the Line tool, change its color to **bright green** with a **5-pt.** size and no stroke. Draw a geometric pattern on the case that looks like a circuit board. Work your way around the crate until you are happy with the look. Create a new layer and merge all the line layers with the new layer. Name the layer *lines*, and change the layer mode from Normal to **Color Dodge**.

   What would a space crate be without some control buttons?
8. Using the rectangular shape tool, create four or five different-colored squares. Line them up vertically. Collapse them into one layer, and with the Distort tool, fit them into the space at the top of the crate (**FIGURE 4.77**). Make sure the perspective works. Now select Bevel & Emboss and Drop Shadow.

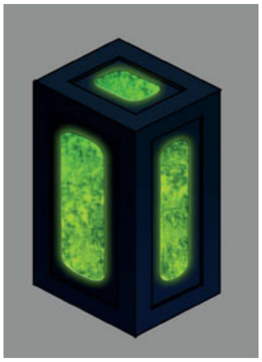


FIGURE 4.76 In blending options, turn on Bevel & Emboss.

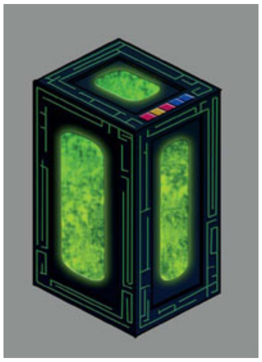


FIGURE 4.77 Add control buttons to finish the crate.

Restoring a Rusted Metal Crate

The rusted crate is a bit of a different story because its gray texture is just the beginning. To make a rust texture, you need to incorporate a lot of transparent layers all working together (**FIGURE 4.78**). You are trying to replicate years of neglect.

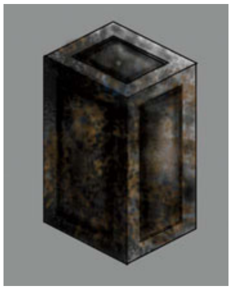


FIGURE 4.78 Produce as much visual noise as possible using a reddish-brown color and a scatter pattern.

1. With the Magic Wand select around the crate, and then choose Select Invert to select the crate.
2. Create a new layer (you will do this a lot with this crate.) Choose the brush tool with a scatter pattern and a **reddish-brown** color. On the same layer, repeat the technique with a **black** color, and then with a **gray** color. Keep choosing different brushes and sizes. The goal is to produce as much visual noise as possible. When you are done, set Opacity to **50%**.

3. Create a new layer and repeat the process. This time start with a **dark orange** color and work in a **red** color. Choose Filter > Noise, and adjust the noise level to resemble rust particles.
4. Create a new layer, and in the blending options, turn on Bevel & Emboss and Drop Shadow. Choose the brush tool with a scatter brush and a **gray** color. Paint in some sections to create the look of chipped paint.
5. Create another layer, and using the brush tool with a scatter, paint some dirt on top of the painting you just did. You may need to place several layers of paint and dirt to get the look right. Don't worry about the number of layers you have to use; they are free.
6. Create a new layer, turn on Bevel & Emboss and Drop Shadow; then repeat the process you used to create nails for the wooden crate, except call this layer *rivets*, as you are now working with a metal object. Pepper them around in patterns (**FIGURE 4.79**).
7. Choose the Custom Shape tool, and select a foreboding emblem. As you can see in **FIGURE 4.80**, we used the classic radioactive symbol. Create a new layer, merge the symbol down, and use the Eraser tool to distress it a bit.

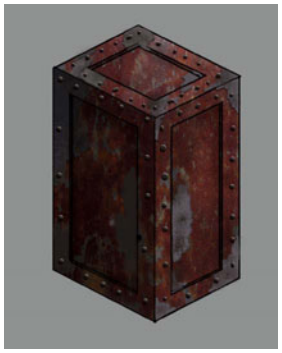


FIGURE 4.79 Create nails.

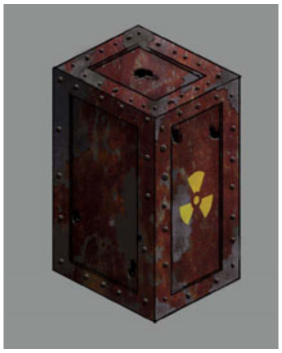


FIGURE 4.80 Create erosion on the crate.

8. One more thing: Create a new layer and turn on Bevel & Emboss. Make the bevel go down. Select a scatter brush with a **black** color. Use this brush to poke holes in the crate. That's just what you want to see when running across a crate with a nuke symbol: a few holes in the side.

Prepping the Fluffy Pink Bunny Crate

The pink crate is the easiest of all your crates (**FIGURE 4.81**), and that's because you only need to do a few things to it. The pink color is enough to sell the image.

1. Create a new layer called *dirt*. Choose the brush tool with a scatter brush. Select a **pink** color that is a bit darker than the pink on the crate. Pepper the darker color around to add some visual interest.

2. Create a new layer, and call it *bunny logo*. Create the cutest bunny logo you can, and then duplicate the layer. You always want to keep the unaltered original in case you need to change it in the future.
3. Use the Distort tool or the Perspective tool to place the bunny logo on the left side of the crate.
4. Duplicate the original logo, and using the same technique, place it on the top panel of the crate (**FIGURE 4.82**).
5. Create a new layer, and just as you did before, use a soft brush with a **black** color and an opacity of **30%** to darken the shadows around the inside edges (**FIGURE 4.83**).

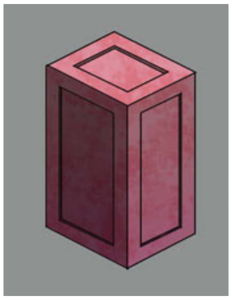


FIGURE 4.81 Create the cutest bunny logo you can.

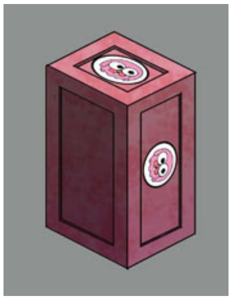


FIGURE 4.82 Duplicate the original logo.

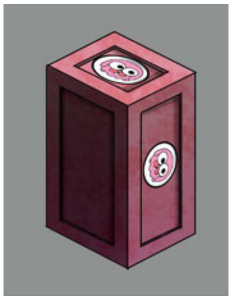


FIGURE 4.83 Use a soft brush to darken the shadows.

6. Save your file and you are done.

As you can see in **FIGURE 4.84**, all of the crates match the specs from the designer. You have wood for the wooden crate, space matter for the space crate, rust for the metal crate, and adorable bunnies for the pink crate.

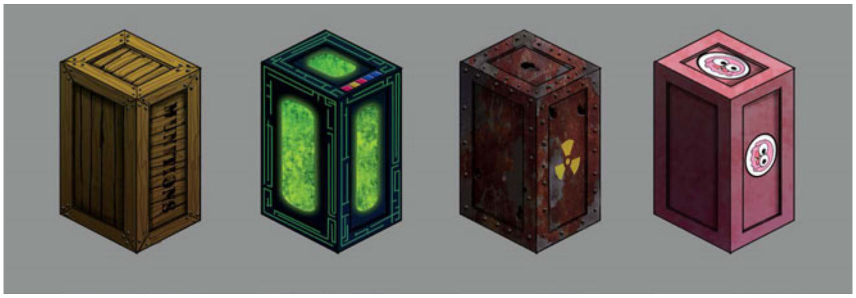


FIGURE 4.84 The four crates with the final touches.

These crates may seem a bit cartoony, but they get the point across. A modeler could take any one of these and create a 3D object with very little effort. Even if these crates were meant to be rendered realistically, your modeler would have a pretty good idea of what to include in the mesh and the texture.

CREATING A CRATE TEXTURE MAP

You will now create a texture map for one of these crates. This will bring together all the techniques you have covered in this chapter and give you a sneak preview into the next chapter.

Making Side Pieces

Just in case you didn't know, a texture is an image that is applied to a 3D object to give it color or tone. It can also be used as a vehicle to deliver lighting or displacement information. The texture you're making for this crate is set up just like the layout you did for the texture atlas with a bunch of information squeezed into a square. However, when the image is applied to only one object, it is just called a texture. You will make your texture in three parts. You could have gotten away just using two parts, but the game engine is going to calculate the area for that third part no matter what you do, so you might as well use it.

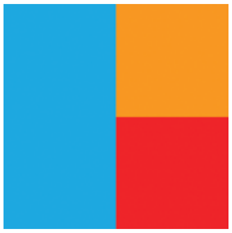


FIGURE 4.85 Create two 512x512 squares.

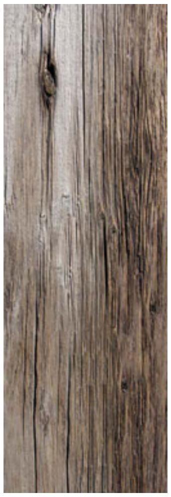


FIGURE 4.86 Use a photo as the base and work from there.

1. Open a new file that is 1024x1024, and name it *crate_wood1*.
2. Create two 512x512 squares, and stack them on top of each other on the right (**FIGURE 4.85**). Be very careful with your positioning. In a texture, every pixel counts, so you want to make it nice and tidy from the beginning.

   The larger part will be used for the four longer areas, and the two squares will be used for the end caps. In previous exercises, you illustrated your images to get the look you needed; but this prop requires a little more realism, so you will use a photo as the base and work from there (**FIGURE 4.86**). Actually, most 3D programs have premade wood textures that work really well, but in the interest of advancing your skills, you will make your own.
3. Find a picture of a piece of wood that looks like it's from an old crate. You are looking for an image of a board around 3 feet in length.
4. Import the image into your main file, and just like a carpenter, build a frame. Scale your wood picture thinner, if necessary, and slide it against the right wall. Duplicate the layer, and slide that up against the left wall. Duplicate that layer, and rotate it 90 degrees, and move it to the bottom. Duplicate that layer one more time, and move it to the top. You should now have a wood frame with a blue center (**FIGURE 4.87**).

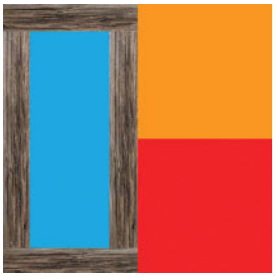


FIGURE 4.87 You should now have a wood frame with a blue center.

5. Because you are a good carpenter, you're going to give your frame a miter cut, which means that you cut the top and the bottom pieces at an angle (**FIGURE 4.88**). (You needn't cut the side pieces because it's just an image.) Select the bottom piece layer, and with the Polygonal Lasso tool, select an area from the bottom corner to the inside joint where the two pieces come together; then select around the outside edge of the board image. Close your loop, and cut out the selection. Repeat this on the other side, and with both pieces of the top.
6. This looks great, but it's not as realistic as it might be. Merge all three layers, and in blending options, select Bevel & Emboss and Drop Shadow.
7. Create a new layer called *nails*. In the blending options of this layer, turn on Bevel & Emboss and Drop Shadow. Now select a neutral gray color and hammer some nails into the corners of the wood frame (**FIGURE 4.89**). Or, as you say in Photoshop, select a brush tool with a hard edge and 100% opacity and flow, and put two dots on either side of the seams of the top and side pieces.
8. Create another new layer and call it *wood edges*. Select a brush tool with a sharp brush and set Opacity to **85** and Flow to **60**. Using a **black** color, work a deeper line into the section where the two pieces of board meet. You may also want to add some subtle highlights. While you are there, dirty up the nails a bit by scribbling on top of them with the black and the highlight color (**FIGURE 4.90**). Don't worry about accurately rendering your nails. The lighting in the engine and the resolution constraints will fill in the rest.

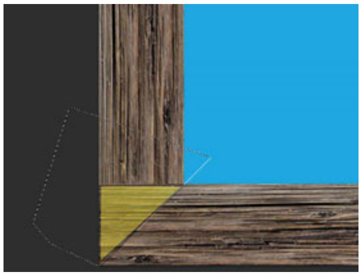


FIGURE 4.88 Give your frame a miter cut.

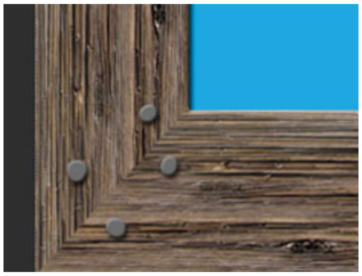


FIGURE 4.89 Hammer some nails into the corners of the wood frame.

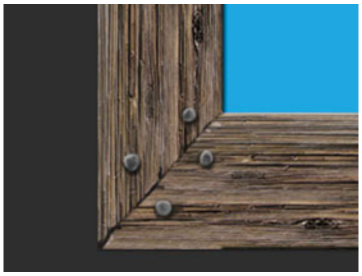


FIGURE 4.90 Dirty up the nails a bit by scribbling on top of them.

9. Import your wood image and rotate it 90 degrees. Move the new layer under the wood-frame layer. Duplicate it four or five times. If you need to scale it a bit to fit, that is fine. It is important to maintain the spaces between the boards. You need to see blue between each one. In the blending options, select Bevel & Emboss.
10. Now return to the blending properties and adjust the drop shadow to appear a little stronger (**FIGURE 4.91**). Set Distance to **28**, Spread to **3**, and Size to **8**.
11. Now for some detail. Select the Text tool, and type **MUNITIONS** as you did previously. Rotate the text 90 degrees clockwise, and place it in the middle of the wood frame. Make sure that the text layer is the top layer.

12. Create a new layer, and merge the text layer down to make it a raster image and, therefore, editable. Using the Eraser tool with a hard-edged brush, erase the portions of text that are between the boards. Now choose a scatter brush. Set Opacity to **33** and erase portions of the text to give it a worn look that matches the wood (**FIGURE 4.92**).
13. Save your file.

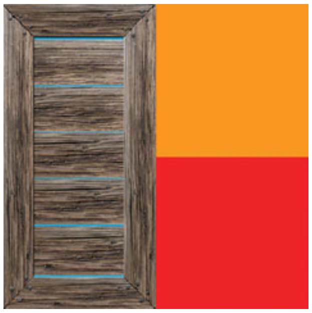


FIGURE 4.91 Maintain the spaces between the boards.

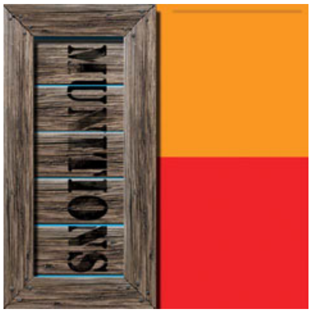


FIGURE 4.92 Give the boards a worn look.

Texturing Cap Pieces

Now that you have created the sides of the crate, the cap sections should be easy. You have all the assets that you need. You just have to do a little creative cutting and pasting.

1. Select and duplicate the wood frame, nails, and wood edges layers. Merge the layers, and then slide them to the other side of the screen (**FIGURE 4.93**).
2. Select and copy the lower half of the new wood frame layer, and then paste it in place. Translate it up until the bottom of the frame covers the orange background area.
3. Choose the Eraser tool at **33%** opacity with a soft-edge brush, and blend in the two pieces (**FIGURE 4.94**).

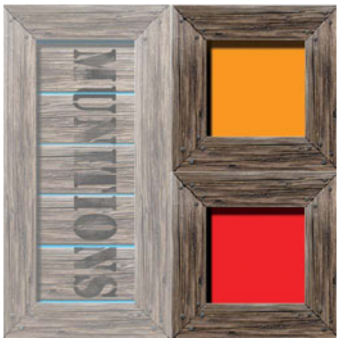


FIGURE 4.93 Slide the merged layers to the other side of the screen.

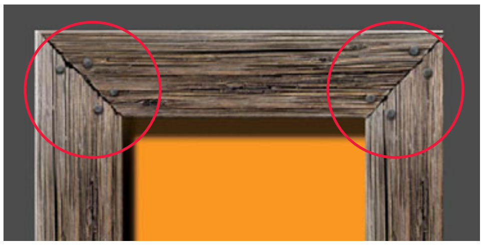


FIGURE 4.94 Blend in the two pieces.

4. Merge the two layers, and using a marquee selection tool, select the entire square frame, and then duplicate and paste it. Move it down to the lower position, covering the red area (**FIGURE 4.95**).
5. Bring in your wood again and rotate it 90 degrees. Move it into place, and cut off or erase the extra portions. Duplicate the layer three or four times, and fill the two frames just as you did previously.

   You have added a decal of some bullets and a break in the wood with some bullets showing through. The break was made by erasing the wood layer a little bit, and then adding an image of bullets under the wood layer (**FIGURE 4.96**).

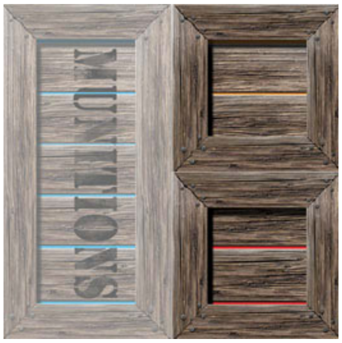


FIGURE 4.95 Move the square frame down to the lower position, covering the red area.

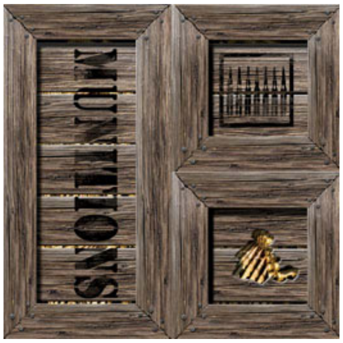


FIGURE 4.96 Add a decal of some bullets to the crate.

Because you started from a real-life object, the texture has a far greater chance of looking like a real object in the game world (**FIGURE 4.97**). Other factors will need to be addressed to fully sell it as a realistic object, but you will learn about those techniques in Chapter 5.

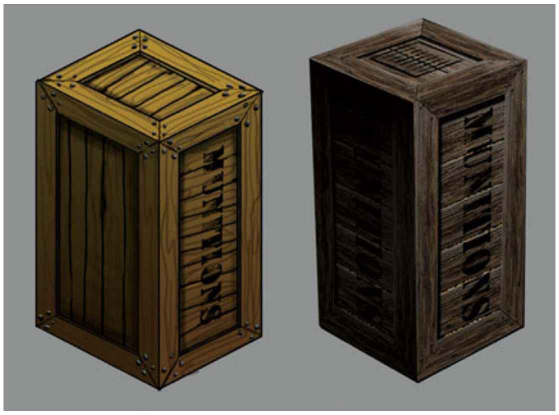


FIGURE 4.97 Using an image of an actual object results in a more realistic-looking crate.

CREATING FX FOR THE MOBILE SPACE

Creating *fx*, or effects, for use in the mobile space is a bit different than in the social or the console worlds. As you've done throughout this chapter, you will need to make effects that will work within a mobile game engine, but aren't so memory heavy that they stop the game. This is where art and tech meet. Most of these methods require a mixture of 2D and 3D work from the artist, followed by implementation by an engineer.

Using 3D Effects

NOTE A particle system in a 3D package replicates the generation of multiple (usually small) items.

3D effects replicate an event using the tools that exist in a 3D program or engine. They may consist of any number of 3D functions played together. For instance, an explosion effect might have a particle system running for the debris, another particle system running for the expanding light particles, and another for the smoke. All three of these particle systems coming together make what looks just like something blew up.

Particle systems tend to consist of an emitter, which acts as a nozzle that distributes the particles in a certain direction. In most 3D packages, it is represented by a square with a pointy bit in the middle.

The second piece of the particle system is a small bit of geometry that is replicated and given physical properties such as weight. It can also be textured and have lighting applied to it. 3D software packages usually include several premade shapes to choose from that have been proven to be low-poly-count friendly. But you can also substitute your own if you like.

The last element is the force applied to the particle, which includes parameters such as speed, strength, and turbulence.

Most 3D software and game engines have premade effects to replicate smoke, fire, explosions, and water. These effects are open, so if you wanted to alter the color of the fire or produce a bigger explosion, you could drag some sliders to make those changes.

Starting off with a premade particle effect is always easier than creating one from scratch. You can purchase a ton of very inexpensive effects libraries for this purpose. If you consider how long it could take you to make an effect on your own, premade effects are often cheaper.

Using 2D Effects

2D effects, often called *sprites*, are simply an animated sequence of images that are played when an effect event is triggered. This means you can illustrate them, or use a photo, or even create them in 3D and then render them as a movie sequence.

How does a 2D projected image sequence hold up in a 3D world? Not too bad. It is possible to always align the effect to the camera view so that no matter how you rotate around the object in the game, the effect is always seen from the perspective you intended. Do 2D effects look as good as a 3D-rendered particle system? No, they do not always look so great, but if your choice is a 2D effect or nothing (as is often the case in the memory-stingy mobile world), you take the 2D effect.

The problem is that 3D effects are just too memory expensive to use all over the place, so they are generally saved for the more important moments such as killing the end boss. The hundreds of disposable bad guys you shoot on your way through the level just get the 2D treatment.

EXERCISE 9: CREATING A 2D SPRITE-BASED EFFECT

You're going to create a 2D explosion effect to show your robot's gunfire hitting an enemy. It will be a 16-frame sequence ready to bring into Adobe After Effects, Flash, or straight into the game engine.

A sprite sheet is a single image containing multiple frames of an animation sequence that a game uses to display as an effect. It usually has a lower memory cost and is a bit old school but is still very commonly used.

Setting Up Your Page

Setting up a page for a *sprite sheet* is easiest if you know ahead of time how many frames you would like to make your animation. Because the images are read by an engine, registration is very important. If you are even a few pixels off with some effects, you can get a jitter. So, if you know you need 16 frames of animation, that should also tell you to create a file with a size divisible by 16. It really doesn't have to be of that, but it makes the entire process much easier for you.

1. Create a new file in Photoshop, and make it 8 inches x 8 inches at 72 dpi.
2. Turn on the rulers. Select the Line tool with a **black** color, and mark off every 2 inches vertically and horizontally. This will produce a grid with perfect 2x2 boxes.
3. Select the little blue guides and position them on the 1-inch marks, both vertically and horizontally (**FIGURE 4.98**). This identifies the exact centers of those boxes.

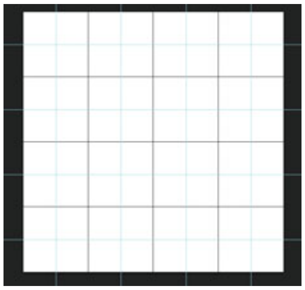


FIGURE 4.98 Mark off every 2 inches vertically and horizontally.

4. Using the Magic Wand tool, select the first box. Create a layer, and using the Fill tool, add a radial gradation. Make sure it is white in the middle and black on the outside edges (**FIGURE 4.99**). Do not let the white area travel outside the square area.

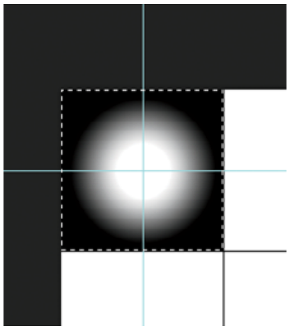


FIGURE 4.99 Make sure radial gradation is white in the middle and black on the outside edges.

5. In the grid layer, use the Magic Wand to select the square just below the one you were just in. Create a new layer, and use the same technique to create a radial gradation, but make it a bit smaller.
6. Repeat this technique for the other two vertical grid spaces (**FIGURE 4.100**). You should have a small radial gradation (fuzzy circle) at the bottom and three more, each a bit larger going up the line.
7. Now duplicate all four layers, and move the new layers 1 grid space over. Repeat this two more times until the grid is filled (**FIGURE 4.101**).

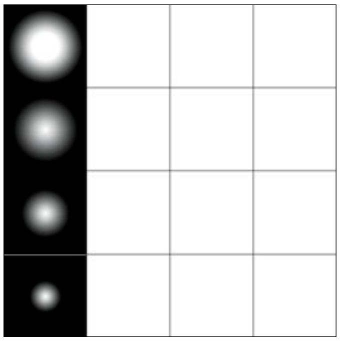


FIGURE 4.100 Repeat this technique for the other two vertical grid spaces.

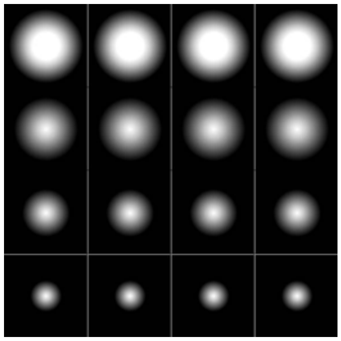


FIGURE 4.101 Duplicate the layers and move them 1 grid space over.

This will be your animation sequence. As you can see, every four frames the explosion gets smaller. You will also help it along in a second, but first let's add a little cloud texture and some color (**FIGURE 4.102**).

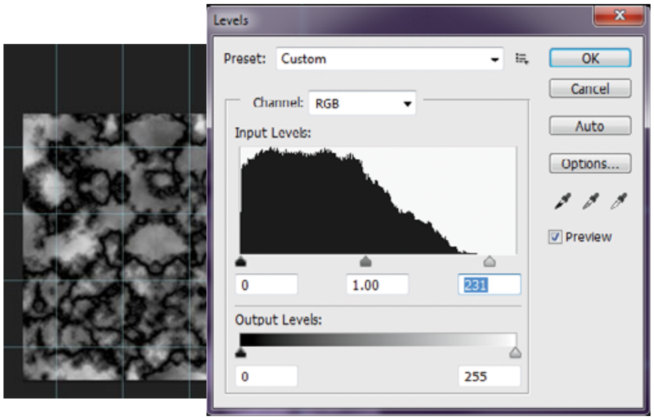


FIGURE 4.102 Add a little cloud texture.

8. Choose Filter > Render > Cloud to add a smoky texture in the box. This looks great, but it is gray and square, which will not read well in the game. Choose Image > Adjust > Levels and increase the contrast a little.
9. Call this layer *cloud*, and change the layer style from Normal to **Multiply**.
10. At the bottom of the layer window, click the “Create new fill or adjustment layer” button, a little circle icon that is half white and half black. From the popup menu, choose Gradient Map. Move the layer to the top of the stack. Change the layer style from Normal to **Color**.
11. Click the gradient bar. Change the first stop to **black** and the stop on the other side to a **pale yellow**. Slide the pale yellow stop to the midway point on the line. Click just under the bar to add a new stop between the black and the pale yellow stops. Change the new stop to a **bright red** and slide it a little toward the black stop (**FIGURE 4.103**). Feel free to experiment with the color until you have something you like.

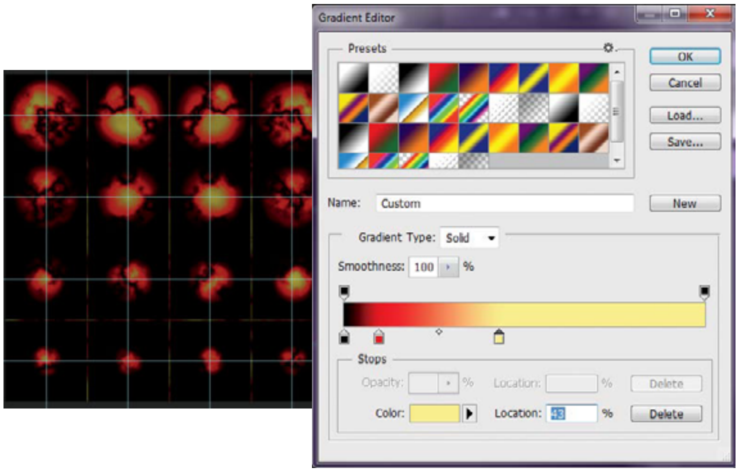


FIGURE 4.103 Add color to the cloud texture.

12. Create a new layer at the top of the stack. Choose a brush tool with a scatter brush. Select a **black** color and paint out some of the explosion in the far-left frames (**FIGURE 4.104**). You do this because the explosion should dissipate as it is being read by the engine from left to right. You can also use this layer to add more color, or lightning, or whatever else you might need.

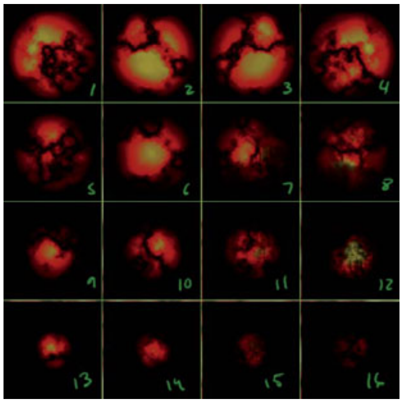


FIGURE 4.104 The final sprite sheet.

13. Save your file.

    You have numbered the frames so that you can see the sequence they will play in. Most sprite sheets are read left to right, top to bottom. On a game production, this sheet would be given to the engineer, who would apply code to it to play the frames at a determined rate, or backwards, or however you wanted them to play.

As you can see, a 2D effect may lack a bit of the 3D finesse, but for something that is on the screen for only 16 frames, or 0.5 seconds, it will look pretty good and be far less greedy with the memory.

CHAPTER 4 WRAP-UP

In this chapter, you covered a great deal of the specific pipelines when creating assets for a mobile 3D robot shooter game. You learned why specific assets are made the way they are, and how to build art with an eye toward saving memory.

As the mobile world progresses, the quality of the art assets you can make for the space will also increase. It is fortunate that the need for faster, better-looking graphics drives the mobile market because with each new advance in mobile technology, you'll get a few more pixels to play with.

INDEX