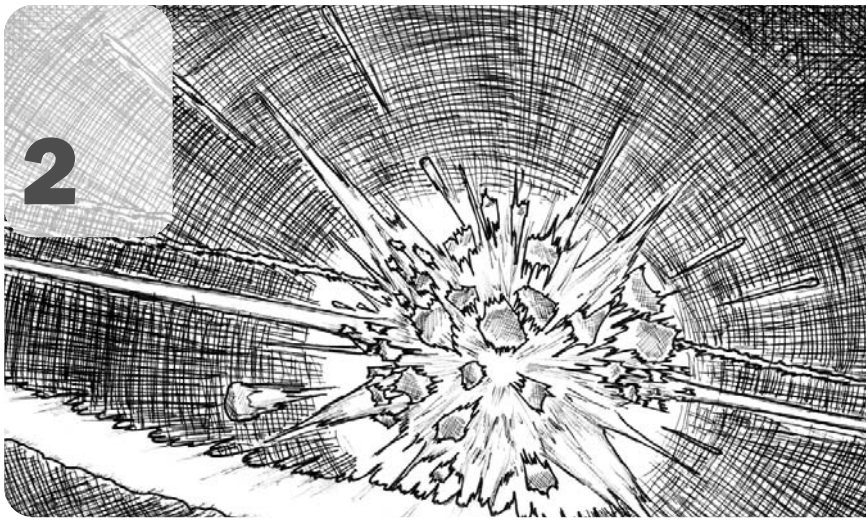
Chapter 2



The Process of Game Development

At this point, you should have a good idea of what components and assets you need to produce for your game or project. So now you might be asking "How do I go about acquiring all this new content?" In that respect, you're lucky because this book is designed to tell you how to construct each major asset to complete your game. However, even after you have learned how to produce all your material, you might still ask yourself the infinitely more difficult question "Where do I begin?"

This chapter presents a basic schedule and planning system for game content development. You are more than welcome to use it as a framework for your development process, or you can design your own based on your specific needs and your project's requirements. Much of what's presented in this chapter is designed for first-time developers who might need some guidance in developing their game. Everyone has a different workflow, so be sure to use the methods that work best for you, especially if you already have experience in the field of game development.

Building Your Foundation: Before Production

You need to consider many factors before you jump into game development. If you're a gamer, try to think of the number of people you've heard say "Hey, I've got a great idea for a game!" Sometimes, the ideas really are good. You might have heard one or two that would make successful games. Perhaps you've even had such an idea yourself.

Just thinking of a good idea, however, is not enough in the game development world. The best ideas out there will probably never see the light of day. What you need is a detailed plan of action in which you've considered all the possibilities and pitfalls you might encounter throughout the course of the project. Creating this plan can seem overwhelming at first, but with enough time, effort, and planning, just about anything can be accomplished in the game world.

In the following sections, you get a taste of some guidelines you need to keep in mind to turn a good idea for a game or project into a reality. The sections in this chapter are designed to guide inexperienced game developers and to paint a complete picture of the game development process.

Developing Your Idea

You might want to develop your idea after you have formed a team, but usually it's better to have a solid idea and plan of attack in place when you approach prospective team members. Make sure you think your idea through to its end. The more you know about your project, the easier it is to select your team. Take into account the volume of assets you need to create versus the timeframe in which you need to finish them. This factor helps you decide how large your team needs to be. If your plan is a small-scale one that doesn't require a team, getting insight and feedback on your idea from others who understand games or game modding is still a good idea.

Your idea should be as original as possible to set it apart from other games in the genre. The game design world is a competitive environment, where new games always try to outdo the old. Game graphics have become more sophisticated, but this factor isn't always enough to attract customers. You have to come up with a way to make the game fun and playable, regardless of its graphics. Try to come up with something that hasn't been done before in a game, or at least put a new spin on a popular game element. The following example has been divided into three sections: the story, the player's perspective, and the pitch.

The Story:

Many games have some sort of plot or story to help push the action along. Even though the game doesn't yet exist, a story is an important part of your idea because it helps you determine the type of assets your game is going to need. The following paragraphs outline an opening story from a yet-to-be-developed game called Eternal Exodus: The Fall of the Creators.

Project:

Single-player first-person shooter/space flight game (some multiplayer elements)

Working Title:

Eternal Exodus: The Fall of the Creators

Plot Introduction:

It is the year 2359. Humans have destroyed their homeworld of Terra through overmining and pollution. They have since escaped to the stars and established numerous colonies across their native galaxy. They have scoured deep space in vain searching for planets as abundant in resources as their fallen home planet. As their industry devours whatever materials they find on their travels, the corporations driving those industries have become ultimately powerful, superseding or supplanting all governments and regulations. The most powerful of these corporations is the Lathius TransGalactic Corp., which holds a monopoly on all interspace travel.

From the decadence of society has risen a relatively small but quickly growing resistance movement known as Fidus Terrenus. This activist militia focuses on the belief that humanity is destined to be more than interstellar parasites and that what's happening to colonial planets is the highest atrocity. The Fidus Terrenus have been known to engage in anti-corporate activities, and most corporate governments have officially labeled them as terrorists.

During a skirmish between an FT fighter group and a Lathius deep-space gas mining fleet, a large disruption happened. Two small ships of unknown design entered the fray, quickly destroying all other craft in the engagement. One of the FT ships was able to send out a distress signal that included an incomplete scan of one of these ships.

FIGURE 2.1 Eternal Exodus: Fall of the Creators.

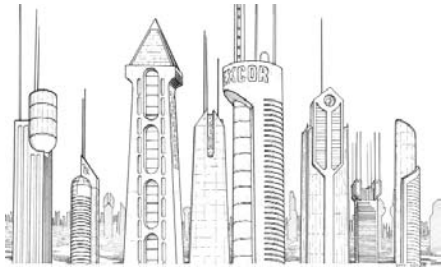


FIGURE 2.2 Corporate buildings, circa 2359.

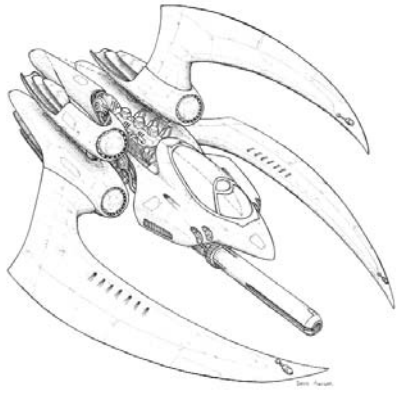


FIGURE 2.3 Lathius TransGalactic Interceptor ship.

FIGURE 2.4 Fidus Terrenus troops.

Days later, a massive signal was transmitted to virtually every Terran colony. The signal's force was so intense that only the most powerful receiving stations were able to monitor it without overloading. The transmission was translated via supercomputer within just a few hours, suggesting that the source of the transmission wanted it to be understood quickly.

The message was simple but enigmatic: "We come to put an end to that which we have begun."

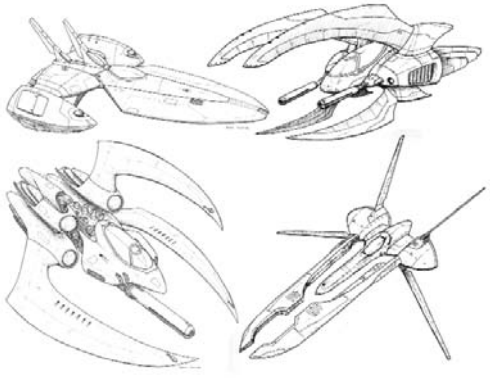


FIGURE 2.5 Ships from Lathius TransGalactic and Fidus Terrenus.

The Player's Perspective

After you have a story, drafting out how your players are exposed to the plot throughout gameplay is a good idea. This step can be a little difficult to nail down, so take your time. Also, don't be too surprised if your idea is forced to adapt to changes that arise as your game is developed. This draft doesn't need to include every single level and nuance of your game, but it should give your development staff an idea of how the game will progress. The following sections give you an overview of how players experience the story of this make-believe game.

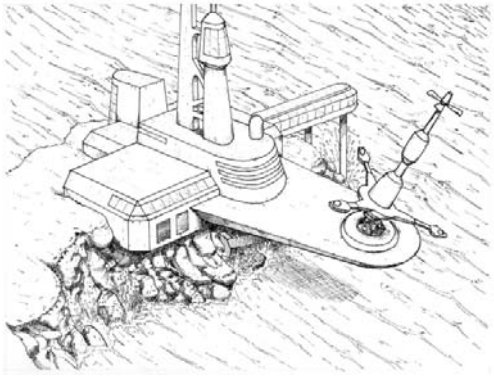


FIGURE 2.6 Receiving station where the original message was intercepted.

Overview of Game Plot from the Player's Perspective

The game begins with the player known as Jace Delaroix controlling the main character. Jace is a contracted escort fighter pilot and corporate bodyguard for some middle-to-upper management of Lathius TransGalactic. The message from the alien race, who call themselves "The Creators," was received months ago, and several of the outlying Terran colonies have been destroyed.

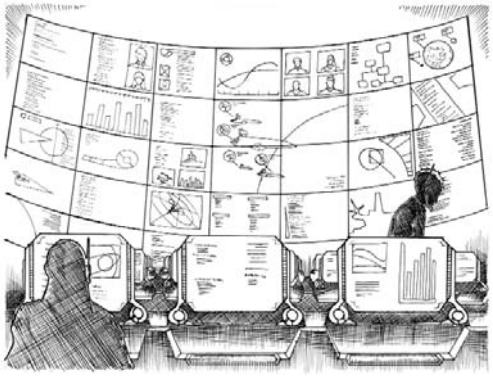


FIGURE 2.7 Control room of the receiving station.

The first few missions of the game include the player acting out the role of corporate bodyguard, in space and on foot. It's a time of turmoil, and smuggling, looting, and piracy are running rampant. Basically, players will have their hands full completing each task.

Eventually, Fidus Terrenus approaches Jace, asking him to abandon the corporation. The player can accept or refuse, and this decision determines the next several missions. If the player chooses to join Fidus Terrenus, Jace will have to complete such tasks as smuggling runs and skirmish assaults against corporate establishments. If, on the other hand, the player sticks with the corporation, Jace will combat the FT, trying to cut their supply lines and end their reign of terror over the colonies.

Either way, the Creators' threat eventually spreads to the player's location, and a massive battle ensues between The Creators and the planet's colonial inhabitants. Battle is on foot at first and then in space, as Jace tries to escape the planet and avoid the imminent doom brought on by the Creators. The only hyperspace-capable transport belongs to Fidus Terrenus, and Jace is forced to join their ranks if he wants to survive.

Jace is then taken to FT headquarters, where he finds out that a plan has been put into motion to retaliate against the Creators. At first, the plan requires capturing some of the Creators' ground forces, an exceptionally difficult task, as they have a tendency to self-destruct when defeated or overcome. Jace helps with these missions and eventually leads an assault to capture a Creator ship. During these missions, Creator technology is used to create new weapons and defenses.

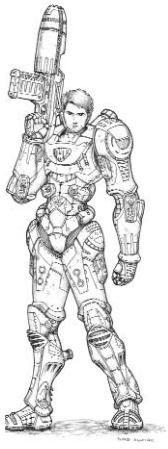


FIGURE 2.8 Player character Jace Delaroix.

FIGURE 2.9 Jace approached by Fidus Terrenus.

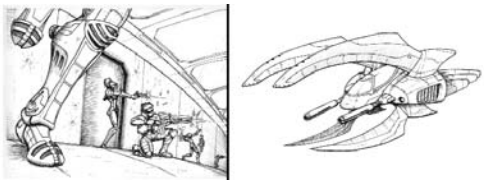


FIGURE 2.10 The battle begins, and the player is right in the middle of it.

The FT then learns that the Creators are a race of beings who found a rare kind of planet, which soon became Terra, and spawned mankind as an experiment, losing contact soon after developing a race now known as the Egyptians. During the next several thousand years, mankind achieved space travel, destroyed its homeworld, and began to spread like a virus across the stars. Mankind became the Creators' greatest failure, so the Creators have come to destroy mankind once and for all and put an end to their "experiment."

The FT also discovers that the Creators know of a planet similar to Terra, which is accessible only through an unstable wormhole. The remainder of the game consists of missions revolving around running from the Creators, getting the remnants of the FT to this planet, and destroying the wormhole, effectively removing the threat of extinction. The game ends with a sequence in which players find out that the Creators actually created the wormhole leading to this planet. This discovery suggests that the FT might have an encounter with them in the future, should the Creators choose to construct another wormhole.

The game has multiplayer aspects as well, including ground-based skirmish levels, such as FT forces fighting Lathius battle squads. Space fights with ships are also included, which could be team based or free-for-all. Finally, the multiplayer aspect could include a cooperative mode so that at least two players could move through the single-player campaign.

The Pitch:

If your game or project is more than just a hobby, the pitch is the most important part of your plan. To convince others to be involved in

FIGURE 2.11 Jace meets the leader of Fidus Terrenus, Shelia Sorenson.

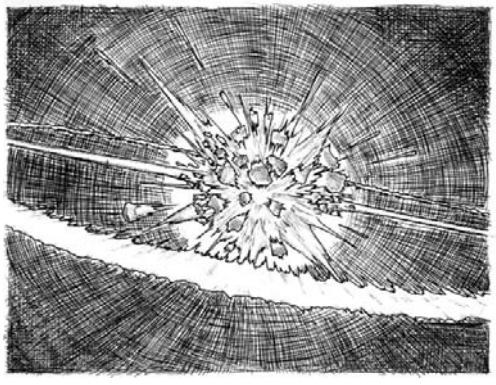


FIGURE 2.12 Planet exploding as a result of the Creators' attack.

FIGURE 2.13 Space shot of wormhole.

production, you need to persuade them (at least on some level) to participate. You need to form a sales pitch to help people get as fired up about your idea as you are. Developing a pitch is also a good time to figure out whether your idea is a good one. Keep in mind that if you can't sell someone on your game's concept, you probably won't be able to sell anyone the final result. Remember that not everyone shares your ideas of what's cool or exciting. Your pitch needs to present the project in such a way that most people can see why your idea is marketable.

Gathering the Team

Gathering your team is a more important step than many people realize. If you have ever tried to work with a group of people to accomplish a goal, you know exactly how important this step is. Working with a group to finish a project is often like trying to order a pizza that satisfies everyone. You have to take into account personal schedules, emergencies, obligations, and other day-to-day stuff that pops up for people who live in the real world. The following sections explain what factors you should consider when trying to get a team together.

Know Who You Need on Your Team

Take some time to figure out exactly what kind of jobs are necessary to complete your project. It's pointless to seek out team members until you have done this step. Remember, however, that when working with a small group, often team members can fill more than one role. On the other hand, if your project has complex tasks, such as extensive modeling, you might need to sign on more than one person to complete those tasks.

The following list describes some possible positions for members of your team:

- *Concept artist*—This person provides all the drawings and sketches needed for your project, such as characters, weapons, vehicles, and map layouts. For example, the earlier sections in this chapter about the story for Eternal Exodus included several pieces of what could be considered concept art. Naturally, concept artists need to exhibit exceptional skill at creating 2D art based on ideas and descriptions. Having a good design sense, too, is a tremendous help in enhancing the final look and feel of the game's elements.

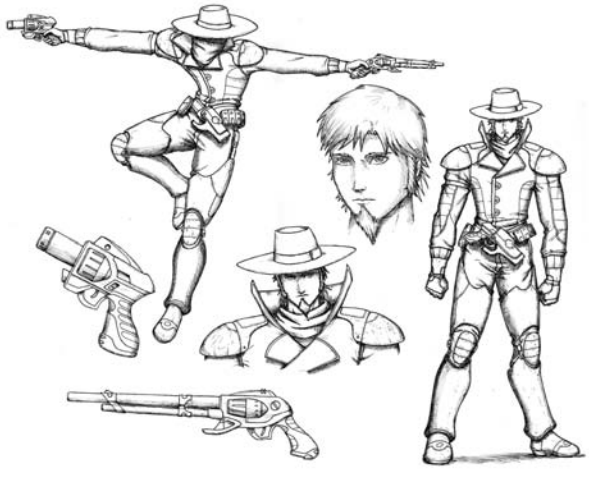


FIGURE 2.14 Character concept sketches.

- *Level designer*—You need someone on your team who knows how to design and create maps for your game or project. That's where the level designer comes in. This team member should have a thorough knowledge of UnrealEd as well as what can and cannot be done in a game map. Level designers need to have a good overall understanding of game design, such as what look is popular in today's games and what will look best for your specific idea. They also need a good aesthetic sense of interior design for placing lighting, decorating levels, and making certain that the overall look of each map is consistent throughout the game.
- *Modeler*—This team member converts the concept artist's 2D artwork into 3D assets that can be imported into your game. Typically, modelers' work needs to be relatively low in polygon count, depending on the amount of physical detail your project needs. Modelers need to be familiar with the polygonal modeling tools in your chosen 3D package and should be ready to work closely with concept artists to make sure their work follows the original designs. Modelers must also stay in close contact with level designers so that they're modeling all the necessary static meshes for each map.
- *Texture artist*—The tactile look of your game elements are left to the texture artist, who is the team member responsible for making sure your project's objects seem to be made of the correct materials and have appropriate colors and shading. Texture artists must have extensive knowledge of 2D computer art applications, such as Photoshop, and know where to find sources of real-world textures.

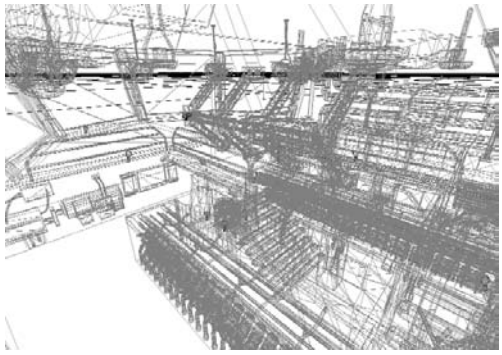


FIGURE 2.15 Wireframe image of a level.

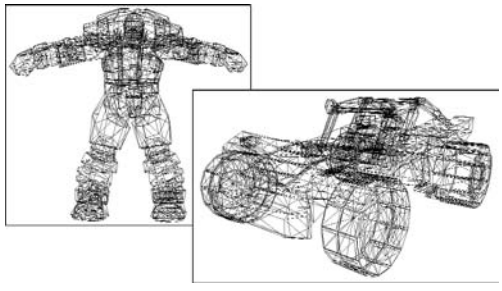


FIGURE 2.16 Vehicle and character wireframes.

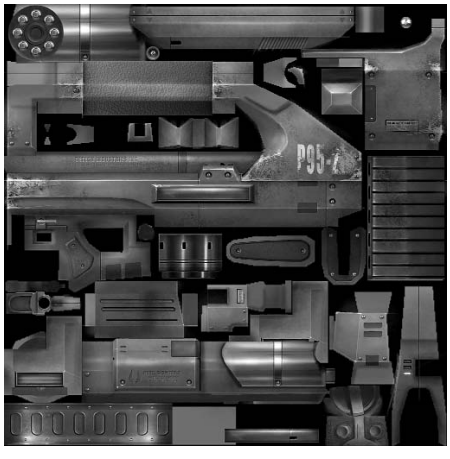


FIGURE 2.17 Assault rifle texture.

- *Creature setup technician*—This team member is responsible for creating the control system that turns your character models into digital puppets that the animator can later set to move. Creature setup technicians need to have a thorough understanding of character rigging techniques in the 3D package you have decided to use for the project, such as Maya or 3ds max. Often, especially on smaller teams, this person doubles as an animator.

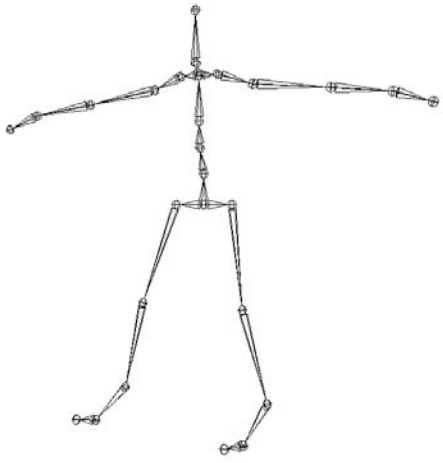


FIGURE 2.18 Bones for a game character.

- *Animator*—When motion is needed, the animator is the one to call. The animator not only understands animation methods in your selected 3D animation software, but also has a thorough knowledge of the way objects move, especially their timing. If the animator isn't doing the creature setup, you should make sure the creature setup technician and animator work together closely so that there are no conflicting ideas about how control systems should be designed.
- *Programmer/coder*—This team member knows how UnrealScript works and how to use it to integrate the necessary functionality into the game or project. In most cases, programmers also understand many of the fundamental concepts that drive the Unreal Engine, so don't be afraid to go to them for advice on your idea. Good programmers should also understand the capabilities of the UnrealScript language so that they can judge the difficulty—or even the possibility—of integrating a new idea into the system.
- *Project manager*—This person doesn't usually create any of the game's elements, but still plays a vital role by making sure each part of the project is being completed on time and any internal problems are solved. In many cases, this team member might turn out to be you. If, however, you don't feel comfortable with or capable of keeping your workers on schedule, you should probably consider finding someone else for the job. If you have someone other than yourself in this position, make sure you stay in close contact with him or her throughout the project so that you're informed of the project's progress.
- *Web designer*—The web designer can be any member of the team who has experience in creating web pages, but you might want to bring in a person dedicated strictly to designing a site and keeping it maintained and updated. Get other online game community sites and forums to mention your game or project so that people check out your site and get excited about its release. Promoting your project is an important aspect of its success. An easy way to get your game noticed is to design a web page that informs the public about your project and offers updates on the progress of its development.

Choose the Right People

Find recruits who are reliable and hard-working. This advice cannot be emphasized enough. You need people for your project who are just as serious as you are about seeing it through to the end. Don't be afraid to reject potential team members who don't seem as though they can stick with the project. If you don't think someone is cut out for the project, be honest and explain why you feel that way. Your honesty might even provoke him or her into "proving you wrong" by working hard and becoming a strong member of the team.

Find people who enjoy working on jobs such as your project and are willing to dedicate time to it. Remember that a lot of people think game design is cool and will jump at the chance to get involved, but they might not realize the tremendous amount of time and work involved. You need to find team members who understand the difficulties ahead and are willing to work through them.

Know Their Capabilities

Don't add someone to the team as an animator unless you know this person can animate well. Ask to see references or past work. Depending on the professional level of your project, you might request a formal resume, demo reel, or documentation of previous related work. Ask people who have worked with your potential team member in the past so that you have a better idea what to expect of his or her work and attitude. At the very least, make sure you know whether this person is capable of the work and skill level needed for the project.

Define the Roles Clearly

Many problems that happen in a team occur when the boundaries between different team members' roles blur. This can happen when lines of authority aren't clearly drawn or when your team hasn't been thoroughly informed about the tasks for which they are responsible. Head these problems off before they occur by letting people know ahead of time who their supervisor is supposed to be, whether it's you or someone else. Don't allow team members to step on each other's toes. For instance, if your animator is spending time editing a character model that your modeler said was finalized, you have a problem that needs to be addressed immediately.

Make Sure They Are Willing to Listen

There's no point in adding a team member who doesn't listen to directions. Make certain your workers respect authority, are willing to listen to whomever is in charge of their work, and follow the instructions given to them. At the same time, don't bring in team members who can't accept criticism of their work from you or other team members. Acknowledging problems is the first step in repairing them, and if one of your workers can't tolerate being told that he or she has done something wrong, you might want to reconsider keeping this person on the project.

Compatibility Is Important

Choose team members who can work well with others and have a positive attitude toward the project. In a perfect world, even people who don't see eye-to-eye can still work together. In reality, however, this isn't always the case. If a team member's attitude is detrimental to the rest of the team, that person should go. Of course, you can't *make* all your team members like each other, but you can let popular opinion help you with decisions. For instance, if a potential team member just can't get along with most of your group, make sure this person is working closely with those who *can* tolerate him or her, or perhaps just drop this person from your team.

Honesty Is Still the Best Policy

Be honest with people from the start. If you know the schedule will be tough, let your team members know when they sign on. If team members aren't working up to your standards, tell them immediately, but remember to be fair in your criticism of their work. Fairness and honesty go a long way toward preventing conflicts. Your honesty can help your team members trust you and place more value on your opinions.

Keep Personal Schedules in Mind

Make sure you know what personal obligations your team members have. For example, if a team member must attend a weekly meeting, make sure to fit your schedule around that meeting, if possible. At the same time, remember that you can't have an effective team if everyone needs time off at different points throughout the week. Be sure you speak with your entire team and find a schedule that everyone can agree on. Note that finding a workable schedule isn't always as simple as it seems.

Compensating Team Members

Not everyone on your team can work for free. To anyone who has worked for a company or corporation, this statement seems obvious, but the subject of compensation seldom comes up with nonprofessional projects, games, or mods, as self-promotion or recognition from the game community is quite often the goal. However, you might have a team member who can't donate his or her time to your project but would be a tremendous asset to production. For example, you might know a professional programmer who could cut your coding time in half, so you might want to make that person's involvement in your project worth his while. Whatever the case, be sure that you carefully weigh costs versus benefits when compensating your team members.

Refining Your Ideas with the Team

Now that you have signed people on to help you achieve your vision, you should meet with them and gather feedback on your project. Take advantage of your increased base of ideas. There's a good chance that a team member will think of something new, and its implementation could prove worthwhile. At the same time, be open to the possibility that some aspects of your plan

might not flow as well as you thought, and be willing to accept opinions and feedback on what to do. Also, meet with team members individually and hear what they have in mind for their part of the project. For example, you might want to speak with your concept artist to see how he or she envisions the characters and objects in your project, or make sure your level designer has a clear idea of your project's look.

Generate a printout or an email that covers your idea in its entirety and give each team member a copy. After they read it, encourage them to write down ideas they think would enhance the original concept. You can then share these ideas with the group and come to a consensus on whether to include them. Having these meetings is also a good way to find out whether your team is capable of completing every aspect of the project or you need to bring in additional help.

Creating a Production Schedule

Few things are more discouraging to a project's participants than falling off schedule. Keep in mind that during production, delays are bound to happen, and you and your team should be ready to strive to maintain the set schedule. During the project, remember to make the most of your project manager's skills to keep your team's progress on time.

How do you set up a workable schedule for your project? It takes close communication between you and your team members to make sure they know exactly what they need to do and the order in which to do it. They also need to let you know how fast they can complete each task. With that knowledge, it's much easier for team members to have a schedule they're able to maintain without undue pressure.

Scheduling is not a quick process, so don't rush it. Creating an individual schedule for each team member is probably a good idea; this schedule should state specific details for each task the person needs to do and outline precisely when the task should be completed. You should base this schedule on the team member's capabilities and availability.

After you have created these individual schedules, you should bring them together to form a master schedule. Team members can refer to this generalized schedule at any time to check on the progress of other departments, without having to see the specific details of each task to be completed.

Beginning Construction: During Production

Now that you know exactly what's going to happen during your production and when each task is going to take place, it's time to get to work! Production is a process that you should find fulfilling and rewarding. Nevertheless, you will probably encounter a variety of problems during your project. Although the intention of this chapter is not to cover every possible problem, a few situations that commonly arise are discussed as well as how you might go about solving or reacting to them.

Maintaining the Schedule

After production has officially begun, the schedule is of maximum importance. Make sure your project manager follows the progress of all team members or departments and keeps everyone on schedule (or as close to it as possible). Often this job can be quite challenging for your project manager, so be sure to offer advice or try to help as much as you can.

The project manager needs to keep team morale in mind, too. Remember the difference between constructive and destructive criticism: Constructive criticism points out problems yet encourages team members to keep trying to improve. Destructive criticism only shames, alienates, insults, and, in the end, drives your team members away from the project. For example, telling a team member that his recent work isn't up to his usual high standard and could improve with some extra effort goes much farther than simply calling his work "a pathetic excuse for game content." Sometimes there's a fine line between these two types of criticism, so try to keep team members' overall morale in mind when criticizing their work, but remember to stay honest at all times. This guideline is especially important for team members who are volunteering their time because only their morale and enthusiasm will keep them from leaving your team.

Dealing with Delays

Remember that delays happen. They can occur for a variety of reasons, from personal crises to software issues. The key is not to let delays bring you or your team down. Try to get back on schedule, or if that isn't possible, form a new schedule that takes your delay into account. However you deal with a delay, make sure you inform the public, regardless of the disappointment this announcement might generate. In the game development world, delays are common, but this doesn't mean they should be taken lightly. A project with zero delays is probably never going to happen, but a project with too many delays can lose the interest of an eager public.

Bringing It Together

After you have enough project assets completed, it's time to integrate them into a functional game. This task is usually the work of the level designer and the coder, as they have the clearest idea of how the Unreal Engine works. This step includes developing the functionality of game assets and making sure all assets work correctly when combined. They also need to ascertain how the game, mod, or project plays when it's finished.

The Unreal Engine is extremely powerful and capable, but you might still need it to perform tasks it doesn't do in an existing game, such as UT2004. If you're experimenting with the Unreal Engine, you'll find that even simple functionality, such as weapon firing and behavior, needs to be added to the system's code for use in your project. Your programmer needs to establish this functionality even before assets are put into place in the project.

For example, when playing a first-person shooter game, one of the simplest functions is firing a weapon. When playing, you press a button, your Shock Rifle fires an energy beam, and the beam

strikes something and causes damage. This sequence sounds simple from a player's perspective, but internally, a lot of things are going on. First, an UnrealScript weapon script is making a call to a weapon-firing script, which then initiates a projectile script. As the projectile script runs, it keeps track of whether the fired projectile strikes something. If it does, it implements the damage function of the object being hit, which responds by taking note of what type of projectile just hit it and how much damage the object suffered as a result.

Sound complex? That's just a small taste of the kind of functionality needed to make a game behave properly. On top of that, there's the behavior of bots and vehicles and how physics work in your level—the list goes on and on. Don't be surprised if integration takes a while or if certain aspects need to be reworked into the project during the development process.

Testing and Acquiring Feedback

After you have integrated enough of the game that it's playable, it's a good time to begin testing. In nearly all professional projects, testing begins long before the project is completed. The process is fairly simple: Have someone play the game, knowing which parts have been completed and which haven't, and make sure those completed elements are working correctly. You can get the whole team involved in this process, or if they are too busy, call in some extra help. Usually, little effort is needed to convince a fellow gamer to play-test a new game. Testing typically comes in two phases: alpha testing and beta testing. The following sections describe each phase and explain how this testing can be used to help finish your project and maintain its quality.

Internal Testing: Alpha

Alpha testing is usually done in house, among the team members. Depending on the size of your staff, however, you might need to recruit outside testers. Your testers should be aware of how the project development process is progressing and precisely which elements have and have not been fully implemented. In most cases, you don't want to release an alpha edition of your software to the public because you're sure to get plagued with a barrage of emails from people who have no idea how your game is being created. Make sure security is tight for your alpha testing phase, and ensure that your team members aren't revealing it to the web community.

Say, for example, that someone leaked your game's alpha version onto the Internet. At this point in your production, you have integrated player mobility and weapon functionality for only 2 of your 26 weapons. In addition, none of your vehicles have been programmed completely, so trying to enter one of them causes the game to crash because of the incomplete code. When alpha testers play the game, they know ahead of time exactly what they can and cannot do, so a game crash isn't a problem for them. The general public, on the other hand, won't be able to understand your project's current limitations. If they see your incomplete game, they could start generating bad publicity for your project based on their judgment of an incomplete game.

Alpha testing is a good way to collect data, make changes to your project, tweak functionality, and streamline the various aspects of your game. Eventually, you have each element of your project fully integrated and working correctly. Then you're ready to move on to the next level, beta testing.

External Testing: Beta

Beta testing essentially works the same as alpha testing, but it's performed on an almost complete project. In most cases, you won't want to release your beta version to the public either. Your beta version will eventually become the finalized game or project, so you don't want it to be freely downloaded before the official release.

Be careful who you choose as a beta tester. Security is always a concern. Also, it's not enough for beta testers to just be gamers. They also need to be interested in helping you with your project by looking for problems that gamers could find, such as "holes" in the maps, nonfunctioning elements, and incompatibility with certain hardware or software. Also, make sure your beta testers are prompt with their feedback and aren't just taking advantage of you to play your new game.

In many cases, you should take applications for your beta testers. An application doesn't need to be formal, but it should ask potential testers what kind of experience they've had with beta testing, what kind of games they like to play, and what kind of computer they have. This last question can be important because you want to make sure your game or project is tested across a wide variety of platforms and video cards.

Remodeling and Closing: Ending Production

You've built your game and are in the middle of the final tests, getting ready to release your project. At this phase, you should be wrapping up your beta testing, getting your web designer to start promoting the launch date, and finalizing all parts of the project. But what do you do with all this beta test data and, more important, when do you quit beta testing?

What to Do with Beta Test Info

The purpose of beta testing is to get feedback from a larger base of users to find and fix any nonfunctional aspects of your game or project. Collect as much data as you can from your testers and cross-reference it with other testers to see what the biggest problems are. Use this data to fix your game's most important problems. Processing all this data is a major task, and you will likely want to get your entire team involved. In the end, much of the repair work will probably fall on the level designer and the programmer, so try to keep them from getting tied up with other tasks.

Feedback from your beta testers is vital to finishing your project. You need to know what kind of video cards your game supports, what kind of drivers are needed, and so on. This feedback can also help you discover and repair the small (but still important) problems that arise in most game development. Beta testers can often tell you where loopholes are in your project.

For instance, say your map has a section guarded by two massive creatures, and it's supposed to be difficult for players to pass. As it turns out, in a previous part of the map, players can drive a tank and, with just the right driving skills, drive it all the way to where the two monsters wait. Players can then use the tank's cannon and eradicate the monsters with practically no effort, thereby avoiding a challenge that's been carefully crafted for them. Avoiding obstacles in this manner isn't cheating, really. It's just a section of your map that needs to be tweaked. With your testers' feedback, you need to edit the map so that perhaps a large obstacle prevents the tank from being used on the monsters. You need to remedy as many of these problems as possible.

Keep in mind that your project will likely go through several beta versions. After you finish repairing many of the problems that come up, make a new version available to your testers to help narrow down the number of problems even more before the project is released.

When to Stop Beta Testing

Remember that a game can be like a work of art. Often, it's difficult to tell exactly when it's finished. When deciding whether your project is finished, you need to take a few factors into account. First, take a look at the kind of feedback coming in from your beta testers. The number of severe issues should be winding down. More important, remember your schedule! Don't go too far off schedule during the tweaking process.

Keep in mind, too, that most games (excluding console-based games for systems such as the Xbox or PlayStation) aren't released in what most gamers might consider to be a fully completed state. Despite developers' valiant efforts, bugs or glitches can often slip by the testing process and wind up in the final game. This is the reason that many games offer patches and post-release fixes for problems. You want to make sure your game is as clean as you can make it and that no errors exist that could hinder or cripple gameplay. Realistically, however, it's likely that you won't be able to think of everything before the project is released.

Finalizing and Releasing

Congratulations! You've made it through the course of production, and now it's time to make your project available to customers. This time could prove to be busy, depending on the size of your project. At this point, your game could "go gold," which means it's being manufactured on CD, or your mod or project could be made available for download to the online community. Your project is officially finished, but the work isn't over for the entire crew. You still need to support your game by creating and releasing game patches for any technical problems that come up. Plus, you want to continue promoting your game to the community!

Summary

This chapter has covered many different topics to consider before launching into a major gaming project. Although it's a given that some of your projects might be so small that you don't need to use all the techniques and ideas presented here, eventually you might want to try your hand at a project with a larger scale. Also, this chapter has been designed to give you a better idea of some of the work that goes into even the simplest games. At this point, you should have a better understanding of the hurdles that await those who want to take on a serious game development project.