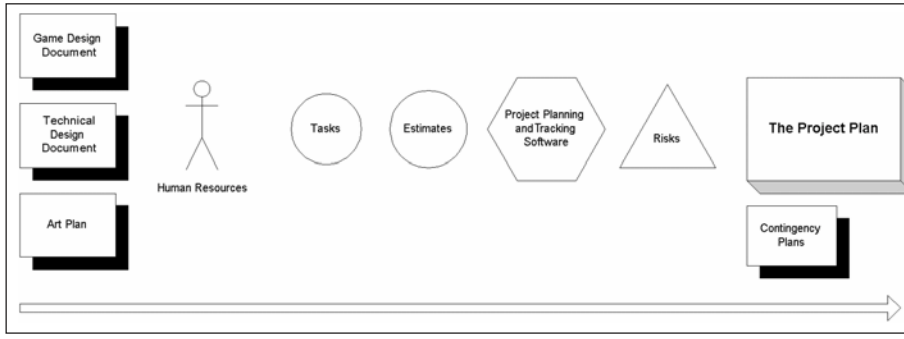
Chapter 10

The Project Plan

What Is the Project Plan?

The project plan is the culmination of the planning articulated in the game design, technical design, and other design documents such as an art style guide. The heart of the project plan is a schedule that describes what will be accomplished, how long the tasks will take, and who will perform these tasks. The project plan contains other information such as milestone dates, task dependencies, and a risk management plan. The information in the project plan is published to both the executive management for progress reports and to team members in the form of tasking. It is also used by the project manager to level tasks between resources, identify critical paths, and develop contingency plans. A good project plan will act as a major tool to avoid surprises. All this seems like good stuff, so let us get on with making a project plan.



Components of a project plan: estimates, resources, tools, tracking, dependencies, risks, and alternate plans

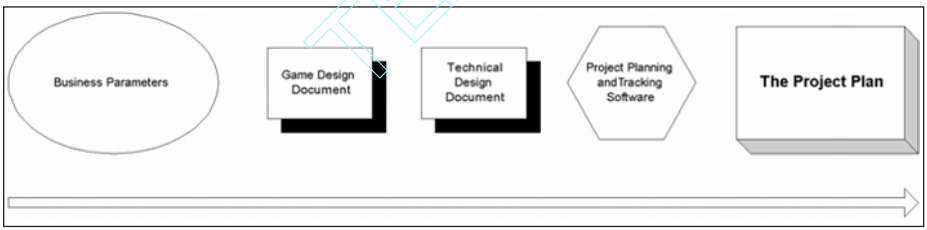
How Do We Create the Project Plan?

To create the project plan we will need a list of tasks to be completed, who is available to perform those tasks, when the critical project dates are such as milestones, and what the relationships are between the tasks. The list of tasks, the estimates of how long it will take to perform these tasks, and their dependencies will come directly out of the

game design document and technical design documents.

Critical project dates such as milestone and release dates should be iteratively arrived at with the executive management team as the project plan is compiled. Many projects are schedule driven; however, the most common for the game industry is the holiday shopping season from Thanksgiving to Christmas every year. Often projects will be planned by walking backwards in time from November to discover the critical dates like beta, alpha feature lock, first playable. With these projects it will be the project plan that adapts to the critical dates. This is discussed later in this chapter.

Developers Conference I discovered there was a fairly wide range of project management rigor applied to game development. Some shops considered themselves too small to plan their work; they just worked on whatever was the most pressing task at the time. Many developers just used simple spreadsheets in Excel to plan their projects; some folks used Microsoft Project to plan their tasks and then followed up in Excel to perform their task tracking; the more determined developers used Project for both planning and tracking; and one large French developer that was part of a construction firm used Project to plan and Microsoft Team Manager for task tracking.



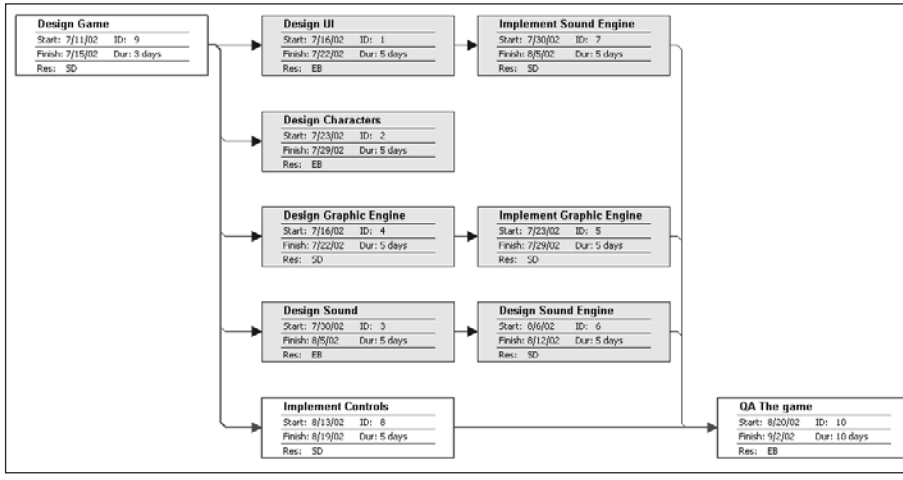
The project plan pipeline

All of this project information will need to be compiled into a usable format for project analysis and report generation. With tools such as Microsoft Project or Primavera's SureTrak products, a myriad of reports and graphs can be generated to review the workload across team members, understand what the critical path is, measure project progress, and a whole host of other views of your project status. Many people are intimidated by project planning, or they have seen project planning only partially implemented that failed to work. From my roundtables on game production at the Game

Gantt and PERT Charts for Organizing Project Tasks

There are many reports, graphs, and charts used in project planning and tracking. The two most commonly used charts are the PERT and Gantt charts. The acronym PERT stands for *Program Evaluation Review Technique*, a methodology developed by the U.S. Navy in the 1950s to manage the Polaris submarine missile program. The PERT chart places each task in a rectangular box with a line drawn to the predecessor task and a line to the next task; thus the whole diagram looks like some sort of tree. The PERT chart's key feature

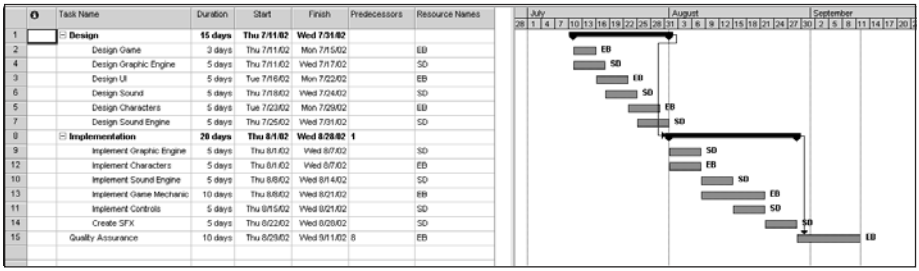
is the visual ease in identifying the relationship between tasks and the critical path of the project as a whole. The drawback of a PERT chart is that its utility is limited to just the higher-level view of a project. When individual tasks of any nontrivial project are displayed, the resulting chart is crisscrossed with lines and is too unwieldy for the viewer to absorb.



A PERT chart from Microsoft Project

excels in data entry, as there is no end to fussing about where to put the boxes as in a PERT chart. The project manager usually just needs to enter the task name, estimated time to complete, and a resource to complete the job. A tool like Microsoft Project will automate the graph side of the chart. The Gantt chart will also accept task dependency information like the PERT chart and draw arrows between tasks to show what order the tasks must be created in; however, the Gantt chart produces a more flat graph that does not show off the dependencies of tasks as well as the PERT chart. The visual clutter of a Gantt chart can be minimized to a great extent by nesting the tasks into a hierarchy of task, subtask, sub-subtask, etc. Microsoft Project allows for a total of nine levels of task nesting.

The Gantt chart turns out to be the most generally useful of the project planning and tasking charts. It features a spreadsheet-like data entry on the left-hand side of the chart with any number of columns, the minimum being task name, task start date, task duration, and resource name. On the right-hand side of the chart is a modified bar graph where each task is a horizontal bar organized in a cascading hierarchy as time progress. The Gantt chart



	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	Design	15 days	Thu 7/11/02	Wed 7/31/02		
2	Design Game	3 days	Thu 7/11/02	Mon 7/15/02		ED
4	Design Graphic Engine	5 days	Thu 7/11/02	Wed 7/17/02		SD
3	Design UI	5 days	Tue 7/16/02	Mon 7/22/02		ED
6	Design Sound	5 days	Thu 7/18/02	Wed 7/24/02		SD
5	Design Characters	5 days	Tue 7/23/02	Mon 7/29/02		ED
7	Design Sound Engine	5 days	Thu 7/25/02	Wed 7/31/02		SD
8	Implementation	20 days	Thu 8/1/02	Wed 8/28/02	1	
9	Implement Graphic Engine	5 days	Thu 8/1/02	Wed 8/7/02		SD
12	Implement Characters	5 days	Thu 8/1/02	Wed 8/7/02		ED
10	Implement Sound Engine	5 days	Thu 8/8/02	Wed 8/14/02		SD
13	Implement Game Mechanic	10 days	Thu 8/8/02	Wed 8/21/02		ED
11	Implement Controls	5 days	Thu 8/15/02	Wed 8/21/02		SD
14	Create SFX	5 days	Thu 8/22/02	Wed 8/28/02		SD
15	Quality Assurance	10 days	Thu 8/29/02	Wed 9/11/02	8	ED

A Gantt chart from Microsoft Project

In my experience in game production I have found the Gantt chart to be crucial and the PERT chart fun. By fun I mean that the PERT chart is so easy to digest visually that seeing the key tasks getting completed and checked off as the project heads towards completion is a visual treat. The problem with the PERT chart is again you must reduce the task resolution to just the highest level tasks. This results in relatively chunky task descriptions like "implement 3D engine," "script campaign one," and "alpha test," and these relatively chunky tasks are actually composed of many tasks spread over a great deal of time. The PERT chart becomes dissatisfying when you want to mark off a PERT box when something is 90 percent complete even though the final 10 percent will not be completed for some time. For this reason I do not use PERT charts for my own projects.

NOTE: Please see Chapter 20 for a quick survival guide to Microsoft Project.

Focusing on the Gantt Chart

So how exactly do we create a Gantt chart? Obviously we need to know what the tasks are, who is going to do them, and how long they will take. The ideal Gantt chart entry is a single, clear, discrete task with a short duration (debate rages but aim for between .25 day to 3 days in your task resolution). An example of a poor entry would be *3D engine, 4 months, Bob.* This task is poorly described for two reasons: The first is the name itself, *3D engine*. What does that mean? Test it? Design it? Debug it? Implement it? Review it? Break it? Fix it? Vague project task names must be attacked ruthlessly and reduced to a lean, aggressive name like *Create static design of the core 3D engine*. The second thing wrong with this task is that *it is four months long!* Good grief, why are we even putting together a schedule? How will it serve to measure progress when we can only look Bob up after 15 weeks and ask if he thinks he will make it next week? With such coarse resolution we are simply not getting enough incremental task progression data to have a meaningful analysis of whether the project is tracking. For if we are not tracking, maybe we should cut features in the 3D engine, or maybe we need to add another programmer to work on the custom shaders, or maybe we should kill the new 3D engine altogether and make do with the previous engine or integrate a commercial 3D engine. All of these tough choices can be uncomfortable or even impractical for you and your project, but these choices are

certainly not more comfortable when Bob has been working for 15 weeks and then admits that the 3D engine turned out to be tougher than he thought and that in four more weeks he will know more!

There is a time and place for coarsely defined Gantt charts—very early in the project when you are defining your business parameters. At this time it is useful to block out your project with these coarse task granularities to get an idea of how many people you will need and about how long it will take to get the job done. This proto-Gantt chart can then be used iteratively to help define the costs of the project while they are still fairly malleable in the early project negotiation phase. In the ideal world a publisher would sign up a project and pay for three months of preproduction to determine the detailed project schedule; this rarely happens. Instead, you often work on the rough size and scope of a project and then use the early milestones to refine your schedule and kill features to make the project fit into the negotiated costs. I have to warn you, creating these proto-schedules is not a substitute for going through with your full preproduction phases and determining your task estimates in detail! You should also avoid creating a proto-schedule if you have little experience in project planning, or where the game has not yet jelled into a clear vision, or where there are a lot of associated technical risks because you or your team have not developed a similar game.

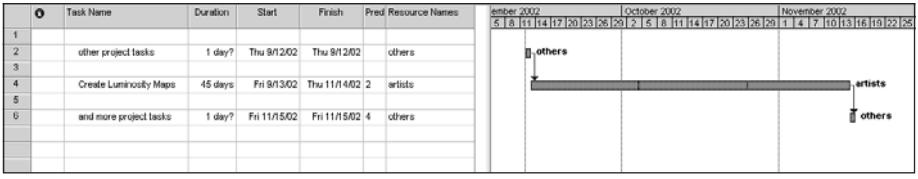
WARNING!—Creating proto-schedules should only be done by experienced project planners who have managed similar projects and where the game scope is well understood.

Using the Technical Design Document

The technical design document is supposed to have the information for the technical tasks. Depending on how you organize your team members, the game design document, the technical design document, or a stand-alone art asset document will describe the art tasks. Wherever the information is coming from, as you sit down to enter these tasks and time estimates into Project, you will discover that these tasks are not ready for immediate entry. For instance, the 30 luminosity maps needed for your starships will be listed as 45 man-days from your art director. Should you enter that as a single task named *luminosity mapping, 45 days, artists?* No you should not; that would be creating a vague task entry like the previous 3D engine example. Will you have all of your artists working on this for 45 days? Will you burn one of your artists on this tedious task? Do you need all of the luminosity maps done at the same time? These are the questions you need to ask yourself as you translate the task estimates from your leads into your schedule. For a task like this I would write up 30 1.5-day tasks and distribute them evenly across the artists I had available to perform these tasks (in my management style, if there is something boring and repetitious, I generally distribute the tasks evenly, perhaps a bit heavier on the junior team member). Getting ahead of myself into a discussion of risk management and task dependencies, I would schedule these repetitious, low-risk tasks towards the end of the project with perhaps some sample luminosity maps done early on to verify we understood

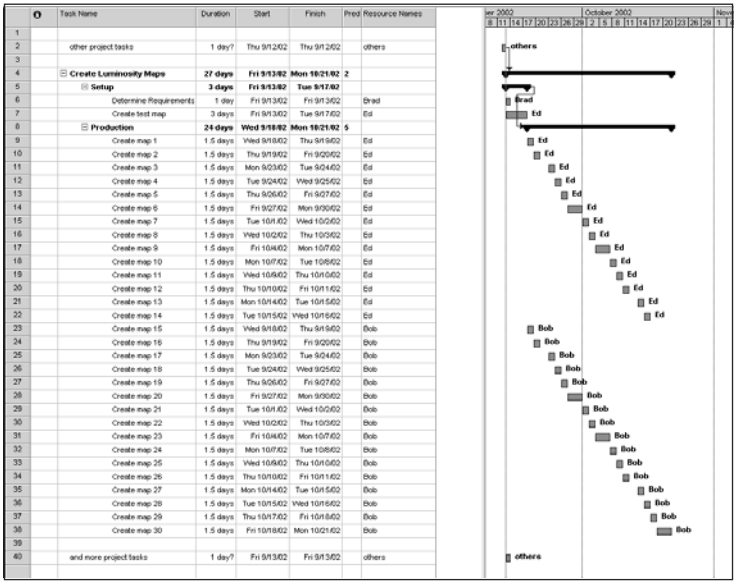
the production path and time estimate to create the maps. These 30 1.5-day tasks would be an eyesore to look at if we were to enter them flat into the schedule. To handle that bit of dust, sweep these 30 entries into a super-task named *create luminosity maps*. This way we can view this individual information easily by expanding the super task *create luminosity maps* and hide it when it is not of immediate interest. Each of the 30 subtasks should also refer to the specific starship that the map is for or at the very least be uniquely identified as in *create luminosity map for Federation Enterprise-E, 1.5 days, Ed.*

It will be difficult to always break down tasks into their proper subtasks. For example many times you will get a reasonable-sounding task like *investigate pixel shaders, 3 days, Tom.* It has a fairly clear verb—investigate—right? Well, does it mean Tom will spend three days on learning what is going on with pixel shaders and then move on? What is the deliverable for this task? Will Tom merely know more about pixel shaders or are you expecting to implement pixel shaders? Will the artists need to perform additional work to support the pixel shaders if Tom gets them done? I recommend this task be broken down into the following tasks: *Investigate the*



	Task Name	Duration	Start	Finish	Pred	Resource Names
1						
2	other project tasks	1 day?	Thu 9/12/02	Thu 9/12/02		others
3						
4	Create Luminosity Maps	45 days	Fri 9/13/02	Thu 11/14/02	2	artists
5						
6	and more project tasks	1 day?	Fri 11/15/02	Fri 11/15/02	4	others

A poorly broken down task—too long



	Task Name	Duration	Start	Finish	Pred	Resource Names
1						
2	other project tasks	1 day?	Thu 9/12/02	Thu 9/12/02		others
3						
4	Create Luminosity Maps	27 days	Fri 9/13/02	Mon 10/21/02	2	
5	Setup	3 days	Fri 9/13/02	Tue 9/17/02		
6	Determine Requirements	1 day	Fri 9/13/02	Fri 9/13/02		Brad
7	Create test map	3 days	Fri 9/13/02	Tue 9/17/02		Ed
8	Production	24 days	Wed 9/18/02	Mon 10/21/02	5	
9	Create map 1	1.5 days	Wed 9/18/02	Thu 9/19/02		Ed
10	Create map 2	1.5 days	Thu 9/19/02	Fri 9/20/02		Ed
11	Create map 3	1.5 days	Mon 9/23/02	Tue 9/24/02		Ed
12	Create map 4	1.5 days	Tue 9/24/02	Wed 9/25/02		Ed
13	Create map 5	1.5 days	Thu 9/26/02	Fri 9/27/02		Ed
14	Create map 6	1.5 days	Fri 9/27/02	Mon 9/30/02		Ed
15	Create map 7	1.5 days	Tue 10/1/02	Wed 10/2/02		Ed
16	Create map 8	1.5 days	Wed 10/2/02	Thu 10/3/02		Ed
17	Create map 9	1.5 days	Fri 10/4/02	Mon 10/7/02		Ed
18	Create map 10	1.5 days	Mon 10/7/02	Tue 10/8/02		Ed
19	Create map 11	1.5 days	Wed 10/9/02	Thu 10/10/02		Ed
20	Create map 12	1.5 days	Thu 10/10/02	Fri 10/11/02		Ed
21	Create map 13	1.5 days	Mon 10/14/02	Tue 10/15/02		Ed
22	Create map 14	1.5 days	Tue 10/15/02	Wed 10/16/02		Ed
23	Create map 15	1.5 days	Wed 9/18/02	Thu 9/19/02		Bob
24	Create map 16	1.5 days	Thu 9/19/02	Fri 9/20/02		Bob
25	Create map 17	1.5 days	Mon 9/23/02	Tue 9/24/02		Bob
26	Create map 18	1.5 days	Tue 9/24/02	Wed 9/25/02		Bob
27	Create map 19	1.5 days	Thu 9/26/02	Fri 9/27/02		Bob
28	Create map 20	1.5 days	Fri 9/27/02	Mon 9/30/02		Bob
29	Create map 21	1.5 days	Tue 10/1/02	Wed 10/2/02		Bob
30	Create map 22	1.5 days	Wed 10/2/02	Thu 10/3/02		Bob
31	Create map 23	1.5 days	Fri 10/4/02	Mon 10/7/02		Bob
32	Create map 24	1.5 days	Mon 10/7/02	Tue 10/8/02		Bob
33	Create map 25	1.5 days	Wed 10/9/02	Thu 10/10/02		Bob
34	Create map 26	1.5 days	Thu 10/10/02	Fri 10/11/02		Bob
35	Create map 27	1.5 days	Mon 10/14/02	Tue 10/15/02		Bob
36	Create map 28	1.5 days	Tue 10/15/02	Wed 10/16/02		Bob
37	Create map 29	1.5 days	Thu 10/17/02	Fri 10/18/02		Bob
38	Create map 30	1.5 days	Fri 10/18/02	Mon 10/21/02		Bob
39						
40	and more project tasks	1 day?	Fri 9/13/02	Fri 9/13/02		others

The previous task broken down into 30 bite-sized 1.5-day tasks distributed to two artists, with an early phase to determine the validity of the task estimate for both artists, rolled up under a super-task.

feasibility of pixel shaders, 1 day, Tom; Implement core support for shaders, 1 day, Tom; Implement simple shader for ripple effect, 1 day, Tom; Determine what additional work the Artists must perform, .25 day, Tom; and then wrap up all of these tasks under a super-task named *implement ripple shader*. Picture in your mind that it is your job to view each and every incoming task as a crystalline rock that you examine closely, looking for the fissures that represent the subtasks inside of the project. Then you grasp the task firmly in your hands and break it up along these fissures.

Breakin' down tasks

Task Granularity and Task Leveling

Task leveling is the act of distributing the workload across your developers so that no one developer is stuck holding up the show while the rest kick back at the beach. Task leveling is a difficult and imprecise business. No two developers on your team will produce code, art, or other game development bits at the same rate and with the same level of initiative and independence. Task tracking is such a central activity of game production that the next chapter is dedicated to its discussion. However, here in the planning stage we can set ourselves up for success later by planning our task leveling now. In the previous section I stressed breaking down large, vague tasks into clear, crisp, small tasks; it turns out that breaking tasks up into crisp bite-size chunks is also critical for effective task leveling. By breaking up the tasks into their smaller pieces you will not only see more clearly just how much work you have to do, but you will also be able to better analyze how to distribute your tasks across your company.

How Long Will That Task Take?

As you enter the data you will not only need to break up the tasks into smaller tasks, but you will also need to spend a moment chewing on the time estimates being reported by your team members. There is a lot of debate in the community about padding tasks by two times or three times to count for the chronic underestimation that developers are prone to make. I fundamentally disagree. I think it is a very bad thing for the development team to think in terms of "programmer-hours" and feel assured that their management lead will take responsibility for padding the schedule to accommodate their optimism. If you think about this for a moment, it does seem ludicrous to take the developer's estimate and institutionally lie and come up with another number. I believe the reason organizations do this multiplying technique is they have found that taking the developers estimates has resulted in previous projects slipping and going over budget. The answer is the development process is flawed, and that is why the project is late, not because a developer makes poor estimates. How can a project succeed when such arbitrary estimates are tossed around?

So how do you get good time estimates? First of all I do not make creating the time estimates *my responsibility* as the project manager; I make that the *developers' responsibility.* Is this just a semantic nuance? No, the way to success is to push down to them the *responsibility*, the *authority*, and the *accountability* to create their own time estimates. I will not be performing the work; they will. Your team members are not just coders or pixel pushers; they are game developers. Grow your organization so they understand that creating quality estimates is part of their job and that they need to make an estimate they can live with.

Will pushing estimating down to the team members work? What about the new artist; does he know how long it will take to texture the level? How about the AI programmer; now that he has been tasked to create the networking code, how will he come up with a quality estimate? I am not saying that the senior team members such as the art director and the lead programmer as well as the project manager should not participate and help develop the estimates. What I am saying is that my team performs best when they are working under a schedule they drafted. It may look like I have not solved the time estimate problem; it may look like I just moved it down to the developer, but that is too casual a statement. When you walk up to Sally and ask her how long it will take to create a mission editor for the game, she might reply with a shrug and a soul-searching glance at the ceiling and come back with an estimate of two months. This is a low-quality estimate. Much better is to walk up to Sally and say to her, "I want you to think about what it is going to take to get the mission editor done; specifically, I want you to review the technical and game designs for the mission editor and break it down into a task resolution of one to three days each and enter your tasks into Microsoft Project. Would Friday be okay with you to review your schedule?" This is much stronger because you gave a clear task of getting her area estimated and put into a schedule, and you told her how to get it down with the comments on the time resolution and Project. You also gave a firm date and gave every indication that it is her responsibility.

So what do you do when developer estimates are too short or too long? You are the project manager, and you have responsibility for running the project. While the buck stops with you, your job is to get the right people matched to the right tasks with the proper tools and resources to get the job done. It is the artists and the art director who are responsible for the art estimates. You said that before, Erik, but what do I do with a time estimate that is clearly too short? I want you to review every time estimate for a reality check, a second opinion, and for your own benefit to build up a better mental map of how long the myriad of development tasks take. What I suggest you do with a short time estimate is interview the developer and/or lead for that section and ask them why they thought they could accomplish it so quickly. Maybe you will find out something you did not know; that would be a good thing. Maybe they will shrug and admit they didn't give it enough thought. Or maybe it is a feature they very much want to

see get done and do not want to see it cut so they are "selling you" the feature.

Short Time Estimate Possibilities

If the developer did not give the estimate enough thought, then simply kick it back for a revision. If you simply were not aware of something that will make the task quicker to complete—no problem, accept the estimate. However, when it turns out they are selling you on a feature, this could be a problem. First of all, this means you have a flaw in your schedule that needs to be corrected or the rest of your schedule will be affected. The hard part is that your developer is selling you this feature because she really wants to see it get in the schedule and she felt she needed to underestimate the task to get it on the schedule. You have three choices: Kill the feature, allow the feature, or allow a fixed amount of time to work on the feature. Each situation is unique, but I tend to ask the developer why she thought it was so important to implement the feature. If she does a reasonable job convincing me it is a desirable feature but I cannot afford to rearrange the schedule to fit in the true time for this task, then I will encourage the developer to drop the feature. Many times the developers will be passionate about getting it done and will propose to keep the time estimate to what the schedule can afford, and they will work hard to squeeze it in. This I feel is fair; the manager should not create schedules that require overtime, but I do feel comfortable with developers working as many hours as they like to create the highest quality game they can.

Estimating Research Tasks

How do you estimate how long it will take to get something done that no one has done before, or no one in your organization has done before? Perhaps there is little in the way of journal articles or books to give direction. How do you estimate how long one of these tasks will take? The first step is to break down the research task into as many small, discrete tasks as possible as we discussed previously. An example: Elaborate on a task named *research pixel shaders* and modify the task to a series of tasks like the following:

1. Install video card with pixel shader support
2. Install DirectX 8.0
3. Review DirectX 8.0 sample shader code
4. Create stand-alone test bed to explore pixel shaders
5. Create water effect through pixel shaders
6. Create fire effect through pixel shaders
7. Design architecture for the 3D engine to utilize pixel shaders
8. Implement pixel shader architecture
9. Unit test the pixel shader code
10. Implement fire effect—attach to fireball spell
11. Implement water effect—attach to water blast spell
12. Test the fireball spell
13. Test the water blast spell

By breaking down *research pixel shaders* into 13 subtasks, we can put good estimates on most of the tasks. Only task number four, *Create stand-alone test bed to explore pixel shaders*, looks like a type

of research that resists being nailed to a firm time. The solution here is to set a *time box,* a fixed period of time you will allocate to the task. At the end of the time box you will either be done with the research or it will have turned out to be too expensive to continue. Mm, yes, what is that? How can you walk away from something not done? Well you might have to. Say you have 15 months to get your game done with ten developers, five of them programmers. Allowing three months for preproduction and three more months for testing and transition leaves nine production months or a total of 45 programmer months. This is your time budget; if the rest of your project is looking like 44 programmer months, then you have just one month left over to play around with your pixel shader. Put a time box of one month around the pixel shader work. These are the types of hard decisions you will have to make if you are going to run your project on budget.

Oh, so the pixel shaded spell effects were a core feature? Everyone thinks that is what it will take for your Diablo killer to make it over the top? After the one month passes and you are still not done, would you feel it is still so important a task that you would allocate more time to get it done? If so, then your original time box was not honest by taking into account your priorities. Time boxes only work if you stick to them. If the feature is really that important, then you should have allocated two months or three months. When setting a time box, set the maximum amount of time you are willing to spend on a feature of that priority level. Too many times when we are deciding whether or not to implement a feature, we just ask how cool it will be or whether the competition has it, in the end deciding to implement the feature for a number of compelling reasons. Remind yourself that the great games all have a slim feature set that was executed with excellence. Think about that cool research-intense feature; do you really need it? Only a project with unlimited financing and no requirement for shipping can afford to implement features without asking the cost. Think of time boxes as stones in a stream where the rest of the tasks flow around these blocks of time; a few rocks are cool, many rocks is a stretch of rapids, and a wall of rocks is a dam. Determining a task's priority deserves its own subsection.

Task Prioritization

Assuming you and your team are creative folks and that you are making a game with a budget of time and money, you will always face a situation where you have too many ideas for cool features and not enough time to implement them. You are then faced with the job of prioritizing your features to be sure you get the critical features accomplished at the right expense of the less important features.

I have a reliable method for task prioritization: First discover all the absolutely required overhead tasks your team must accomplish or you will not even have a shipping game. These tasks include preproduction, beta testing, getting hardware manufacturer approval, getting licensor approval, creating milestones, and responding to milestone feedback. These are what I call zero-level tasks. Also do not forget to estimate the number of holidays, vacation, and sick time your team members will take, and make a

reasonable provision for turnover (I use one developer for every ten developers per year). Subtract all of this nonproduction time from your overall schedule; this will leave you with the real production time you have to work with. Enter all of the zero-level tasks into your Microsoft Project Gantt chart. (See Chapter 20 for a quick overview of Project and such tips as customizing your team's calendar.)

The next step is to take your design documents and toss every task into one of three buckets: core tasks, secondary tasks, and tertiary tasks. Take your time with this. I highly suggest discussing the relative priority of the tasks with various team members to build consensus and to have some solid feedback.

Now that you have your three buckets, lay out all your core tasks in Microsoft Project using good task articulation techniques, and assign the tasks to the resources on your team. Now that you have your zero-level tasks and your core tasks entered into your Project file, use the project-leveling tool to see how the zero-level and core-level tasks will lay out over time. If you were conservative with what you labeled as a core task, then you should have some extra time left over to start plugging in your secondary tasks. However, if the buckets ended up with too much to do for even your core tasks on the first pass through, then you have to prioritize your core tasks and convert enough of them to secondary to make up the difference. This means that the secondary and certainly the tertiary tasks are unlikely to be completed if you are having trouble accommodating even the core tasks.

JARGON: *Leveling* is the term in project management for the related tasks of seeing how the tasks will lay out over time and how loaded each of your resources are, and the process of distributing tasks across your team to achieve a more even workload.

How do you prioritize the core tasks when you already consider them core? First realize they cannot all be core. A rigorous development process requires developing good time estimates, and you have done that; now you are looking at a body of tasks that are core and *features you really want but do not have the budget for.* Perhaps you can make a strong enough case for these features to get approval to expand your project's budget. If you can do that, great—problem solved. If you are still holding to your original budget, then let me show you how I do low-level task prioritization. It's a crude method really, but it is effective: Take all your core tasks and enter them into a spreadsheet (use Excel) with a column labeled priority next to each task and a task time estimate. Now quickly run down your tasks, reading the task names and saying out loud the first gut-level priority that occurs to you for that task such as 7 or 3 or 10 if it is really critical. Go down your whole column of tasks whether it is ten core tasks or 200. Do this first pass quickly; taking longer will only make it harder. Now you will have a first pass priority for all of your core tasks. Have the spreadsheet software sort the core tasks from most important descending to least important. If you are like me, then you will see that you have stubbornly labeled too many tasks with a 10 or 9, and too few tasks have earned the label of 3 or 2. The way to solve this is to allow yourself

only three level 10 tasks, three level 9 tasks, and so on. Start at the first item labeled 10 and take your time thinking deeply about the feature, discuss it with your team if you have to, but one by one you are going to demote your 10s to 9s until you are left with just three must-do 10s. Repeat this process all the way down your list. The mathematically astute will notice that this specific labeling system will fail if you have over 30 tasks. The exact labeling scheme is not important; it is just important to force yourself to make these prioritization choices. You could use the numbers 999 to 0, you could use the alphabet, or you could use a three-letter alphabetic core like AAA to DDD; whatever you use just leave yourself a set of three tasks at each prioritization level. The size of your task set should be roughly one-tenth of the overall numbers of tasks to be prioritized. Now just draw a line where you run out of time for core tasks, and toss the lower priority tasks in with your secondary tasks.

Bug ID	Bug Title	Priority
2929	CD-Key	
2953	SP - Klingon Campaign - Beginning Stardate is 112400.1, twice what it should be	
2979	Dynaverse - Fleets do not have accept / forfeit options in mission panel	
2987	Dynaverse - Hex changed color to red when Fed was leader and Kling was member	
3031	Dynaverse - Romulans can transfer in Borg officers	
2561	Global - Freighter Convoys do not have escorts	
2607	Campaign Screen - Player's ship gets stuck in Hex	
2609	Tactical Sim - Fed vs Fed fights	
2617	Dynaverse - Jumped from Lt. Commander ranking to a Fleet Admiral ranking	
3110	Dynaverse - Ten turn countdown results in stuck in Hexes	
2624	Dynaverse Campaign Screen - Fleet leader is not clear	
2632	Dynaverse - Can make movement bar disappear when leaving a Hex with refit	
2633	Dynaverse - While being attacked, attacking another will teleport player	
2637	DYNA - Map Screen not refreshing on completion of Mission	
2641	Campaign Screen - Ships inconsistent for Convoy between Attacking or Defending	
609	SP - Campaign Screen - States we are partners with the Contested Sector	
2058	Dynaverse - Cause of numbers appearing after player names in the chat box	
2119	New Conquest - Music stutters and pointer freezes loading new Conquest	
2701	Global - When AI forfeits it stays in the Hex	
2763	Dynaverse - Role of convoys	
2765	Access Server not using list of IP addresses	
2780	SP - General - Player can initiate a battle then auto move kicks in	
2784	SP - General - Player can be attacked when auto move kicks in	
2797	Hex information should appear in game display	
2799	SP - General - There needs to be a message when auto move is enacted	
2817	Dynaverse - "Stand by for mission briefing" panel repeats text	
2826	Dynaverse - Spectate does not work	
2841	Campaign Screen - All races should begin equally allied to Neutral Hexes	
2868	Dynaverse - Able to access buttons (campaign screen) anywhere on Hex map	
2869	Dynaverse - Enemy AI kills fleet member AI - Defeat with prestige	
2880	Dynaverse - Borg cubes appear very infrequently in Shipyard	
2309	Dynaverse - Destroyed enemy ship reappears on Hex map immediately	

2318	Tactical Sim - Visioneer opinion on buying Starbases
1681	Dynaverse - Officer advancement text cut off in message board
2452	AI doesn't know to go back to a repair station to repair hull
2981	Dynaverse - Severely damaged AI attacks healthy players
951	Dynaverse - AI does not team up properly in a Hex
2023	Dynaverse - Able to join missions in old Hex after leaving for new Hex
1790	SP - Tactical Sim - Able to click on map behind "Campaign Over" screen

The list of bugs and issues unprioritized

Bug ID	Bug Title	Priority
2929	CD-Key	A
2953	SP - Klingon Campaign - Beginning Stardate is 112400.1, twice what it should be	C
2979	Dynaverse - Fleets do not have accept / forfeit options in mission panel	B
2987	Dynaverse - Hex changed color to red when Fed was leader and Kling was member	C
3031	Dynaverse - Romulans can transfer in Borg officers	B
2561	Global - Freighter Convoys do not have escorts	B
2607	Campaign Screen - Player's ship gets stuck in Hex	A
2609	Tactical Sim - Fed vs Fed fights	B
2617	Dynaverse - Jumped from Lt. Commander ranking to a Fleet Admiral ranking	C
3110	Dynaverse - Ten turn countdown results in stuck in Hexes	A
2624	Dynaverse Campaign Screen - Fleet leader is not clear	C
2632	Dynaverse - Can make movement bar disappear when leaving a Hex with refit	C
2633	Dynaverse - While being attacked, attacking another will teleport player	
2637	DYNA - Map Screen not refreshing on completion of Mission	
2641	Campaign Screen - Ships inconsistent for Convoy between Attacking or Defending	
609	SP - Campaign Screen - States we are partners with the Contested Sector	
2058	Dynaverse - Cause of numbers appearing after player names in the chat box	
2119	New Conquest - Music stutters and pointer freezes loading new Conquest	
2701	Global - When AI forfeits it stays in the Hex	
2763	Dynaverse - Role of convoys	
2765	Access Server not using list of IP addresses	
2780	SP - General - Player can initiate a battle then auto move kicks in	
2784	SP - General - Player can be attacked when auto move kicks in	
2797	Hex information should appear in game display	
2799	SP - General - There needs to be a message when auto move is enacted	
2817	Dynaverse - "Stand by for mission briefing" panel repeats text	
2826	Dynaverse - Spectate does not work	
2841	Campaign Screen - All races should begin equally allied to Neutral Hexes	
2868	Dynaverse - Able to access buttons (campaign screen) anywhere on Hex map	
2869	Dynaverse - Enemy AI kills fleet member AI - Defeat with prestige	
2880	Dynaverse - Borg cubes appear very infrequently in Shipyard	
2309	Dynaverse - Destroyed enemy ship reappears on Hex map immediately	
2318	Tactical Sim - Visioneer opinion on buying Starbases	
1681	Dynaverse - Officer advancement text cut off in message board	
2452	AI doesn't know to go back to a repair station to repair hull	
2981	Dynaverse - Severely damaged AI attacks healthy players	
951	Dynaverse - AI does not team up properly in a Hex	
2023	Dynaverse - Able to join missions in old Hex after leaving for new Hex	
1790	SP - Tactical Sim - Able to click on map behind "Campaign Over" screen	

Prioritization starting (use A, B, C or 1, 2, 3)

Bug ID	Bug Title	Priority
2929	CD-Key	A
2953	SP - Klingon Campaign: Beginning Stardate is 112400.1, twice what it should be	C
2979	Dynaverse - Fleets do not have accept / forfeit options in mission panel	B
2987	Dynaverse - Hex changed color to red when Fed was leader and Kling was member	C
3031	Dynaverse - Romulans can transfer in Borg officers	B
2561	Global - Freighter Convoys do not have escorts	B
2607	Campaign Screen - Player's ship gets stuck in Hex	A
2609	Tactical Sim - Fed vs Fed fights	B
2617	Dynaverse - Jumped from Lt. Commander ranking to a Fleet Admiral ranking	C
3110	Dynaverse - Ten turn countdown results in stuck in Hexes	A
2624	Dynaverse Campaign Screen - Fleet leader is not clear	C
2632	Dynaverse - Can make movement bar disappear when leaving a Hex with refit	C
2633	Dynaverse - While being attacked, attacking another will teleport player	B
2637	DYNA - Map Screen not refreshing on completion of Mission	A
2641	Campaign Screen - Ships inconsistent for Convoy between Attacking or Defending	A
609	SP - Campaign Screen - States we are partners with the Contested Sector	B
2058	Dynaverse - Cause of numbers appearing after player names in the chat box	C
2119	New Conquest - Music stutters and pointer freezes loading new Conquest	C
2701	Global - When AI forfeits it stays in the Hex	C
2763	Dynaverse - Role of convoys	B
2765	Access Server not using list of IP addresses	A
2780	SP - General - Player can initiate a battle then auto move kicks in	A
2784	SP - General - Player can be attacked when auto move kicks in	A
2797	Hex information should appear in game display	C
2799	SP - General - There needs to be a message when auto move is enacted	B
2817	Dynaverse - "Stand by for mission briefing" panel repeats text	A
2826	Dynaverse - Spectate does not work	B
2841	Campaign Screen - All races should begin equally allied to Neutral Hexes	B
2868	Dynaverse - Able to access buttons (campaign screen) anywhere on Hex map	B
2869	Dynaverse - Enemy AI kills fleet member AI - Defeat with prestige	B
2880	Dynaverse - Borg cubes appear very infrequently in Shipyard	C
2309	Dynaverse - Destroyed enemy ship reappears on Hex map immediately	A
2318	Tactical Sim - Visioneer opinion on buying Starbases	C
1681	Dynaverse - Officer advancement text cut off in message board	C
2452	AI doesn't know to go back to a repair station to repair hull	C
2981	Dynaverse - Severely damaged AI attacks healthy players	C
951	Dynaverse - AI does not team up properly in a Hex	B
2023	Dynaverse - Able to join missions in old Hex after leaving for new Hex	B
1790	SP - Tactical Sim - Able to click on map behind "Campaign Over" screen	C

All tasks have received a priority.

Bug ID	Bug Title	Priority
2929	CD-Key	A
2607	Campaign Screen - Player's ship gets stuck in Hex	A
3110	Dynaverse - Ten turn countdown results in stuck in Hexes	A
2637	DYNA - Map Screen not refreshing on completion of Mission	A
2641	Campaign Screen - Ships inconsistent for Convoy between Attacking or Defending	A
2765	Access Server not using list of IP addresses	A
2780	SP - General - Player can initiate a battle then auto move kicks in	A
2784	SP - General - Player can be attacked when auto move kicks in	A
2817	Dynaverse - "Stand by for mission briefing" panel repeats text	A
2309	Dynaverse - Destroyed enemy ship reappears on Hex map immediately	A
2979	Dynaverse - Fleets do not have accept / forfeit options in mission panel	B
3031	Dynaverse - Romulans can transfer in Borg officers	B
2561	Global - Freighter Convoys do not have escorts	B
2609	Tactical Sim - Fed vs Fed fights	B
2633	Dynaverse - While being attacked, attacking another will teleport player	B
609	SP - Campaign Screen - States we are partners with the Contested Sector	B
2763	Dynaverse - Role of convoys	B
2799	SP - General - There needs to be a message when auto move is enacted	B
2826	Dynaverse - Spectate does not work	B
2841	Campaign Screen - All races should begin equally allied to Neutral Hexes	B
2868	Dynaverse - Able to access buttons (campaign screen) anywhere on Hex map	B
2869	Dynaverse - Enemy AI kills fleet member AI - Defeat with prestige	B
951	Dynaverse - AI does not team up properly in a Hex	B
2023	Dynaverse - Able to join missions in old Hex after leaving for new Hex	B
2953	SP - Klingon Campaign - Beginning Stardate is 112400.1, twice what it should be	C
2987	Dynaverse - Hex changed color to red when Fed was leader and Kling was member	C
2617	Dynaverse - Jumped from Lt. Commander ranking to a Fleet Admiral ranking	C
2624	Dynaverse Campaign Screen - Fleet leader is not clear	C
2632	Dynaverse - Can make movement bar disappear when leaving a Hex with refit	C
2058	Dynaverse - Cause of numbers appearing after player names in the chat box	C
2119	New Conquest - Music stutters and pointer freezes loading new Conquest	C
2701	Global - When AI forfeits it stays in the Hex	C
2797	Hex information should appear in game display	C
2880	Dynaverse - Borg cubes appear very infrequently in Shipyard	C
2318	Tactical Sim - Visioneer opinion on buying Starbases	C
1681	Dynaverse - Officer advancement text cut off in message board	C
2452	AI doesn't know to go back to a repair station to repair hull	C
2981	Dynaverse - Severely damaged AI attacks healthy players	C
1790	SP - Tactical Sim - Able to click on map behind "Campaign Over" screen	C

And now they are sorted.

Resource Leveling

In a real schedule it will be much more likely that the bulk of your core tasks will fit in your schedule but one or two of your developers have been overscheduled.

If at the end you have leveled the tasks the best you can and you are still left with an overloaded resource, then you will have to take their tasks and run them through a rigorous task prioritization session with the spreadsheet as I described above. Find the true core tasks and relegate the rest to a secondary phase.

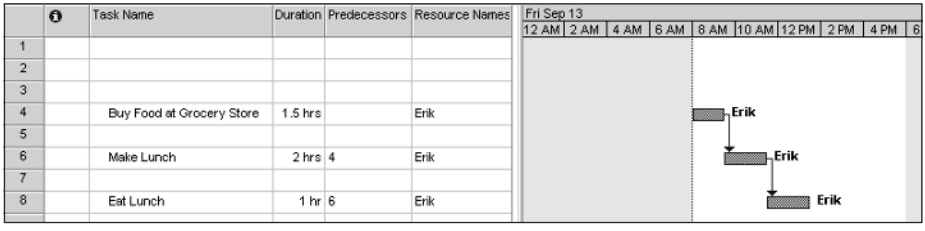
Hold on to your secondary and tertiary task lists. When you create schedules that your developers can accomplish, they will appreciate it and respond with timely execution. It is common for them to be excited and push themselves to see how many of the secondary and tertiary tasks they can pick up. See the next chapter on task tracking for more tips on how to keep your team humming along.

If you were conservative with your original labeling of core and secondary tasks and you did have a surplus of time, or if you had a surplus of time with part of your development team, then now is the fun time of piling on your secondary tasks until you are out of time with your resources. Use the detailed task prioritization method on the secondary tasks if you are having trouble deciding which of the secondary features you will implement.

Task Dependencies

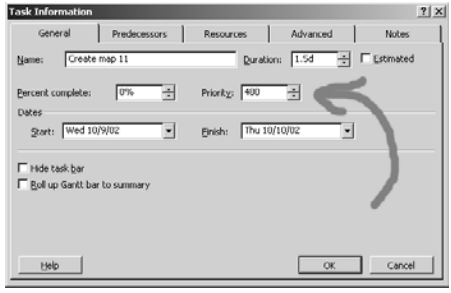
Creating the schedule is not too bad so far, is it? Painful decisions about what will be a core task and what will be a secondary task is about the only difficult job. A rather tedious job, I admit, entering tasks into Project, but mechanical and straightforward. Project planning enters a new level of complexity when task dependencies are taken into account. Task dependencies develop when one task depends on the completion of another task. A great example is all of your production tasks should be dependent on the completion of the preproduction milestone. After you have entered all your zero-level and core-level tasks (as well as any secondary tasks you found time for) you will now need to draw dependency lines between tasks that are *truly dependent* on each other. In Microsoft Project there are two easy ways to link tasks: One is to draw a link between two tasks by simply left-clicking on one task and dragging the pointer to another task and letting go. The other method is to simply type in the task ID number in the Predecessors column.

JARGON: *Dependent tasks* are two tasks that are linked such that work on task B cannot start without the completion of task A. This makes task and resource leveling more complicated.



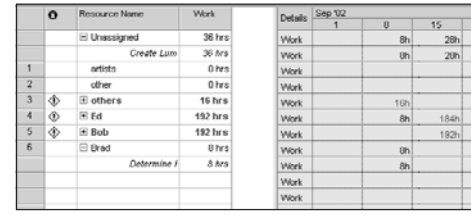
An example of linking tasks

Try not to link too many tasks; specifically, link only tasks that are dependent on each other. Some people, out of frustration with Project's leveling algorithm, start linking all kinds of unrelated tasks to get their project to flow in time the way they plan for production to follow. In other words, do not use the task dependency links to establish task priorities. Microsoft Project has a field for task priority for every task entry. Now run the Project leveling tool; if you are very lucky, all of your tasks will politely level out and none of your developers will be overscheduled due to the task dependencies you entered.



Setting the priority level of a task from 0 to 999

Most of the time, however, entering task dependencies will cause one or more of your developers to go over schedule. Now you will earn some of your salt; this is an area where it is difficult to give general advice that will apply to your specific overallocations due to dependencies.



The resource usage screen; the red numbers indicate an overallocated resource.

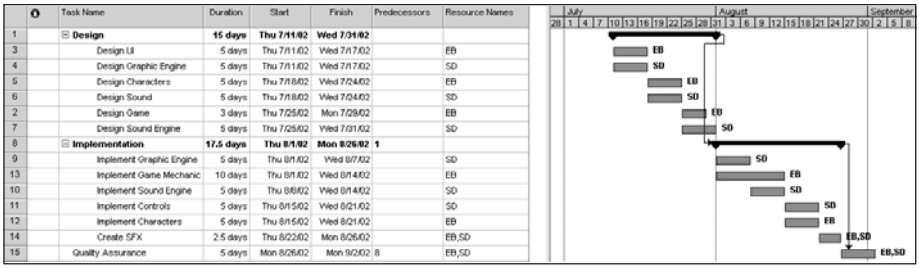
You first need to study the Gantt chart and the resource usage charts to understand what your dependency problem is all about. In all cases your developer was okay before the dependencies were drawn in from the earlier stage where you determined the zero-level and core tasks. So looking back up the chain of tasks you will see one or more tasks that are holding up the show for your overscheduled developer. This will create a pocket of free time for this same developer earlier in the schedule as he is stalled waiting for work. Now the most elegant solution would be if the work he is waiting for is something he could do himself; then you can simply assign it to him and fix the dependency problem. And in turn you will need to take some other work off this developer and exchange it with the original developer who was the bottleneck before.

	Resource Name	Work	Details	Aug '02					Sep '02			
				28	4	11	18	25	1	8	15	22
1	EB	344 hrs	Work	40h	40h	40h	40h	40h	40h	8h		
2	SD	240 hrs	Work		32h	40h	8h			32h	40h	8h
			Work									

The resource usage report shows holes and gaps indicating a problem of one resource waiting on another.

	Resource Name	Work	Details	Aug '02					Sep '02
				28	4	11	18	25	1
1	EB	284 hrs	Work	24h	40h	40h	40h	40h	4h
2	SD	300 hrs	Work	40h	40h	40h	40h	40h	4h

The gaps filled by task reassignment



	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Design**	**15 days**	**Thu 7/11/02**	**Wed 7/31/02**		
3	Design UI	5 days	Thu 7/11/02	Wed 7/17/02		EB
4	Design Graphic Engine	5 days	Thu 7/11/02	Wed 7/17/02		SD
5	Design Characters	5 days	Thu 7/18/02	Wed 7/24/02		EB
6	Design Sound	5 days	Thu 7/18/02	Wed 7/24/02		SD
2	Design Game	3 days	Thu 7/25/02	Mon 7/29/02		EB
7	Design Sound Engine	5 days	Thu 7/25/02	Wed 7/31/02		SD
8	**Implementation**	**17.5 days**	**Thu 8/1/02**	**Mon 8/26/02**	1	
9	Implement Graphic Engine	5 days	Thu 8/1/02	Wed 8/7/02		SD
13	Implement Game Mechanic	10 days	Thu 8/1/02	Wed 8/14/02		EB
10	Implement Sound Engine	5 days	Thu 8/8/02	Wed 8/14/02		SD
11	Implement Controls	5 days	Thu 8/15/02	Wed 8/21/02		SD
12	Implement Characters	5 days	Thu 8/15/02	Wed 8/21/02		EB
14	Create SFX	2.5 days	Thu 8/22/02	Mon 8/26/02		EB,SD
15	Quality Assurance	5 days	Mon 8/26/02	Mon 9/2/02	8	EB,SD

The Gantt chart of the fixed schedule

If exchanging and rearranging tasks still leaves you with a pocket of dead time and a later overallocation of one of your developers, then you will have to trim off the overallocation and bring up a secondary task to fill the void. That will be the best you can do if all other efforts to exchange, rearrange, and distribute the overallocated tasks fails.

The Top Ten Risks Document

By far the schedule is the major deliverable of the project plan, but there is one more document that is critical: the top ten risks document. For this document enumerate the ten most significant risks to the project. Choose only ten items; a longer list will lose its focus. With each of the risk items also list what actions you have taken or will take to contain or address the risk. Hopefully you will be able to create a positive solution to each of your risks; however, that is not a requirement. The important thing is to create a short, focused document with one through ten of your risks that you can share with your executive management and with your development team.

This document should be maintained with delivery of each of your development milestones from preproduction to the game's release. You will then see a much greater awareness from your executive management of the risks, and you should be able to address these risks with more energy. In fact, these short top ten risk documents are the most effective way I have found to communicate to my executive management just how much I need something: another programmer, two more artists, or timely audio asset delivery.

DATE: 3/1/02

Rank	Risk	Effect	Solution
1	Mission design slips	slip	Finalize Missions ASAP
2	User interface design slips	slip	Finalize UI ASAP
3	QA resources added late to the project	low quality, slip	More QA Resources earlier
4	Voice-over assets delivered late	slip	Finalize dialogue ASAP
5	Feature creep	slip	Stop adding features
6	Late solicitation to beta testers	slip	Submit to beta testers earlier
7	Server stability	low quality	Create testing tools
8	Design process overly distributed	washed out quality	Reduce number of authorized designers
9	Overextended use of overtime	slip, low quality	Address slip issues
10	UI overcorrected for mass appeal	lack of distinction	Fewer designers

A top ten risks document

The Non-Zero Chance of Delivery

At the end of the day your job as the project planner is to create a plan for how long it will take to get the job done, *not the earliest possible date with a non-zero chance of delivery*.