Level Design Goals and Hierarchies

3

Just like in any other creative endeavor, it is important in level design to define a set of clearly understood goals before any major work is done. However, this is easier said than done, if there is no consensus on just what those goals may be. Can we even speak of goals specific to level design, or are such goals related to the game design as a whole? Do level designs for differing games share similar goals? Should we even try to formalize a set of specific level design goals?

The longer we look at these questions, the clearer it becomes that they are important and fundamental questions that need to be addressed early in the level design process. This chapter will address a number of level design goals that are important to keep in mind.

Success Definition

What are you trying to achieve?

This is a question that should always have an answer. The *why* and *how* aspects of the same question are addressed separately, but those are all moot if the *what* question is not answered first. Being able to define clearly what it is you are trying to achieve means that you can clearly describe a number of goals that need to be met. That is to say that any goals that are set are subject to an overall *success definition.* This means that the goals are not arbitrary, but instead are set to achieve a desired overall result, which is a *definition of success*. A goal is just part of a strategy for achieving this successful outcome. This may seem too obvious to state, but in my experience this most basic element of level design is often forgotten.

The success definition(s) can be game-wide[1]; indeed a clear definition should exist at the top of a game's *hierarchy* of goals. It can be level-wide, or can

[1] Some game companies refer to such a game-wide goal summary as *The X*.

even occur on smaller gameplay scales. For example, a hypothetical level-wide success definition could state:

"Train players' combat skills throughout the level and build up the players' level of confidence and ability in order to prepare them for the introduction of a new enemy creature."

A game-wide success definition might read:

"Create an action-horror hybrid that can be described as "Dracula meets The Terminator.[2]"

In our case the success definition describes what the *level designer* needs to achieve. This is something that should remain in focus as the ultimate goal throughout the level design and level creation process. All other goals are by definition subordinate to that overall aim. Whatever the final success definition(s) incorporate, we need to be aware of what they are.

External Level Design Goals

There are all kinds of goals that may need to be taken into consideration when determining success definitions. Some are independent from direct gameplay considerations, insofar as they that they refer to goals outside of the gameplay experience. The level design may have to

- be appealing to both genders,
- stay true to the brand image,
- sell X amount of units,
- push new engine technology,
- win a BAFTA award,
- receive a meta critic[3] rating higher than 75%.

These are the overarching goals that are derived from the requirements of the game's development as a commercial or even artistic production. They describe the purpose of the level design components as much as they define the internal level design goals, which are derived from direct gameplay considerations. It is a mistake to dismiss these external factors, as they are the ultimate reason why a game is made in the first place.

Good examples of external goals like this can be found in the marketplace, so let's take a closer look at commercial considerations:

[2] I have seen worse pitches.

[3] See www.metacritic.com.

Commercial considerations

It is important to note that, aside from alternative funding models and independent game development, in most cases, games are supposed to make money. There are many ways one can try to achieve this goal, but it should not be forgotten that for many people it is the fundamental reason for developing a game in the first place. It is wise to note this simple fact. In the case of publisher-funded development, for example, a game that does not sell is likely to be seen as a failure by the financing publisher. *The most important reason that publishers fund game development is because they think it is likely to make them money*. I am not saying this is bad or good, but merely stating a fact that offers another point of view on the function of a game and, by extension, of its design. If the game design within this context doesn't support a commercially viable end product, it has failed to fulfill its basic function *from the publisher's perspective*. This is a possible interpretation of game design that, even if unwelcome[4] at times, nonetheless needs to be understood.

Only after a game's creation has been approved through these external goals can the production go ahead. However, these goals do not in themselves determine fun gameplay, which is the realm of internal level design goals.

Internal Level Design Goals

As stated earlier, internal level design goals *are derived from direct gameplay considerations*. Outside of the external macro goals described earlier, there will always be level design goals that are just part of the art form itself, driven by internal factors. They are the goals that contribute to the overall good gameplay. They deal with requirements such as the following:

- Empower the player.
- Teach the player how to have fun with the game.
- Don't break the player's suspension of disbelief.
- Give the player a sense of achievement.
- Reward the player for exploration.
- Provide addictive, fun gameplay.

These are requirements that stem from supporting the game or the gameplay on a very pure level. They are as important to define as external factors.

It is important to never forget that it is the job of a level designer to implement, interpret, and exploit the game design to give it its best possible applica-

[4] It is good to note that commercial considerations do not have to be crass or in conflict with artistic merit.

tion. A game's *level design* is not an isolated construct, just as *a game design* does not function independently. They need to complement each other, or both will suffer. (See Chapter 1, "Game Design vs. Level Design.")

Internal level design goals are the bread and butter of our profession. They give us very specific goals to aim for in order to provide a good gameplay experience. In a hierarchy of internal goals, the game's *overall gameplay experience* always comes out on top, and the level design needs to support this.

This also means we have to start putting our level design goals into some type of *hierarchical* order.

Level Design Hierarchies

We looked at level design goals before and determined that they must support and put into practice the game design. This is easier said than done, of course, since the level design components will have to fulfill a number of requirements. This is their *purpose*. Nonetheless, let's assume that we have a clearly defined set of goals, and that we have translated them into requirements.

So how do we know what value to assign to these goals and requirements? What components do we focus on first when we start working on a new level? Which elements are more important? *What if it doesn't work?*

These are all typical questions that come up during a game's development. They are all related to the component's importance and place in the project. To determine these factors, we first have to take a step back and devise a way to assign value to individual level-design components. We have to give them a place within a logical *level-design hierarchy*.

Development Hierarchy

To create a situation where we can measure the value and implementation of such things as AI encounters, item placement, puzzle design, and hundreds of other level design components, it is helpful to define some kind of hierarchy. If we can figure out where things fit in the scheme of things, we can make choices on what and how to prioritize, and where dependencies occur.

In a game-design setting, here are a number of known stages and areas of development. They tend to fit into a structure where one area supports another in a certain hierarchical order.

Generally, we have to look at the following questions:

- Why are we making the game?
- What game are we making?
- How are we making it?

Figure 3.1. Game-wide hierarchy.

When we have answered these questions and have logically placed our general development components in this framework, we can see a hierarchy emerge. For some games, this general hierarchy may look like the image in Figure 3.1.

In this pyramid shape, each segment is supported by the segments below it, and were we to try to answer our why, how, and what questions with it, the following would emerge:

- Success Definition (Why),
- External Goals (Why),
- Internal Goals (Why),
- Game Design (What) – Level Design (What/How),
- Technical Design (How) – Art Direction – Audio Plan (How).

This diagram gives us a possible insight on where level design exists within the overall context of the game's development. I won't go into too much detail when it comes to production issues, as that is not the focus of this book. However, it is at least interesting, insofar as that it shows some relationships between level design and the rest of the game's production areas.

Things become more relevant when we try to find a similar approach within the level design parameters for a game. We might ask those questions again:

- Why are we making the level?
- What levels are we making?
- How are we making them?

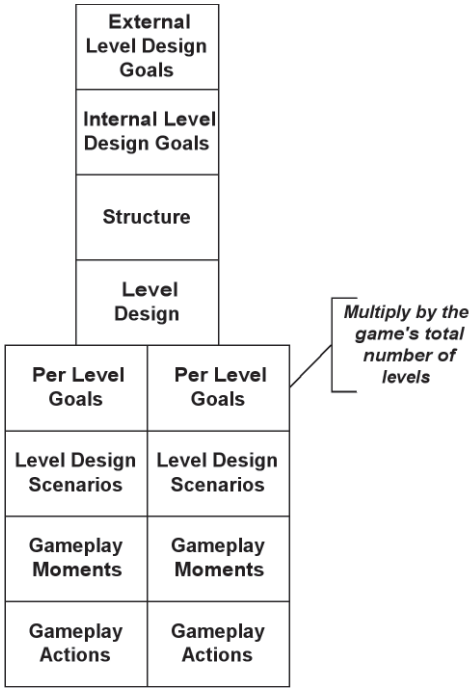

Figure 3.2. Level-wide hierarchies.

To answer those questions, a game's level design hierarchy might be made up of the elements shown in Figure 3.2.

And to repeat the exercise once more: in this pyramid shape, each segment is supported by the segments below it, and were we to try to answer our why, how, and what questions with it, the following would emerge:

- External Level Design Goals (Why),
- Internal level Design Goals (Why),
- Structure (What),
- Level Design (What),
- Per-Level Goals (How),
- Level Design Scenarios (How),
- Gameplay Moments (How),
- Gameplay Actions (How).

From this point on, we can go into real detail and start to assign things like function and resulting tasks to the broken-down elements.

Dyson, a Case Study

To illustrate some of these concepts against a real world example, I am going to use the game *Dyson,*[5] designed by Alex May and myself, as a case study. *Dyson* was created in a one-month competition for the *independent games focused* website and community *TIGSource*. The game development competition was defined by the following rules:

> The theme of the game is PROCEDURAL GENERATION, with an emphasis on generating new content every time the player starts the game. Procedural generation, or procedural content generation, is a term to describe the creation of content algorithmically "on the fly" (i.e. after the software executes, rather than before). In video games it is generally used to either generate "random" content (like graphics or levels), or to reduce file size. This competition is focused on the former, but you may choose to apply the theme however you see fit.[6]

With this background in mind, we can now examine the *why*, *what* and *how* questions and the level design hierarchy they imply. In the case of *Dyson,* all these elements are represented, as discussed below.

External level design goals

Dyson's levels first and foremost had to show off the concepts of *procedural generation,* as defined by the competition rules. They had to do so in such a way that they still ended up being enjoyable to play, without the procedural nature being a gimmick. In other words, the procedural elements of the game needed to be explicit in the levels and the gameplay they would hold.

All this had to be achieved within 30 days with a very limited amount of time and resources. (The team consisted of only one programmer, one designer, and one audio designer.)

Resulting strategy. To meet these goals, we decided that *Dyson* would be a game that Incorporated scientific theories that show off procedural generation in a gameplay format. Freeman Dyson's *Dyson trees* and *Astrochicken* met John Von Neumann's self propagating machines, and everything was turned into a design that allowed small seeds of data to grow into complex environments.

---

[5] This is not meant to promote the game, but simply to provide a clear example of the concepts we have just discussed. A fully featured version of the game is still being developed at the time of writing this book by Alex May and myself, with music and audio being done by Brian Grainger. Have a look at it here: http://www.dyson-game.com.

[6] "Procedural Generation Competition," *Procedural Generation Competition,* http://forums.tigsource.com/index.php?topic=1646.0, 2008.

To restate that in understandable English: *Dyson* was chosen to become a game of space conquest. Flying robot seeds conquer territory and exploit resources in an asteroid belt. Other seedling factions compete for the same resources, and almost all the game art was to be procedurally generated, rather than modeled.

Internal level design goals

To meet the external goals, we decided that the levels had to provide strategically interesting choices despite a very limited amount of resources at hand. This meant that the levels needed to work as a replayable experience, where players would get different content every time they play.

Resulting strategy. To provide enough variation within a limited number of levels, we decided that almost all factors in the game had to be affected by a select number of variables. To do this, a limited number of variables had to lead to a maximum sense of interesting complexity.

To facilitate this in level design terms, we decided that the levels had to work in two dimensions, and that all territory was governed by simple and recognizable attributes. In the game, the seeds are planted on asteroids, and they turn into trees whose genetic makeup is influenced by the asteroid's variations in natural resources. The trees produce seedlings, which are also affected in their abilities by the mother asteroid, and they can in turn be used to conquer other asteroids.

Structure

Because of the need to make the most out of very limited resources, the levels for *Dyson* had to be replayable. The level structure therefore had to support open-ended strategic choices from the beginning of play, in order for multiple playthroughs to feel very different, depending on the choices that were made by the player early on.

Resulting strategy. To achieve this, we decided to create one very large level, with interesting gameplay complexities introduced by AI agents that all vie for the same resources as the player. Because the resources and layout of the asteroids change every time the game starts, replayability is guaranteed.

Per-level goals

In this case, only one level was chosen to be in the game, but it had to have the appeal and scope to keep the player interested for as many playthroughs as possible.

Resulting strategy. For this to work, the gameplay goal of the one level we ended up with for the competition had to be one of *total conquest of a territory*. The win/lose conditions for a level had to be simple, so that players could easily understand the task at hand and try again if it didn't work out.

Gameplay scenarios

The level in question could only maintain its long-term appeal if a number of discreetly interesting gameplay scenarios would play out time after time, without losing appeal. The level had to be balanced in such a way that these kinds of gameplay loops would occur often enough and be enticing throughout.

They had to be those kinds of moments that operate at a level where the player is trying to find and perfect success strategies.

Resulting strategy. Out of the possible scenarios to evolve from the core gameplay elements, we focused our attention mainly on the following ones:

- Invest and grow: Choose to keep seedlings close by and use them to plan more Dyson trees).
- Grow and expand: Try to create an army of conquest and take over other territory.
- Choose units and attack enemy: Focus on the variations in seedling abilities and pick seedlings best suited to defeat other seedlings in the game.
- Defend: Take a defensive approach and try to hold territory while repelling attacks from enemy factions.

Gameplay moments

To have lasting appeal, gameplay scenarios need moments of excitement or must otherwise be engaging. These are the decision moments that involve players at a deep level and have them watch the game unfold with real interest.

Resulting strategy. We tried to make sure that enough of these special moments would occur to serve our other goals in the level-design hierarchy. We focused on gameplay moments that included

- choosing a strategically strong asteroid to attack,
- finding the right moment to sacrifice seedlings and plant a new tree,
- setting up forces, to swoop to a heavily fortified enemy asteroid,

and so on.

Gameplay actions

Finally, the immediate actions the player can perform had to be balanced against the overarching goals and had to fit the environment. Fundamental level-design choices had to be made with basic actions like

- planting a tree,
- sending units to a target.

Resulting strategy. To keep things interesting for players, a number of restrictions, or conversely abilities, had to be considered with regard to those gameplay actions. For example, *planting a tree* had to be balanced by an associated cost; in this case a number of seedlings had to be sacrificed. (This had the added advantage of making seedlings work as currency). On the other hand, *sending units to a target* was made to be as easy as possible; all units' seedlings orbiting an asteroid could be sent off simultaneously.

To facilitate the use of these actions, and to serve several of the other goals of the game, we also decided to include tutorial text that helps the player pick up the game rapidly.

Note that it is surprising how easy it is to fit all the disparate elements of a game and its level designs in a fairly sensible and logical hierarchy. The fit is not 100%, and some elements have been left untouched. For example, music and audio play an important role. However, the example illustrates how it is possible to construct a view of the level design process where each individual feature is shown to support many others. The interrelationship within such a hierarchy is useful and shows up important areas to focus on when designing and implementing levels.

X Factor

Another way of looking at the concept of a success definition is by dubbing it the *X-Factor*. I am highlighting this subject, as it often doesn't get the attention it deserves.

A good game is more than the sum of its parts. This can be seen very clearly in some very famous example, for instance in *Grand Theft Auto: Vice City*,[7] which was a phenomenal success, both financially and creatively. However, if studied in isolation, many of the individual game components were lacking in some way or another.

[7] Developed by Rockstar North.

- The aiming mechanic was cumbersome.
- The third-person character controls were frustrating.
- The camera often couldn't cope.
- The geometry and architecture were very low on detail.

There are many other examples as well, but the key point is that deficiencies or excellence in these areas do not necessarily make a bad or good gaming experience.

Many people have tried to copy *Vice City* and have even improved upon individual gameplay mechanics, yet have come away with less enjoyable games. Apparently it is not that easy to copy gameplay mechanics and end up with something of equal quality. The answer lies in the fact that there is an art *and* a science to games development, and that both have a strong impact on the final game. (I am using the term in this context as something different from the art direction). The science is the pure implementation of the required mechanics, but the art comes into play by taking those gameplay mechanics and making them work as effectively as possible within the game's success definition (including the game's level design goals).

To go back to *GTA*: what the game does very well, however, is to place all these individually lacking gameplay elements and put them in the service of the game's ultimate goals, both internal and external. Players forgive the frustrating moments, the repetitive NPC dialogue, the camera issues, and so on, because they are still successful in achieving those goals that the level designers and game designers have put at the top of the hierarchy.

Additionally, what gives *Vice City* added appeal is how all its disparate elements have been made interesting by giving them cultural relevance. The music in the game, the references to 80s pop culture, and the humor, can all be seen as *essential* contributions to the final appeal of the game. It is also interesting to see that they are all integrated in the level design of the game in a way that is deep and practical.

In practical terms, the game successfully combines the art and science of making games. And by applying all these things expertly in its level design, mission-based or sandbox, it manages to fulfill its game-wide and level-wide success definitions and hit the target of the elusive x-factor that so many games strive for.

A Modular Approach to Level Design

So now we know that is useful to try to place our level design components within the context of an *overall level design hierarchy.* A good next step is to make sure

we can have level design components (modules) that are multi-purposeful and effective.

The *modular* aspect of this approach comes from the idea that once you determine what the *function* of a component is, you have more freedom in the way you try to hit the requirements that this functionality implies.

A classic problem with inexperienced level designers is that they have insufficient grasp of the overall purpose of the particular level design task they are working on. It is easy to get trapped in problems due to tunnel vision, I certainly have done so myself on many occasions early on in my career. When this happens, the micro details of a particular level design component become more important than its actual function in the grand scheme of things. Suddenly, that ambush encounter you are working on HAS TO BE about using a particular weapon or enemy, while it may be entirely possible that that encounter can fulfill all its requirements while being staged against a completely different backdrop or by using an entirely different class of enemies.

Realizing that this danger exists and always asking oneself, "What is the function of this component?" goes a long way towards avoiding that trap.

Contingency Planning

Not only does this way of thinking help in designing levels in a more effective manner; it also provides great scope for contingency planning. This is something that within the high-pressure environment of commercial game development is extremely important.

It allows us to create level design components that are scalable. For example, if we know that a certain combat situation in a level is mainly included in order to introduce a new gameplay mechanic to the player, we can of course try to make the event as spectacular as possible by introducing complex scripted events, incorporating intricate enemy behavior and pulling out all the bells and whistles. This section will probably "read" very well on paper, especially early on in the development cycle, when the level designer is assured that the creature AI would eventually be more than up to the task of supporting such an amazing encounter.

Now let's fast-forward twelve months in time. We are now two months away from having to ship the game, everybody is severely stressed and overworked, and unfortunately, the AI of this particular enemy had to be scaled down several levels due to time constraints and will no longer be usable within the original plan for the level design. *Oh dear, what now..?*

This is an example where it is extremely important to realize why the encounter exists in the first place: *to introduce a particular new gameplay mechanic*. With

this overall goal in mind, we find that it really does not matter through which AI encounter this gameplay mechanic is taught and introduced to the player. It can be substituted and made to work by using existing and working game assets, and the final value to the game as a whole will be the roughly the same.

This may seem completely obvious, but it is very easy to get lost in insular goals.

Anybody in games development is constantly bombarded with the following question: *"Wouldn't it be cool if we… (Insert idea X)?"*

As it turns out, idea X may well be cool, but does it really need to be there? Can it be done in a different and more cost-effective way?

Those are the kind of questions you should ask yourself if you want to be able to apply level designs with enough flexibility that you can react to unexpected circumstances.[8]

Application of Modular Hierarchy

If we take all that we have said before into account, we should now have a practical framework for our criteria and requirements that has great value. We should be able to take any game component, find a place for it in this hierarchy, and be able to ask the right questions regarding its place and value in development. This is no minor thing and can save a difficult project from disaster if done correctly. It means that there is much less dependency on specialist or bespoke solutions, and that the developers can be much more flexible about the implementation of the designs without compromising too much on quality.

Conclusion

Hierarchies can offer a very useful tool. They give us a way to break level designs down into individual components and formulate a plan in which all of the level design components fulfill their ultimate requirements. Furthermore, it allows us to check the value and importance of individual components and refer to their place in an overall hierarchy.

Viewing the level design process through hierarchical steps may not always be a perfect fit. For one thing, one level design hierarchy may look different than another, and your mileage may vary between different projects. But it is always useful to try to find a way to define the *why*, *how,* and *what* questions within a logical order or hierarchy that allows you to make judgment calls on the implementation of your levels.

[8] You should always expect unexpected circumstances.

This gives us the practical means to assess what function any individual component or element fulfills, what it relates to, and how important it is. This has obvious uses that go beyond the need of the level designer; it also helps in establishing clear goals and criteria that are understandable to anybody involved in the development of the game.