Modular Level and Component Design

Or: How I Learned to Stop Worrying and Love Making High-Detail Worlds

So the future has finally arrived. Technologies we've only dreamed about are closer than ever, and if you're lucky enough to be on the right project, you're knee deep in it already. With polygon counts off the charts and hardware pushing more detail than you care to create, environments can be simply staggering. But with every step forward in graphics capabilities, many find themselves wondering how to utilize those advances. The question of "Wouldn't it be cool if the engine could do feature X?" is now rephrased, "Do we have the talent to utilize feature X at all?" Nowhere in the industry is that question more apparent than with environment design.

When we can draw limitless detail, it seems limitless talent is required to create those worlds. How do we quickly generate enough consistent quality content to fill a highly detailed world? How do we make sure those worlds can be easily modified and made flexible to design whims? How do we break up the team to handle these tasks? These are daunting problems to address, and even the most stalwart industry veterans will find themselves intimidated quickly at the prospects.

As we stare down the future, it may help to take a page out of history, from tile-based platform games. Much like Super Nintendo worlds of old, it will inevitably boil down to modular components in some way. Until recently textures provided the bulk of a world's detail, but for the foreseeable future, more and more players will expect that detail to be full-blown geometry. That's a big order to fill, particularly in the competitive realm of first-person shooters, where eye-candy often reigns supreme. However, the solutions to these problems are not limited to the worlds of the running guns, they can solve dilemmas for many genres and many settings.

Using prefabricated geometry in creative ways is key to achieving the detail levels players expect to see in their game worlds. This article will look at how we at Epic arrived at the modular solution, how we implement such a system, and the benefits and limitations of a component-driven workflow.

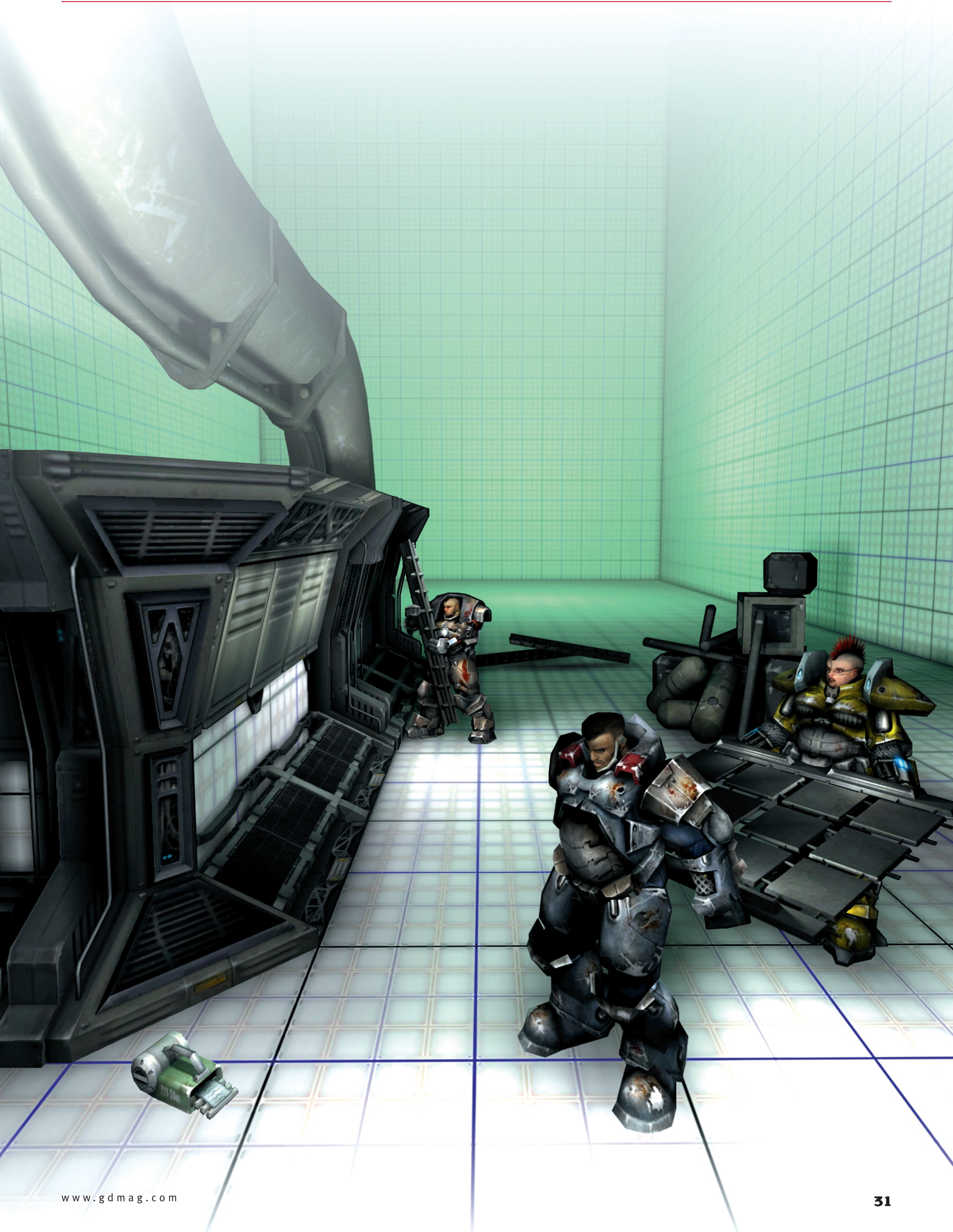
You May Ask Yourself, How Did I Get Here?

The modular level design solution arose from the need to have great-looking, high-detail levels without having to build and texture every nook and cranny of the environment. Asking a traditional level designer to create an environment (usually from simple tools) that holds up to artist-like scrutiny is not a practical idea. Many level designers aren't familiar with high-end modeling packages, and even knowing the tools doesn't necessarily translate to creating great content. The days of aligning textures and moving on have themselves moved on.

Asking traditional artists to design an engaging level is not an ideal solution either. Doing so may produce great-looking yet highly unpredictable gameplay. Questionable gameplay and performance sacrifices arise regularly when artists focus on creating large masterpieces. And both systems easily end with the creator losing inspiration after weeks of working on the same content.

Another system many developers work with is having a dedicated level designer block out an environment and pass that progress on to an artist for final polishing. This works to a degree, but it can still benefit from modular design decisions made early on. Modeling and texturing an entire level at

LEE "EEPERS" PERRY | *Lee is currently a designer and artist for Epic Games. Among numerous other projects, previous work includes art and level director for Ion Storm's ANACHRONOX, and modeling for Square's R&D department. He can be contacted at lee@epicgames.com.*



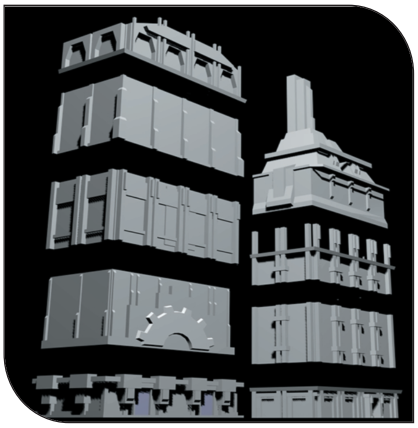


FIGURE 1A. A larger-scale building set.

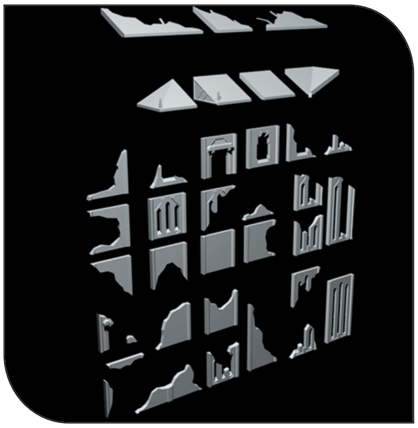


FIGURE 1B. The scale has been taken down a notch, using more components in a single wall.

extremely high detail levels can result in many months of work that may end up needing significant retooling. Eventually someone undergoing such a task will end up aiming to reuse as much artwork as possible from one area to the next; why not keep that tendency in mind from the beginning and work with it as a foundation for your creative process? Following are some of the considerations we keep in mind when designing our environment construction process around modularity.

Scale

With intelligent planning, modularity is not as restrictive as you might think. Indeed, modularity can be implemented on many scales. Some may choose a path along the lines of MORROWIND, wherein large buildings are single prefabricated items, and entire castles may break down into a handful of wall pieces. Other projects, with an eye on very dense environments, may work on a smaller scale that incorporates several modular components to create a single wall's detail, as shown in Figures 1a and 1b.

Where do you start when building a set of prefabs? First you must decide the scale of modularity your project is going to need. This can generally be determined by the scope of your environments. If you're going to be shoving a player through entire cities in a Porsche, you'll want to break everything down into chunks of geometry as large as entire city blocks. If you're going to be creeping a single player through the narrow, abandoned corridors of a derelict spacecraft, you'll want to work at a very fine detail level with layers of components interacting in a complex fashion.

Grids

Regardless of scale, when creating a set of modular components, nothing is as important as the almighty grid. Grids may fluctuate wildly between engines and projects, but whatever system you decide on, stick to it religiously, and always use even divisions of that grid when working on smaller components. If your system dictates that a generic wall be 256 units tall, try to keep smaller details at 128 units, or 64 units, and so on.

Be aware of game mechanics when determining the system. If a character is 128 units tall, or a leaping character can cover 64 units forward, your system should take advantage of those values. Working on a gameplay-conscious grid aids the level design process greatly, while significantly reducing "trial and error" early testing.

Keep animations in mind. Characters may interact with computer panels, use door handles, be able to sit on surfaces, or perform any number of other interactions. Deciding in advance at what height various animations will happen can save you a great deal of pain later. Of particular importance here is standard stair heights and depths. Lock down these standards early, print up guides, and distribute them as soon as possible. A few hours' work here will save weeks of potential headaches during later production.

Planning

Now's the time to sit down with the designers and artists responsible for a given area of the game and discuss what type of pieces will be required. Make a list of key features for the environment and the unique components that define the level. These could be anything from a vast missile hangar to a complex volcano mesh.

Leave the specialized pieces aside, focusing on the meat of the environments, the areas where players will spend the majority of their time. Decide which key modular components are needed and plan out the general use of those pieces. Make a shopping list of components needed for a flexible base set; this could range from hallway sections to generic city streets.

The challenges in this step often come down to transitions. Plan out how to blend one theme into another, and save effort by clarifying which transitions won't be needed. Note which structures may need capping off, and give a thought to how that should be done. For example, if you have a modular river running through your game's landscape, have a method handy for capping off that river by making it run into a grate or sinkhole, or dissipate over a waterfall. If you're building entire buildings from prefab corridors, make sure you've got a piece to cap off that structure.

Your system could have conceptual artists start their work at this point, or they may be involved earlier and have concepts presented when the initial shopping list is created.

Start with the Basics

At this critical point, be sure to go through the prefab list methodically, building the necessary components on the agreed-upon grid system. Don't begin construction with a complicated junction piece, start with the basics and create the base unit from which variations can be created. If you're making a 3D terrain set, start with the base tile, duplicate it, and create variations off the mother piece. After the various themes and sets are completed, lay them out and begin "tweening" them and creating any transitional prefabs needed. Establish an approval process at some point in the basic construction, so you don't end up with a polished set of geometry only to find out the basic theme isn't quite right. Sign off on work along the way.

Test-map your work by periodically dropping what you have created so far into a test environment and verify that your work tiles correctly and works well together. As a bonus, you may find it inspirational to see complex geometry emerging as you work.

If your project allows it (and most do) try to keep multiple purposes in mind for your work as you go. A cave system may have ceilings that could easily be used as floors. Typical metal-plated bulkheads will often look good regardless of orientation.

Don't assume a large structure has to be a single modular piece. For example, instead of building an entire modular corridor, build modular wall sections, ceilings, and floor tiles that snap together to form a complete corridor. This will allow for more flexibility when the set gets into the designer's hands.

It also helps to provide plenty of variations. If you have a large stairway piece, make short, medium, and large versions for added flexibility, and be sure to consider texture variations as well. With a change of color scheme, an office hallway may appear completely different. By incorporating relatively minor variations, an area can feel far removed from another area that uses many common elements.

Find out in advance if your prefab sets will require back-facing polygons. Leaving the back of a prefab hollow guarantees someone will make an environment to display it.

Mirror, Mirror

Designers will love you for creating a flexible set that is painless to work with. Of key importance to achieving that flexibility is making sure your geometry can be mirrored easily on as many axes as possible, so keep a line of symmetry through your work as you go.

Cylindrical structures are an important spot in this regard. It's a good idea to give objects such as pipes a number of sides divisible by four. Initially this may seem odd, but the first time you have a row of pipe sections and try to pivot a seven-sided length of pipe 90 degrees, the logic will become apparent. Seams appear, edges don't line up, and things get messy quickly. With an eight- or 16-sided pipe, you can mirror or rotate an elbow easily without risks of seams.

When working with symmetry in mind, you'll want to restrict any lettering that can't be mirrored to detail pieces. Try to create pieces that work when mirrored vertically as well as horizontally, and consider creating transition pieces that will allow an object to be mirrored to itself for added flexibility.

Origins

You may have a prefab modeled rigidly on a grid, but if your origin is off, it may come into the engine off grid as well. To what degree this happens depends on your particular technology, but be aware of it as you build your meshes in world space. A handy trick for planar wall sections is to place the origin in a corner of the mesh, instead of the center. This allows the wall prefab to be stretched or pivoted around while one corner remains on grid. An origin in the center of a rotated wall leaves both edges off-grid and makes it difficult to line them up again (see Figures 2a and 2b).

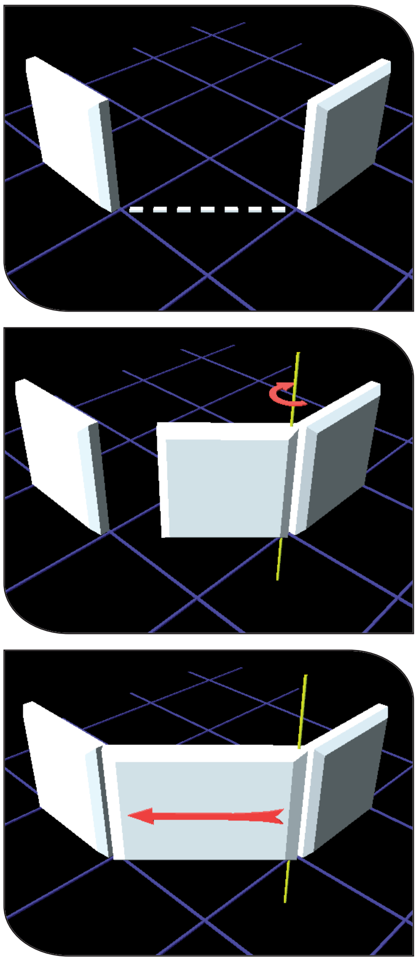


FIGURE 2A. With an origin in the wall's lower corner a wall can be rotated and stretched to fill an angled gap easily, while the origin stays on-grid.

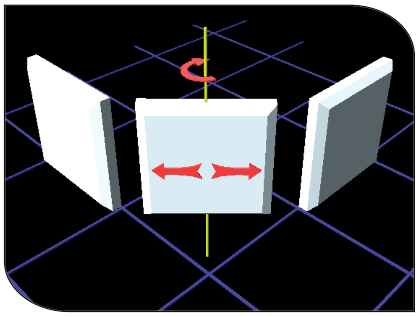


FIGURE 2B. A centered origin results in having to move off the larger grid to cover an angled gap, scaling in multiple directions, and creating seams on both walls thereby using more components in a single wall.

Don't Forget to Accessorize

The next step in the system is to create a handful of accessory pieces that can be used to break up the appearance of tiling. A castle hallway created from modular wall sections can look entirely unique with the placement of a few busted overhead beam meshes, a worn statue, or loose stones from a collapsed wall. It doesn't take much to fool a player into thinking two essentially identical corridors are unique places; their differences just need to have enough visual impact to count.

A good technique to use when creating accessories is to build them in a further modular fashion. If you're planning to build three different detail statues for the castle in the preceding example, you could separate the statue meshes into a lower section, a torso, and a head. Mix and match those parts, and with a little planning before diving in, your castle can be filled to the parapets with unique statue variants in every hallway.

Prefabs can be very flexible if built in a way that frames rough level geometry. Use a shell system like the one show in Figure 3 to give otherwise plain geometry a more detailed profile.

You can also create accessories expressly with the intent to conceal various kinds of level work. Your system might match up very well overall, but it's a good idea to have some generic concealment pieces handy. For example, if you're working with a natural-looking rock wall system, there may be instances where the walls are used at odd angles and cause seams. Example concealment pieces might be a rustic beam that can be placed over the intersection, a large sliver of a stone slab, or perhaps some foliage.

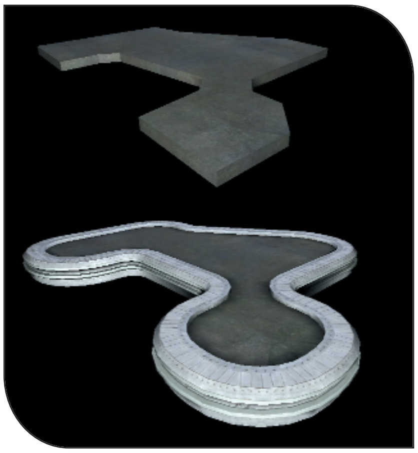


FIGURE 3. Modular boundary systems can give simple level geometry great detail while maintaining flexibility.

Custom Pieces

After you've got a base set and enough accessories to embellish the environment, the next step is to create custom areas that the design calls for. It's important that this step come after the basics, as you'll often need information such as how the basic set will transition into the custom pieces. If the piece needs to fit very specifically, you or the designer may consider blocking it in first and using that work as a template for the final geometry.

You can take a bit more freedom from the grid when constructing a made-to-order section of geometry, but you should get back on the grid at the edges of the piece to ensure everything will fit together seamlessly. Also, when building a custom mesh, as with the accessories, ask yourself if there's an easy way to divide it that would allow it to be used more flexibly than just drag-and-drop. For example, if you're creating a detailed starship control room that connects with modular hallways, look for a way to divide the room so an extra section could be added to expand the area. Or perhaps the customized railing that spans the bridge could be broken into sections and used elsewhere.

Level Assembly

Now comes the payoff, designing the level by snapping together the Lego-like units. The particulars of this process will vary greatly from one project to the next, but there are still a few general tips that can help (see figures 4a-4c).

First, rough in a few ideas before starting construction. Take a blank slate and see what you can do with the pieces you have before going back and requesting more. You might not need that custom piece at all if you can find a clever workaround.

There's nothing wrong with using a piece in a way it wasn't intended. If you create the level simply by snapping together TETRIS unit shaped corridors, the player is likely to reject the modular nature of the environment. It's critical to the workflow's success that users have the ability to improvise and invent new uses for prefabs. Some environment styles are more flexible than others, but there are generally areas of flexibility for any genre. A complicated sci-fi doorway may end up mirrored and used as a window. An altar may be duplicated and used as an archaic column. The possibilities for variation are greatly increased if your particular technology allows for non-uniform scaling of components.

Mix it up even more by using your accessory elements in key construction roles. For instance, you might take a single wooden beam mesh and build a shoddy doorway with it, or take a piece of ductwork and duplicate it in an array to form a unique wall. Each set of geometry presents an array of possibilities.

Finally, if your engine allows for projected textures or lighting, be sure to use them in key spots to further break up repetition of detail throughout the level.

Consider the Benefits

Even if modular construction techniques don't seem like something your project could utilize, there are a few key benefits to consider before making that decision.

First and most importantly, afterthoughts are rampant and a fact of life in game development. Working with prefabs offers an unprecedented degree of freedom to go back and edit an original piece, while having every instance of that piece automatically update throughout your entire world. Don't like the texture on that monitor? Implement it for now, and change it when you have the time. In many engines, an entire piece of geometry could be replaced as long as the new one fits the function of the old piece reasonably well. Working with movable

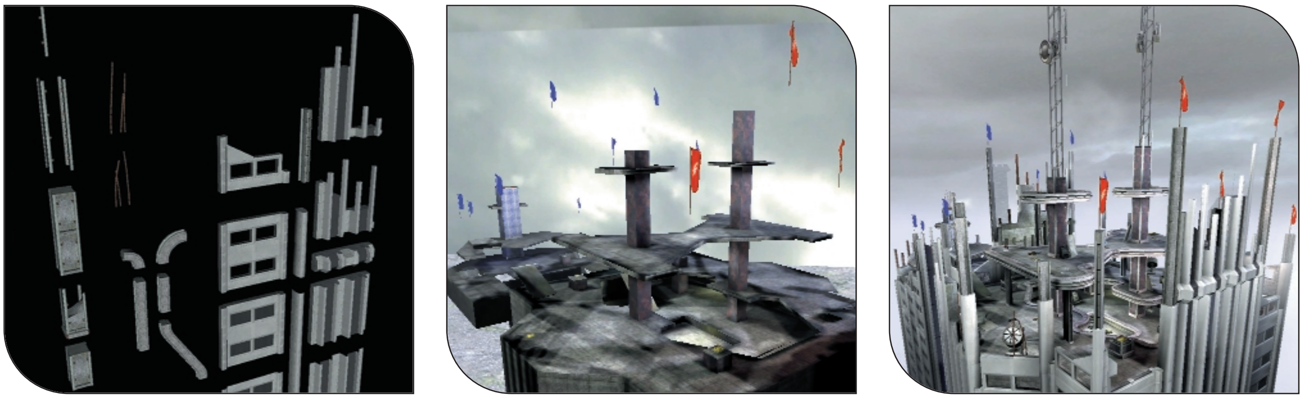


FIGURE 4A-4C (LEFT TO RIGHT). A relatively small selection of instanced prefabs (left) can turn a basic level (center) into a far more complex world (right).

chunks of geometry also allows for easy revisions on the designer's end, without having to go back and consult with artists. Are testers telling you they need an alternative exit from that room? That's no problem in a modular environment.

Regardless of rendering power, memory will always be a concern. Working modular generally allows an engine to work with many instances of the same object. A complicated wall piece can be duplicated many times with minimal memory impact after the initial placement. Even the densest-detail environment can get away with using only a handful of meshes in memory.

Another important consideration is consistency, and with bigger teams it's a bigger concern. By using a prefab set that has been built with minimal cooks in the kitchen, even a sprawling, robust level will maintain a style throughout. Such consistency is far more difficult to achieve in an entire world built of unique geometry.

We all want to reach as many players as possible with our games, and a modular system allows you to create very scalable levels of detail (LODs) for different platform specifications. Developers can flag high-detail extra touches so they won't draw on various video settings. If your technology allows or requires LODs, you'll find a modular construction technique facilitates that very well.

In addition, modularity allows team members to concentrate on doing what they are best at. Level designers with strong gameplay skills needn't worry about creating the loads of details required in high-end graphics. With a modular construction set they can focus on laying out the game and not get bogged down trying to create what should be considered art assets.

Finally, working with prefabs can help greatly if you plan on releasing a game-editing utility. Titles such as MORROWIND, NEVERWINTER NIGHTS, the WARCRAFT series, and countless past titles that have used modular or tile-based techniques have made their editability a key selling point. Editability helps extend a title's shelf life and creates a stronger user community.

Limitations and Drawbacks

Developers on some projects may see the word "modular" and panic. Initially, old-school, graph-paper-designed *Dungeons & Dragons* levels come to mind. Ninety-degree hallway corners and evenly spaced doorways creep into their nightmares.

The bottom line is that modular construction is indeed more restrictive than having a staff of dozens create an entire level as custom geometry and unique artwork. But for most teams, that kind of content creation workflow just isn't feasible. The key point is that having to fix some obvious tiling along the way is still far easier than creating a game full of custom content. Based on your game and its goals, you can decide the right level of modularity for it. If you're nervous, try starting on a smaller scale of prefabs that will allow for more detail in a smaller area.

Some designers find it incredibly stifling to work with geometry they didn't create. This can be a common scenario, but as graphics continue to improve at a startling rate, those designers will eventually have to accept that level-editing tools are not going to be able to create the kind of detail we'll be seeing in the next few years of AAA titles. It's really no different from using textures created by someone else. After getting accustomed to it, it's hard to ignore the benefits of being able to see one's gameplay ideas realized quickly.

Farther Down the Road

Looking farther into the future of game environment creation, who knows what we may encounter? Perhaps modular techniques may lead to truly immersive procedural worlds. Some say that's inevitable to some degree, and others aren't happy about it and want to resist moving in that direction. Wherever we all end up, and regardless of the specific nature of your current project, you're using modular construction in some fashion already. If your artists are creating tiling textures, if you have a standard step height, or if you have a tile-based terrain system, you're already in the mentality of building in a modular way; you merely need to extrude it into the rest of the environment.