GAMEPLAY

INTRODUCTION

Each game in each genre has its own gameplay. No two are 100% alike, but many share the same base. This portion of the book explains the base principal of gameplay that each game needs in order to outline where the 'fun' comes from. This section is divided into two primary sections: Multiplayer gameplay and Singleplayer gameplay. They will each cover many subjects including, but not limited to, floorplans, events, traps, item placement, AI, and more.

Please note that even if your interest lies only in one of the two gameplay aspects, it may still be useful to read both chapters since many of the topics can be applied to either.

BASE PRINCIPLES

At a fundamental level, there are two types of gameplay in a game: Map Gameplay and Core Gameplay. Understanding each is a necessity. Many people think of gameplay simply as 'gameplay' but this does not sufficiently describe the Core game mechanics in reference to the player tools, nor does it adequately address the environment, or map, that the player is dropped into in order to leverage the tools.

• Core Gameplay

The Core Gameplay is the set of gameplay rules and player toolsets provided by the game itself. The Core Gameplay determines how the player will spawn, how fast they can move, if the player can jump, and if so how high, and how the player is able to interact with the world. It also defines the player's goals in the world and the means to accomplish achieve them.

A level designer has little impact on the Core rules. They already exist and cannot be easily modified by the level designer. A level designer's job is to accept the Core Gameplay and then adjust the Map Gameplay to leverage the Core Gameplay.

• Map Gameplay

The Map Gameplay determines where the player will spawn in a world and then gives the player the environmental options needed in order to utilize the Core Gameplay. The Map gameplay provides the player routes to the other team's flag, it determines where the weapons are, what the floorplan is, it controls how the Core Gameplay is interpreted!

The Core Gameplay defines the player's toolset within the game, and the Map Gameplay gives the player a place where they can use the toolset to either their advantage or disadvantage.

Map Gameplay exists to augment and provide variety in the Core Gameplay. The term 'augment' is, perhaps, the most important aspect of Map gameplay. There are 10,000 + Unreal Tournament maps and even more Half-Life and Counterstrike levels out there. Use the Map Gameplay to give the player a reason to play level X instead of level Y. A map or level should offer more than the simple ability to run and shoot. All the other maps already offer that since those abilities are given by the game itself – the Core gameplay. The different ways a designer can enhance the Core Gameplay will determine if people will want to play the level or not and, ultimately, enjoy it.

The Core Gameplay is the foundation upon which the Map Gameplay is built. When beginning a game level, the house is not yet constructed. It is the designer's job to take the foundation and build it up until the house is complete. The Map Gameplay should add depth to the Core Gameplay.

The most common Map Gameplay mistake is to create a level that's simply too simple, and therefore adds nothing to the Core Gameplay. Almost every game out there that also provided a level editor has a 'box level' floating around on the internet. A 'box level' is usually a large cube where all the players or monsters spawn right next to each other and can do nothing other than attack each other. This kind of level adds nothing to the Core Gameplay. The only actions are the Core Gameplay actions themselves: spawn, run, shoot, die, rinse and repeat. The Map Gameplay itself is non-existent.

Some players do find such levels fun, but this is thanks to the game designer – not the level designer. The level designer essentially failed in doing level design – they failed to augment the Core Gameplay through the map's environment – the Map Gameplay.

Finding a game fun to play is quite different than finding a level fun to play. As a level designer, it is absolutely imperative to recognize and understand this difference.

'Good' Map Gameplay supports and motivates complex strategies and enhances the depth and fun already available through the use of dozens of elements. These can include traps, height differences (vertical or 'z-axis' gameplay), planned and balanced item placement, extra story elements, puzzles, and more. One example of leveraging Core Gameplay through Map Gameplay is the use of 'liftjumps' in the Unreal Tournament series. When a player gets on a lift, or elevator, by jumping immediately before the lift stops, they can propel themselves much higher in the air than normal. And depending on the speed of the lift, this can be quite high. Therefore, a level designer can help the player leverage the Core Gameplay by adding Map Gameplay possibilities around lift areas to reward the player for knowing how to 'liftjump'. For example, there could be an extra floor above the lift's normal exit that the player can access by 'liftjumping' and thus be rewarded by a Core Gameplay pickup. Thus, the level, the Map Gameplay, leverages the player's Core Gameplay 'liftjump' tool, by providing a reward for mastering the Core Gameplay. Without Map Gameplay, all the levels of a game would play the same. If a map does not augment the Core Gameplay, it fails in terms of gameplay.

Try to avoid focusing completely on either just gameplay or just graphics. Only a perfect combination and harmony between them can bring about a good game experience. Graphics can not only make the game attractive, but can also influence the gameplay, which itself supports the graphics. They interact and play off each other. A frightening monster setting needs specific environmental elements to function well, and vice-versa; specific environmental elements will convey the scariness of the monster. In a development world where gameplay is increasingly separated from the visual side of levels, I fear than this is often forgotten. Artists sometimes have little understanding of gameplay mechanics and gameplay designers often have little artistic insight. The end result of this separation can sometimes be that they both implement each of their visions into the environment and finish by working divergently rather than together. Don't make the same mistake – one cannot exist without the other.

FLOORPLANS

Although floorplans for SinglePlayer and MultiPlayer levels are quite different, both need to comply with some common basic rules. There are two general styles of floorplan: the Abstract, and the Realistic.

• The abstract floorplan

Abstract floorplans focus entirely on gameplay while the theme takes second seat. In an abstract floorplan, it's perfectly fine to place a hospital next to a factory as long as they look good standing side-by-side. It may not make a whole lot of sense, but it is fun. Many multiplayer floorplans are abstract; the levels in games like *Unreal Tournament* are perfect examples of this. Games like *Quake* also make extensive use of abstract floorplans, sometimes even in singleplayer levels. Other games like *Tomb Raider* and *Serious Sam* also use abstract floorplans.

• The realistic floorplan

Realistic floorplans meld the theme and gameplay together when constructing the floorplan. For example, in a factory, many different areas are needed: storage areas, transportation areas, fabrication floors, control stations, and more. Most of these rooms and areas would need to be laid out in a logical order following the process of whatever the factory makes. For example, control stations would accompany each fabrication area. Then storage areas would be placed ahead of the transportation areas because the items are usually stored before being shipped out. Perhaps the transportation area is both the beginning and the end of the cycle, in which case it needs to be at the head of the area chain, and the back, which would create a circular floorplan. The layout should make 'realistic' sense and serve to accomplish more than simply support gameplay.

Games that place a heavy emphasis on story are of laid out in this manner, for example *Morrowind* and *The Darkness*.

It is important to understand that both areas share pros and cons.

An abstract floorplan is usually much easier and faster to make. Therefore, if time is a limiting factor in the design, then picking an abstract floorplan might be the best choice. In addition, because they're abstract, they allow the designer to put more focus on the gameplay; especially in games with strong gameplay and weak stories.

On the other hand, realistic floorplans are more difficult to implement correctly, but will look better if done right. If the game also has a strong story and theme, then a realistic floorplan may almost be a necessity. Games with strong stories always have realistic floorplans. This is also true for games that rely heavily on certain well known themes/ environments. If the level takes place in a famous city, like London, the floorplan should take into account the general layout of the city. Conversely, a level that takes place on an alien planet would do fine with an abstract floorplan.

Realistic floorplans also tend to be easier to understand. Casual players are usually more interested in levels that seem to resemble something in the real world than levels that are hard to decipher if the theme is too strange or foreign. If the level's, or game's, target audience is casual, it may be a good idea to follow a realistic floorplan.

Of special note is *Portal*. The test chambers in Portal are very abstract in such a way that they do not resemble anything specific. They exist solely for the gameplay. This is perfectly excused, however, by the fact that they are in a test facility, which means that they don't have to adhere to anything other than the test facility. The game manages to portray its levels in a realistic way while at the same time, the levels are fairly abstract. This could also be an option, but it must remain carefully balanced.

Of course, in the end, the choice is up to the designer. Just be consistent and stick to the chosen type. If the two are to be mixed up, blend them together and balance them well so that they're not obvious. I recall a very vague area in *Resistance: Fall of Man*. The entire game had more or less realistic floorplans, yet in one area the player entered a small room through a doorway blasted through a wall. The small room contained a huge door and a plaque next to it with directions on it. Why would a room with only one, very obvious, entrance need to display directions on how to get out? Why is there a huge door for a little room? It just does not make sense. Such situations break the immersion and remind the player than they are just playing a game.

Regardless of the chosen floorplan, the player should never get lost. Being lost is frustrating and is not a valid type of gameplay. Make the player's path through an area very clear, or at least clearly demonstrate where they can and cannot go. Pathing hints can be almost anything: colors, tones, enemies, items, audio, and more. Many games have monochromatic color schemes and are therefore difficult to navigate through because the whole level looks identical. The previous example, *Resistance: Fall of Man*, suffered from this. Bioshock certainly did not as there was often plenty of contrast to help the player navigate. Throughout the book, this concept will often be revisited.

MULTIPLAYER

BASICS

Gameplay, especially in fast-paced multiplayer-oriented titles like Unreal Tournament and Quake, are often misunderstood. Most peoples' opinion comes from watching over the shoulder of someone playing the game. Therefore they usually come to the mistaken conclusion that the primary goal is to create as much blood and body parts as possible. This is certainly not the case.

Talented players who leverage the gameplay understand that the gameplay is more akin to a 3D chess games clothed in fast movement and over-the-top weapons. The level, ideally, should support and enhance this – which is exactly the goal in multiplayer levels. Far too many people create over-simplified floorplans and never touch any of the finer points of gameplay. Why would a player choose this level over that level? What does this level offer that that one does not? How does this level add more depth to the game? How does this level promote teamplay, and that level not?

FLOORPLANS

So how do floorplans fit into the grand scheme of things?

A floorplan should provide flow and the ability to use the area against the enemy. In other words, it should offer strategic potential to the players in the space. The level should offer the player opportunities to think. A few simple rooms connected by corridors is, in most cases, simply not enough because it adds nothing to the core gameplay. Ideally, it will offer extra depth and potential.

The floorplan should encourage the player to move around easily, otherwise known as the 'flow' of the level. Rooms with only one exit/entrance, long hallways, and dead-ends are all level designs that interrupt and stop the flow of a level. Design the area to promote speed, possibilities, and depth. All of these will be addressed in more detail later.

If the player must pause to ask themselves how to return to a previous area, then the flow is lacking. A player should be able to get anywhere in the level from any other location in it without too much thinking. The connectivity should be intuitive – so much so that it almost pulls the player through it all by itself. The player should be able to see the entire level by just constantly moving roughly forward.

Image GP 1

This is a bad example: Long corridors and several dead ends. These both remove the choice of where to go and what to do. There is only one way in and out: it is a trap.

A player who goes in will be trapped once an enemy appears. There's little to no reason for a player to enter such a dangerous area. This can be a way to promote a risk/reward situation for the player, but the reward would need to be fairly large for such a large risk.

The following example is an improved version, but it still isn't perfect.

There may be two entrances/exits, but this is still insufficient. How can a designer offer choices and strategy to the player when they are presented with only two options, which quickly becomes just one option if an enemy enters the room. Thus, the player's behavior is predictable – they will likely attempt to leave through the remaining doorway. There's no strategy in this situation.

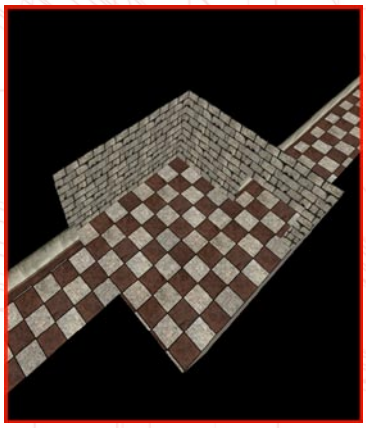

Image GP 2

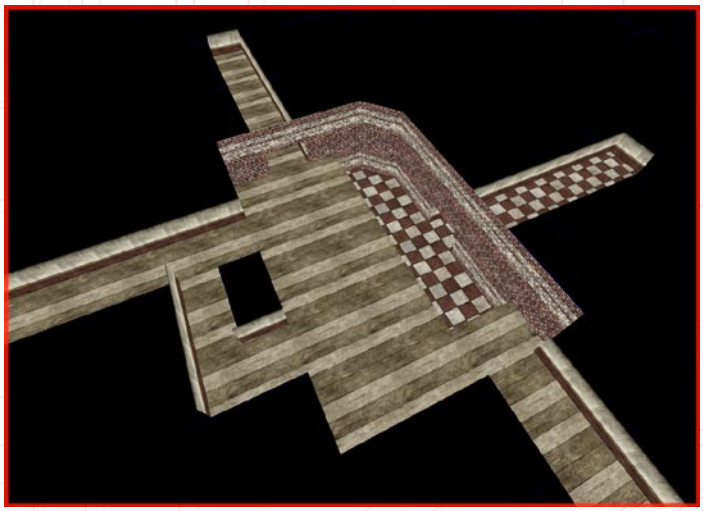

Image GP 3

The best situation is to have three or four exits, as can be seen in the following example.

Why is this better? It's better because there are multiple paths in and out. It flows; the players are given enough opportunities to allow them to make their own decisions instead of being forced into a single corridor. This opens the way for a more variation in the play, and more strategy.

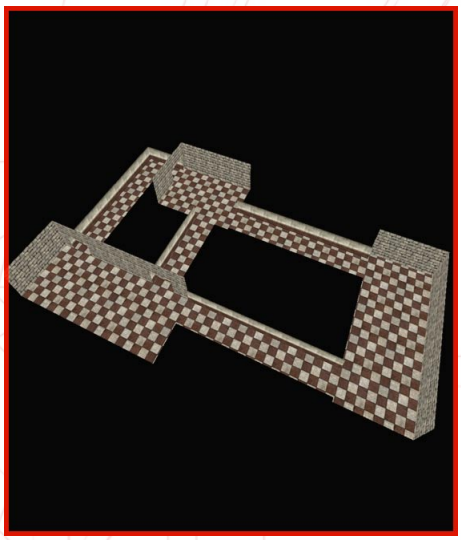

Image GP 4

Connections themselves, for example corridors, can lead to other problems.

In this example the corridors are entirely too long which is not ideal.

The result is a three-fold problem. First, the gameplay becomes overly linear and straight. Second, there's little gameplay in running down straight long spaces. And third, the player is trapped which results in the same problem as the room with two doors – it's too predictable. The player only has one way to move when an enemy shows up on one side and the long corridor makes aiming easy. This endless corridor situation is often described as a Room-Corridor-Room, or RCR, situation. Unless the core gameplay calls for it, this RCR scenario should be avoided.

Corridors should be short, or a medium length in general. Connectivity can also be added by intersecting the main corridor with a side corridor, or even by adding a hole in the floor leading to a different area. The following screenshot shows how a long corridor can be intersected by two other corridors and have a turn in the middle to give the player more freedom of movement, and block the view a little.

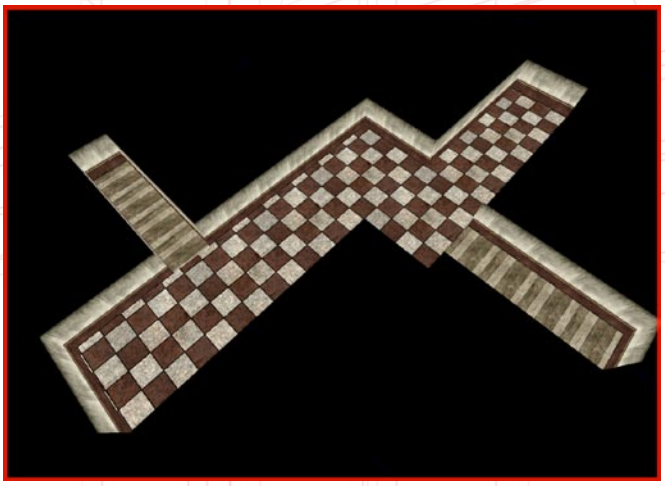

Image GP 5

A long corridor is not always a bad thing though. It can work well in singleplayer games or in specific multiplayer scenarios. In game modes like Deathmatch or Capture the Flag it could be exploited as a danger zone. This will be covered a bit more later on.

Image GP 6

Another common mistake is creating areas that are too large and/or open. Exceptions to this rule largely depend on the core gameplay. Examples of this are games that factor in mixes of infantry and vehicles, for instance the *Battlefield* series. In most shooters that do not involve vehicles, like *Half-Life*, *Quake*, or *Unreal Tournament*, these areas are to be avoided, like in following good example.

In this example the large open space is succesfully broken down into smaller areas by adding two large walls in the middle, and several levels and platforms.

The issue with too open areas is that they rely too heavily on Core Gameplay and aiming. Since the enemy can always see the player, the player's behavior becomes very easy to predict.

Open areas > the player can only jump around and attack > repetitive > predictable > boring

The gameplay goal in these types of games is more than simple aiming skills and some Core Gameplay. Areas should be broken up, closed, and opened in interesting ways to fully enhance the map gameplay beyond the Core Gameplay.

As previously mentioned in the Design Chapter, an important, but oft-neglected aspect of floorplan design is supporting the visual and technical requirements of the game. When I design a floorplan I keep the visuals and technical requirements in mind at all times.

Both of these elements are very important when designing floorplans. Certain rooms, such as large open cubes with catwalks are more difficult for an artist to decorate which increase demand on time and effort. Why not make it easier? A well-designed floorplan will not only help gameplay, but also the visuals and the framerate. One possible issue that can crop up with areas that are too open is that the area may end up being too difficult to optimize with occluders.

The point is that some shapes and sizes can make a level easier, or harder, for an artist to turn into something nice and well-optimized. If the designer is not responsible for the decoration, then talk to the artists who are. If the designer is responsible, then try to imagine how the visuals will flesh out the floorplan ahead of time. How will this room be populated? What kind of architecture and lighting will go well in it? What changes could be made to make it easier to decorate? Would it look better if the ceiling were higher? Perhaps all the interesting angles actually can't be filled with modular assets.

The level, as a whole, must remain in harmony with all its disparate components. Levels are populated by different assets and actors that are created to work together – not independently. Therefore, even when one's task is to only create floorplans, it's still necessary to have some sort of knowledge about all areas of level design. This is another reason why I, personally, am not fond of the trend of splitting up level design into many different design components as actual positions in a team.

ITEMS AND POWER-UPS

One very important aspect of multiplayer games is item and weapon placement. The pickups will determine where the player will travel through the level and the subsequent routes they choose. Pickups create goals and destinations and thus ambushes as well. Pickups can help the player by providing them with health, ammunition, and extra tools, but can also work against them by giving away their location and therefore endanger them. Usually, when a player picks up an item a sound is emitted. If an enemy is near enough to hear that sound, then they will figure out where the player is without having to see them. The level design should grant the player these risk/reward choices and facilitate the consequences of the player choosing to pick up the item, or to pass it by.

This can take the form of many small pickups with meager benefit, such as small armor bits, ammunition, small amounts of mana or health, or larger health portions at key points or connections in a level. They can be placed in doorways, corridors, on stairs or ramps, but with enough room for players to pass them by if they so choose. Give them the choice to gain some benefit but reveal their location, or to pass by silently but miss a slight boost. Powerful pickups, on the contrary, should be located in riskier locales, for example where the enemy can easily ambush the player, in order to offset the increased benefit the powerup gives.

The dead end in this screenshot contains a pickup that doubles the weapon damage. However, if the player decides to pick it up, they will cause a sound to be emitted, and nearby enemies will be alerted. They can then rush the player in the dead end. It's a dilemma. The benefit of being stronger is offset by the risk of imminent attack and no easy way to escape. This therefore balances the gameplay. The player should not get powerful items for free – make the player work for them. The more powerful or special an item is, the more the player should have to work to attain it.

Image GP 7

Another example of a simple floorplan trap is a corridor shaped like a donut. With the entrance and the exit right next to each other, any player entering the donut becomes trapped. Compared to a straight dead end corridor it does give a player a few extra options, but it still is dangerous.

Other examples would be to place the pickup in, or on, a trap like a floor that can collapse or in water. In most games, water slows down players and players moving through water make noise. Thus it's dangerous because the player constantly makes noise and cannot move as fast. In other situations, sometimes a door is placed in front of the pickup to make sound, or the pickup could be at the top of a tower where all the enemies can see the player taking the pickup; also the player picking up the tower item could be at risk of a long fall.

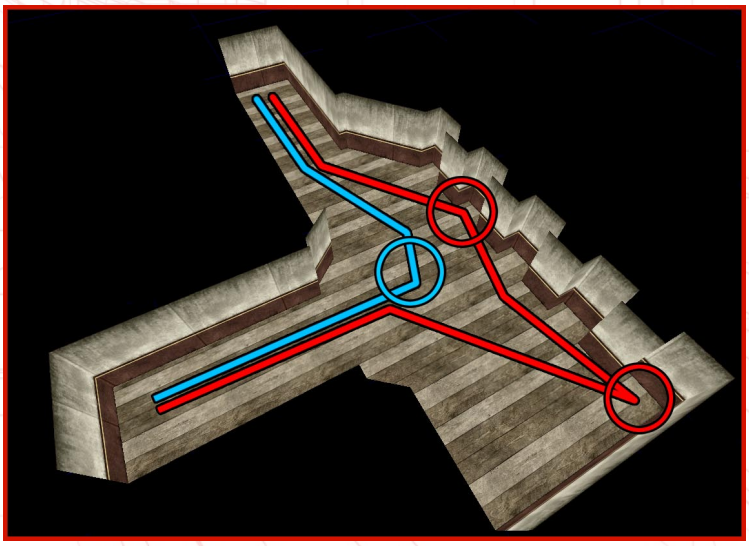

Image GP 8

Items determine how players travel through a level therefore they should be easy to get to.

The red line illustrates bad item placement and the blue line a preferred placement. Following the red lines, it should be clear that the extra moves used to grab a few simple items in the room shouldn't be needed. Allow the players to spend their time on more important gameplay instead of on collecting pickups.

Because of the above, I am not a big fan of games or mods that contain no pickups. Well known examples are Counterstrike and Unreal Tournament's Instagib. By taking away items, an entire layer of gameplay depth is wiped from the game. If it is impossible to add items to the level, then attempt to compensate them through environmental substitutes, which is never a bad idea, even when items are present. Environmental substitutes could be puddles

of water that give away player location based on splashes. Since pickups usually also emit sounds, puddles in their stead would work as well. Another common example would be an old plank emitting a creak when the player walks over it. More examples include shards of broken glass, doors, a movement-sensor attached to an alarm, and more. Not only do these substitutes enhance the gameplay, but also the theme and audio.

TRAPS

Traps can work in both Single and Multiplayer gameplay and can add fun, or frustration, to the game. They are especially liked by more casual players. However, just as in singleplayer games, keep it fair. Players should recognize the danger as soon as they see the trap – before they actually encounter it. An unfair example would be a 'secret' trap that kills the player with no warning. The player should be in control of their own fate in the game, or at least have that illusion. A good example would be the trap from Unreal Tournament's DM-Pressure level. A special chamber in the map holds a few powerful items, but the chamber can be closed and pressurized by players outside the chamber. The players in the map clearly see, and interpret, the danger and the designer offers them a distinct choice: pass it by and live, or enter the chamber to grab the items, but take the risk of getting killed?

Designers often add doors that open and close to levels. While they can work well to block a long sightline or to split up an area, they can also become inadvertent traps. The door should open fast enough that the game flow is not obstructed unless it's a special door. If the player is in a fast moving action game, then waiting for a door to open is not an option. Likewise, make sure the door closes only when there's no chance that a player can get caught in it. Being killed by a door in a game is very frustrating. Make sure the player spends their gameplay time on gameplay that matters and avoid causing 'accidental' deaths. However, doors can also be used for traps. What I occasionally do is add a door that is normally open, but can be closed at a push of a button. The door remains closed for just a few seconds, but this gives the player an opportunity to temporarily trap or delay the other player. It introduces a small bit of extra strategy to the map gameplay.

Another possiblity is adding a door that opens really slowly, but once open, it offers a short cut to another section for just a few seconds, before it closes again.

PHYSICS

In recent years physics have begun to open more gameplay possibilities than have ever been available before. One of the more popular uses has been to use pieces of the environment as weapons, for example *Half-Life 2*, which has made great use of that. Imagine the fun of defeating your opponent with somewhat ridiculous objects, like a refrigerator.

If the game supports an adequate level of physics interaction, then make sure that furniture or other movable objects don't interrupt the flow of the level, or give players opportunities to abuse the system to unfair benefit, most especially in multiplayer games. Imagine how a player with low health would feel if they were running away from an opponent and they suddenly find their escape route blocked by furniture one of their teammates bumped into and moved to the doorway. This could be very frustrating.

Or, instead, perhaps the game is objective-based, like Capture the Flag, and the opposing team has taken all the physics objects and piled them on top of their own flag so that it's now unreachable. Perhaps it would be funny the first or second time, but eventually it just becomes an unfair tactic created from unintended core gameplay.

Such situations should be avoided entirely and it is the designer's job to ensure core gameplay consistency across the map. A possible solution for the above dilemma could be to respawn the objects back in their original locations a set amount of time after they've been moved.

Especially true in multiplayer, ensure physics cannot affect the core gameplay – only level gameplay. An example of a 'bad' physics puzzle in a fast-paced shooter would be that the player needs to collect a few wooden planks before being able to reach the flag. The core gameplay is all about movement and action, yet the task of rearranging wooden planks takes time and slows down the player – thus invalidating the core gameplay – and the player is asked to do this very close to the objective which is when they'll likely be under the most amount of fire. A fair alternative would be to use the wooden planks far away from the flag, perhaps as a means to create a short-cut to the flag area. Although the situation still grinds a bit against the core gameplay, the situation is much more fair to the player as they're much less likely to be under attack when they attempt it. It is an extra step that is not necessary but may pay off and help the player if they decide to put in the time and effort. Make them optional extras instead of necessary tasks.

COVER

If the game uses a cover system, and this goes for both single and multiplayer, make sure that there is no perfect cover spot. Manipulate the cover and the geometry such that every single piece of cover available has a weak angle from which it can be easily countered - either from the left or the right, front or back, or perhaps even from above or below.

Direct gunfire does not have to be the only way the cover can be invalidated. Some kind of trap could be placed near the cover that can be activated from afar – by a button behind enemy lines, for example. Another example could a piece of cover that is obviously well protected – more so than the surrounding cover – but then add a button that lowers it into the floor, or opens a trap door underneath it. The button should be quite visible and within range of the cover spot so that players that are in cover would have the opportunity to fire at enemies moving toward the button, or move away of the cover spot. Remember to let the player decide their own fate.

Countering camper behavior is a beneficial side-effect of avoiding 'perfect' cover spots. Try to ensure that the cover is designed in such a way that campers cannot find areas with strong protection that also shares a long overview of the level, and has health or ammo. Add a second entrance to the cover area that's easily accessible – a long walk up four flights of stairs does not qualify. It could include super-fast methods such as using elevators or jumppads – anything to allow other players quick and especially fair access to a counter-attack.

AURAL FIXATION

Sound is an oft misused but extremely important aspect of gameplay. It can give away the player's position, or help identify where an enemy is, and also aids in the player's immersion in the world. The level should support and reflect all of these.

For instance, if more than one elevator is present, then giving each a different sound, even by using different pitches, helps players localize the source of the sound. If every elevator sounds the same, then the player would not know which elevator the enemy is using; the sound could be from any one of the elevators.

This is the same reason why to place environmental sound emitters, for example puddles of water or rusty metal pieces. Whenever players travel over them, the sound emitted gives away their position. This is also the reason why designers often place small pools of water in corridors or other areas – walking through them makes a sound and thus acts almost like an alarm for the other players.

Also, each item and pickup in the level should have its own unique sound. Thus the player can clearly hear what the opponent has picked up and where they are. High-pitched sounds are easier to hear so they should be reserved for significant powerups.

Other level pieces that can be used as 'alarm' systems for other players include doors, jumppads, teleporters, metal detectors, and more. Ideally, each should use a different sound to clearly give away their location.

A level in which sounds were used with great success is CTF-Coret from Unreal Tournament. Each flag room had two exits – one with a jumppad and one with a door. Whichever exit was taken, the defenders knew which route the enemy took because of the associated sounds. A smart enemy could counter this by first activating the door and then running back to use the jumppad. This is exactly the kind of extra gameplay depth that sound can add to a level. We will return to this example shortly.

IN-DEPTH EXAMPLES

CTF-CORET

I personally love CTF-Coret, the Capture the Flag level from the original Unreal Tournament demo. Its floorplan is relatively simple in that it only has two primary routes to reach the other base and, despite the lack of large open areas, it works well.

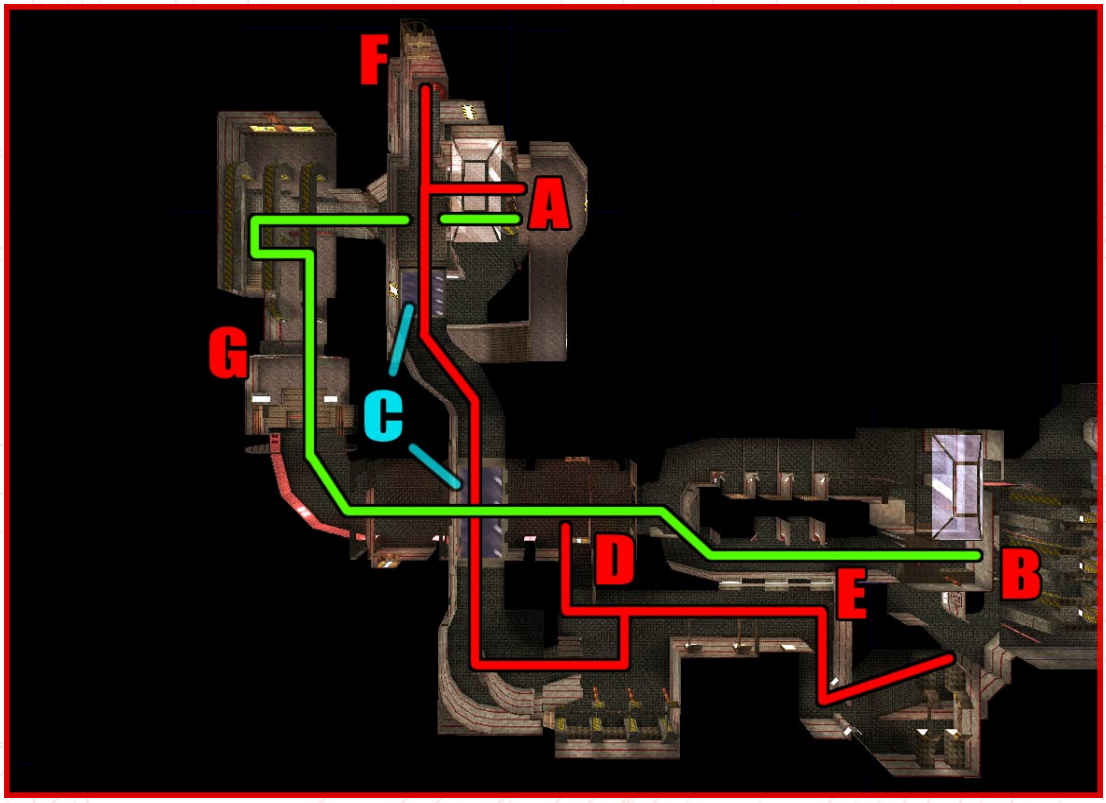

Image GP 9

The flag (A) can be reached via two routes: the upper, red route, and the lower, green route. The upper route can be reached by using a jumppad in the flag room (F), while the lower route will guide the player through a large door (G). There are two glass floors between the upper and the lower route (C) which allows players to see into the other route. Half-way down the middle, the routes cross and intersect several times which allows players to switch routes (D). The wall in-between the two routes contains a hole through which players can see and shoot at players using the other route (E). A third route also branches off the lower route half-way across the level, and continues to the midfield (B) following a slightly different path.

Now, how does all of this impact the gameplay?

Both the hole in the wall (E) and the two glass floors (C) temporarily reveal the position of the enemy to the player, and vice-versa, thus forcing each party to try to trick the other, or else the enemy will know where they will end up. The player could, for example, simply pause for a bit to mess up the other player's timing, or they could return and take another route instead. When the enemy has stolen the flag and is trying to return to their base it can also reveal the position of the enemy flag carrier.

In the flag room, the door on the lower route (D) and the jumppad to the higher route (F) follow the same logic. When a player uses the jumppad, or travels through the door, each makes a distinct sound, thus alerting other players. Choosing either route is not without risk as it gives away the player's position.

The floorplan only has two main routes which is usually ideal for a CTF map. Too many routes can make the level too difficult to defend since the enemy could come from almost any direction. Also an enemy flag carrier could switch paths too easily to mount a defense against. The focus in team-based game types should be the struggle between the two teams – not on which team can play hide and seek the best. Don't give too many options, and keep them focused. Ensure that there are enough central areas to reconnect routes. Usually there is one in each base area and another in the midfield although there can be a few smaller ones along each route, as in CTF-Coret (D). This idea can also be seen in the floorplan from CTF-Lopo which we'll examine next.

The players need the opportunity to change routes along each path so that they have options while trying to avoid the team chasing them. Without this ability, the design would have the same problems of the corridor example discussed earlier where the player's options are too limited to provide meaningful gameplay.

Most team-based levels work well with focused areas in middle of the map and in and/or around the flag base or end objective area. This usually is implemented by constructing fairly open base areas with multiple routes in and out. Initially those routes out of the base are fairly limiting and pretty safe, but soon open up more either in terms of space, or in terms of interconnecting paths. Once in the midfield all the routes connect again. Basically the floorplan boils down to balancing open areas against chokepoints throughout the level. Players can lose their pursuers temporarily while leaving the base but they may be faced with them again in the midfield or in other chokepoints, which balances the gameplay again, ensuring that there is no perfect route which is one hundred percent safe...

CTF-LOPO

CTF-Lopo is a level I made for UT3. The focus of the mappack in which it was included was on great gameplay in a low-polygon environment. It has a more complex layout than Coret, but follows the same core ideas. One difference is that Lopo adds more variation to the gameplay both in terms of height and in the type of combat.

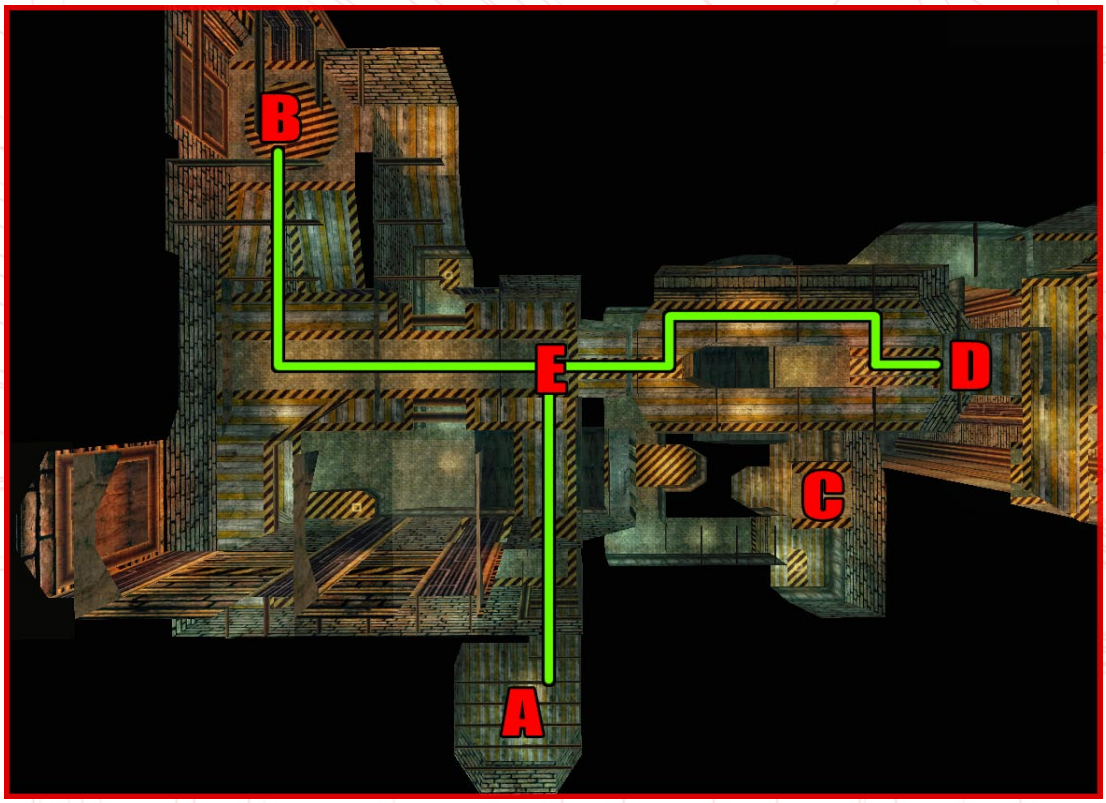

Image GP 10

The flag is located within a side room off the team's base area (B). Most players start off in a spawn room (A). The level consists of two primary routes; an upper route (green) and a lower route (see the next screenshot). The midfield area (D) is a large open area (not pictured). Players have the opportunity to switch routes half-way through the level (C).

Similar to CTF-Coret the midpoint of the map allows the player to travel from the upper route to the lower, or vice versa. It's somewhat small and simple (C) – a narrow corridor with a jumppad to propel players to the higher level. Yet here, the sound the jumppad makes when used can indicate to the other players that the player is changing routes from lower to upper. There is also an armor pickup in the area and is located slightly separated from the main routes so that players will need to invest some additional time and effort in order to pick it up. If it had been placed in the middle of the main route there would be too little risk and all reward for the players.

Although there are only two primary routes into the bases, there are many more routes available once inside. This also occurs in the middle. The area in front of the base acts as a chokepoint but connects to more open or varied routes. This encourages the defending team to plan a defense: do they want to defend close to the flag but potentially be caught by surprise due to the large number of routes immediately around the flag? Or do they want to defend the chokepoint where it may be easier to repel enemies, but leaves the flag itself somewhat unguarded?

The spawn area (A) is in a 'safe' corner where players can spawn and easily get back into the game through a number of routes through the base (E). This allows the defending team to either rush to the flag area or the midfield fairly quickly. If the spawn room was all the way in the back of the base, for example in the bottom left, it would be very difficult for newly spawned players to catch up with the flag carrier as there would be too much space to cover.

The upper route itself has some variation within in: it splits into two routes – one wider than the other – for a few meters. This adds some complexity to the gameplay as both require slightly different tactics. The midfield carries this idea one step further with a very open area. Most of the level focuses on close combat and indoor fighting, however the midfield area switches this up and presents the players with a wide open outdoor conflict area thus adding additional variation to the map gameplay.

In the base area there is a ramp that allows players to travel from the ground floor (A) to the next level up (B) and is bordered by a wall with holes in it. If a player sees an enemy taking the ramp up, rather than chase them up, the player can stay on the ground floor and fire at the enemy through the holes in the wall. As soon as the enemy ascends the ramp, their position is revealed which puts them at a disadvantage. The holes in the wall (C) help expose them and they're also fun to try to shoot through which adds to the satisfaction of having killed an enemy through them.

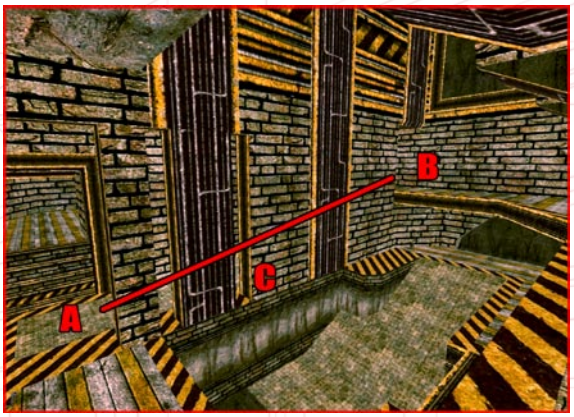

Image GP 11

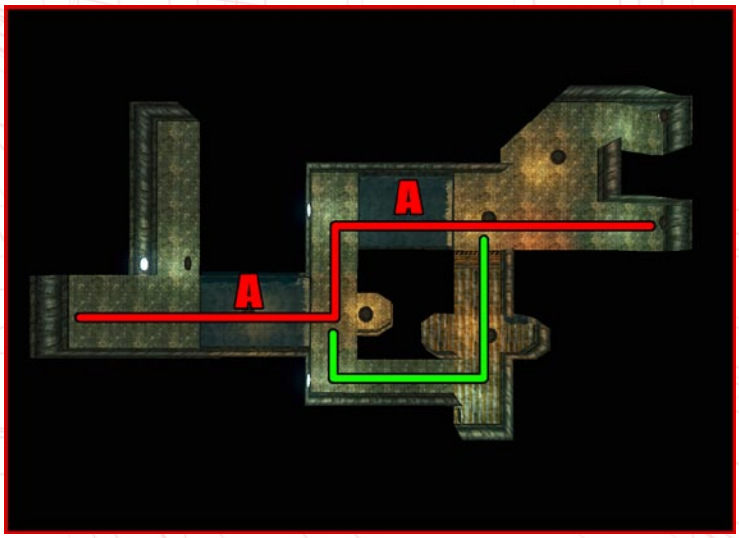

Image GP 12

Now, let's examine the lower route. The base area is to the left and the right leads to the midfield. The routes on the right side split – just as in CTF-Coret. Once past the chokepoints, in the midfield, the routes combine again. The route from the base to the midfield is split into the primary, red, route, and the slower secondary green route. The green route is narrower and thus more dangerous as there is less room to avoid gunfire and therefore usually goes unused. The main route has two large shallow pools of water (A) which give away the position of players traveling through them. And so the players are faced with a dilemma: give away their position but stay safer and use the shorter route of the two, or travel without noise indicators, but spend extra time in a more dangerous place. Regardless of which path they choose they will have to traverse one pool unless they translocate up (a teleport device in the game) near the left pool. However, translocating is likely to reveal their position anyways since there are sounds and particles associated with teleporting. Therefore there is another dilemma.

FINAL THOUGHTS

When designing a level with the player in mind, always think about what the player sees their first time playing the level. If someone playing the level for the first time cannot find important elements without purposefully searching for them then the design is flawed (unless it's a secret). All the fundamental items required for play should be clear on a player's first time in the gameplay area. Traps should be obvious and objectives must be easily discovered. If a level does not offer a certain level of accessibility, or 'ease of use', then people may just quit and play something else.

Certain elements can be added with more experienced players in mind and this is actually encouraged as it increases the depth and life of a level, and possibly the game as well. But a new player should be able to find the weapons, objectives, other basic items, and not get killed by sudden invisible traps. Frustrating the player does not promote re-playability. Keep it simple for beginners, but provide enough more complex aspects to keep more experienced players coming back for more.

Core gameplay and map gameplay are two separate aspects but map gameplay does have its foundation on the core gameplay. For multiplayer, this means offering enough variation on the core gameplay, without altering that core gameplay. Allow the player to add their own tactics to the game. Design levels with enough tactical potential that players can experiment with strategic choices. Satisfy the beginners who will enjoy pushing a button that causes a giant crushing object to kill an opponent, and also the advanced players who enjoy complex floorplans that allow them to mislead and predict their enemies' locations. Beginners are easily pleased by simpler, more obvious level features, while the experienced players will squeeze every ounce of gameplay out of a level. Strive to give both what they want without compromising the other.

SINGLEPLAYER

THE BASICS

The core of singleplayer gameplay is the story, the player's abilities in the world, and the interaction between the two in the level. The over-arching story spans, usually, the entire game with secondary events and/or themes that persist through a few levels. Because the story is a trait of the game as a whole, it is therefore part of the core gameplay experience. The level should not exist outside the storyline in the same way a player should not be able to modify the jump height or walk speed as defined by the game's programming. It can elevate the story and the core gameplay to a higher level, in short, enhance them.

A significant portion of map gameplay in singleplayer is communicating the story to the player in interesting ways. Instead of simply displaying the story through text, the player should experience it first-hand. Show the story to the players and get them involved in it. Try to avoid the player feeling like they are simply visitors to the world, or they are looking through a window (or monitor) at it. The created world must be interesting and compelling to play and experience. Keep it interesting by combining visual and gameplay elements in the level.

On the visual side, levels should contain enough thematic differences to prevent them from becoming repetitive. I personally found games like *Doom 3* and *Bioshock* to contain too much repetition because most of the environments throughout each game gave me the impression of being too similar. By being so repetitive, the levels weakened the atmosphere, story, and immersion of the player.

In addition to variation, the levels should hold enough style, atmosphere, and significant events to support the story. Suck the players in and make them feel like the world around them is real. Atmosphere is needed to make the story believable. Atmosphere is related to emotion and, some exceptions aside (*Serious Sam* & similar), most game stories invoke emotion. The levels should reflect the emotions through atmosphere and events.

On the gameplay side, players can be dragged in through AI behavior, events, unexpected story twists, the environment, and, again, atmosphere. Many designers make the mistake of only examining the gameplay side of the story. Try to avoid such a narrow focus. Singleplayer gameplay draws heavily upon audio and visual cues and vice-versa. It all must balance evenly in order to work well together – it cannot work together when separated. Setting up a scary monster encounter will be less impactful if the environment does not match the scary mood of the dangerous situation. Likewise, the frightening environment will be less so if the enemies do not scare the player through their placement and behavior. The connection between the audiovisuals and the gameplay is much stronger, usually, in singleplayer levels rather than in multiplayer.

This chapter will not cover AI behavior specifically because I regard it as part of the core gameplay. The focus is on map gameplay and not on the ways enemies attack, the behavior of weapons, their reload rates, etc.

Let's move onto a number of important elements of singleplayer levels, and speak about them more in depth.

AI PLACEMENT AND BEHAVIOR

Many people make some common mistakes when dealing with AI placement. Often, AI are placed without giving them a task, or without 'hiding' them. In many games and mods, it's easy to find an enemy standing still, simply waiting for the player to wander by. This usually feels quite unnatural. Enemies are supposed to be part of a living, breathing, game world – it's the place they live and work. This should mean they have their own 'job' in the world. Put them to work! If it looks like enemies are just waiting for the player, it ruins not only the credibility and depth of the level, but also destroys a player's suspension of disbelief. The player should feel like they're in a living world that exists on its own. Don't make them feel like they're the center of it and it's made just for them, otherwise it becomes clear that the world is just an illusion. Have the enemies and/or NPC's perform tasks in the world: operate machines, clean weapons, make fires, work on computers, patrol, have conversations, prepare food, read newspapers or books, hunt for food, protect their young – just about anything other than standing still.

An enemy that is guarding something, for example, does not simply stand still as in the screenshot on the right (yellow circle).

For a player, it would be awkward to enter the area (red arrow) and be faced with an enemy standing still, looking straight ahead until they walk by . Perhaps the enemy has recently had some head trauma, or that it considers staring straight ahead a jolly good time, but to the player it will simply not look natural. Again, put the AI to work – give them a purpose in the world.

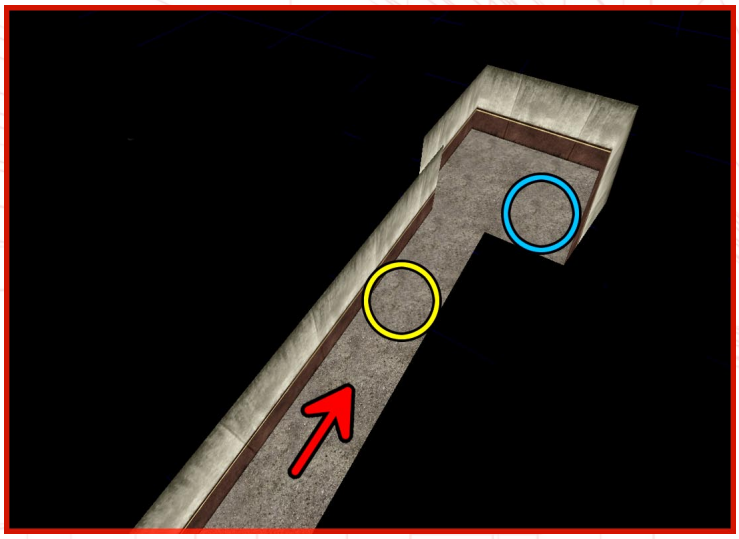

Image GP 13

If setting up their individual jobs is beyond the scope of the tools, or too time consuming, then at least hide them from the player until they're 'activated'. It's alright to have enemies wait for the player as long as the player doesn't see them waiting. In this example, the enemy could be waiting around the corner, in the next room (blue circle). When the player hits a trigger in the corridor, the enemy 'hears' the player, and reacts by running towards the player. This is much better than simply having the enemy obviously waiting in full view of the player. The NPCs should add depth and credibility to the level through their behavior. Standing still in full view easily breaks the illusion of a living world.

Also, be sure to incorporate variation into the NPC placement. Spawning the enemy behind a door every single time eventually becomes predictable and boring. I found the first few levels of *Doom 3* to be rather scary because of the sudden way monsters spawned near the player. However, after about an hour of play, I managed to figure our where the next monsters were going to come from after entering a new area, before there was any sign of them. The game maintained the same approach of spawning enemies throughout the game which softened, or deadened the experience and thus ruined the scary atmosphere. At the same time, it might be wise to repeat some of the NPC behaviors. By introducing some predictability into enemy behavior, it can help the player feel more powerful, or feel like they've 'outsmarted' the NPC's. This can also help the designer in terms of pacing a player's experience through a level. It may be that the designer allows the player to get used to a

particular NPC behavior by using it multiple times, or in certain situations. But then the designer can switch it up and slow the player down (or speed them up) by introducing or re-introducing another type. Just about when the player feels they understand the system, switch it up and replace it with something else. Variation, like most aspects of level design, needs balance. Too much will ruin the experience, too little could cause the player too much frustration.

The type of NPC is also important. Here, again, bad choices can ruin the level's credibility. NPC's should fit the environment they are placed in, according to the overall theme of the game. Generally, medieval knights should not be placed in a futuristic level. While this may seem obvious, this mistake does get made. Creatures or people living in a certain environment should reflect the environment. The environment influences them, and they influence the environment. Make the connection between the two clear. Whatever lives in the game environment should fit the theme in terms of behavior, art style, color, etc. Several types of mismatches can exist: style mismatch, theme mismatch, and color mismatch. An example of theme mismatch would be the knight example mentioned above. Style and color mismatches are often even more important than theme mismatch. Depending on the game's story, or the particular circumstance, maybe a knight does exist in a futuristic world. If so, the knight should still match in terms of style and color even if there is no theme mismatch. It is more important that objects in the world look and seem possible than if they are really logical since people often make judgments at first sight

The opposite can also be true. Sometimes thematic NPCs don't blend in. An example may be a swamp beast that looks out of place in a swamp because its colors do not match the swamp's green and wet look. In this case the type of NPC would fit the environment, but it would still feel out of place and wrong because of its color.

An example of bad styling could be a photorealistic soldier in a very cartoony environment, or a very clean character in a dirty world.

In addition to the above, it is also important that the characters fit into the surrounding environment and that the number of different characters is restrained to a certain 'palette' in order to remain consistent. Forty different characters in the same environment would probably introduce some inconsistencies into the game. Colors, styles, and themes – they must reflect the level and remain consistent.

The inverse holds true as well: if all the characters in an environment are of the same type, the player would quickly become bored. Here, too, the balance is the important aspect. Variation is needed, but it must remain balanced. Stick to a moderately sized, yet varied set of characters whose various skills and attack methods complement each other. One enemy that prefers close combat, and one enemy that prefers gunfights for example.

Invincible and/or invisible enemies are almost never a good idea. Just as with traps, enemies should be fair and reasonable. Anything that looks impossible, or is impossible, will frustrate the player at the risk of shutting off the game. Enemies that damage the player yet can't be seen are not fair to the player. The same is true for the typical boss encounter where the boss will not die, or sometimes won't take damage, until some hidden button or trigger elsewhere in the level is triggered.

To avoid seemingly impossible situations give the player clear signals such as helping them identify who, what, and/or where. A sniper can sometimes be an invisible enemy, or a trap, depending on how it is placed in a level. The player should be warned that there is a sniper before being killed or damaged by it. This could be done by having a sniper shoot the ground in front of the player, or by having friendly NPCs near the player get shot. Just about anything would be better than letting the player die without warning. In *Half Life 2* the snipers had a blue laser that showed the player where the sniper was located, and what they were aiming for. If the player ends up having to use a save game slot to find out there is

danger, then that's bad gameplay. Gameplay should not be based on saved games – they are not part of core gameplay systems. Punishing the player by death should almost always be avoided – if the player dies, they should relate their specific actions and mistakes to the cause of death – not simply the game killing them on a designer's whim. Danger should add to the experience and tension in a level – not detract from it through frustration.

The player should also be able to clearly identify where the sniper is – where the danger is coming from. Signals such as muzzle flashes, or light glinting off the scope, will balance the situation and give the player some measure of control over the experience. Place the gameplay emphasis on the challenge of the fight – not on a scavenger hunt for a too-well hidden sniper. If the player enjoys finding needles in haystacks then they probably would have purchased a different game. Most action/FPS gamers do not usually enjoy that kind of gameplay.

This also goes for invincible enemies. A very strong boss enemy who is invincible at first glance should communicate hints to the player as to how to approach the situation. And when the player does damage the boss, there should be some feedback to the player that they've done the right action. This is why many boss fights involve pain animations and/or pieces of armor or body parts falling off the boss when they get hurt. For example, the titans from Unreal played pain animations when they reached low health. Another example of a boss status change would be when they change how they attack the player, or they run to another part of the level. Basically, some change that indicates the player is making progress.

If the enemy can only be killed by exploiting a weak spot, or through the environment, this should be communicated to the player as well. Clearly illustrate what must be done and what to look for in order to do it. One example would be to switch to an in game cutscene that introduces the boss, but also shows something like weak pillars. The player may get the hint that if the enemy goes under the pillars, perhaps the player can damage them and cause some sort of collapse on top of the enemy. Another approach could be having an NPC nearby that tells the player about a particular weak spot on the enemy. Regardless of how, the key is communicating information to the player.

SCRIPTED EVENTS AND GAMEPLAY VARIATION

One major element of singleplayer games are scripted events. What are they exactly? Scripted events are singular situations in a game that usually happen unexpectedly and help reduce gameplay stagnation, which is something some designers seem to forget. I have played countless games where there was seldom any variation in the core run/shoot/jump gameplay. Without scripted events, these games ended up feeling repetitive and, eventually, predictable. Always provide variation to entice the player to keep playing. After being attacked in the same way, by the same enemy, thirty times in a row players will get bored unless the designer makes sure something different happens. The sole motivation for continuing to play shouldn't only be the enemy difficulty or the fun of attacking them. It should also be 'what's around the corner?'. Don't let the player predict that.

Scripted events can be anything. They could be something simple like an enemy suddenly jumping through a window, or blasting through a door. It could be a surprise attack by a group of cooperating enemies, or a sudden change in the environment like a collapsing bridge or exploding building. It could also be as complex as an event that drastically changes the course of the story or gameplay like killing off an important character or revealing a suite of new objectives in a suddenly flooded area. Here are some examples of scripted events from well-known games:

The idea behind scripted events is to help players enjoy the game, add extra life to levels, and to provide variation. They pull the player into a believable world and keep them entertained and impressed by what they experience.

- In *Unreal*, the player is trapped in a corridor. The ceiling lights progressively go out one by one and then a door opens and the player is confronted by an important enemy (Skaarj) for the first time.
- In *Unreal 2*, a Skaarj jumps down on top of an elevator the player is in and rips open the ceiling.
- The Black Mesa Lab explosion at the beginning of *Half-Life*.
- Several areas in *Half-Life* where soldiers are brought in by a dropship.
- In *Half-Life 2*, during the hovercraft section, a big factory chimney falls across the player's path.
- Also in *Half-Life 2*, Father Gregori in Ravenholm helps the player at various times by shooting enemies.
- Sometimes in *Call of Duty 3* enemies block a door when they see the player approaching.

Another aspect of scripted events to keep in mind is that they help the game world feel like a larger place. They help the player forget that the game world revolves around them. Events don't always have to involve the player. The player can simply be a spectator. Scripted events that are completely irrelevant to the story can add just that extra bit of depth. A plane crashing into a mountain in the distance may not have any significance in terms of the story, but it is impressive to see and helps the world feel more expansive.

Events must occur in the player's view or else they haven't happened. Telling a player a spaceship has crashed is no fun. It is fun if the player experiences it themselves. The player should feel like they are part of it all. In *Unreal* most of the game's big events had already occurred by the time the player got to the various areas. While this fit that particular story well, it also left a dearth of impressive events for the player to experience while they traveled through the world. In contrast, in *Half-Life* events usually happen while the player is there, and it makes a big difference. More immersion and spectacle becomes possible. A player shouldn't have to read about something that happens. It should occur in front of the player's eyes.

Yet another reason to implement scripted events is to introduce new traps, enemies, items, or gameplay changes. If there is an important new type of enemy or a very dangerous trap, a scripted event can be a powerful design tool that can communicate information to the player. Previewing the danger or uniqueness of an item or area can hand information to the player that they'll need once they enter that space. *Gears of War* did this very well in the PC edition when the large Brumak monster is shown to the player several times over several levels before the player finally gets to fight it. Scripted events can build up 'hype' for an upcoming showdown or other significant event. The Brumak would not have had the same emotional apprehension had the player simply rounded a corner and come face to face with it.

Weapons can be foreshadowed by forcing the player to face enemies already equipped with a new weapon so its power and potential is understood by the time the player gets it. Or, as in *Unreal's* Eightball cannon, hyping up the weapon through the story and making it come across as a mythical weapon hidden somewhere in a hard to get to place. Finally, it can be achieved through less complicated means as in *Half-Life 2* where the player was directly given the Gravity Gun through a series of cut scenes/conversations. In any case, incorporating scripted events into gameplay-significant introductions have a much more positive effect than just hiding a new weapon somewhere in a level.

This same idea can also be seen in other media in addition to games. For example, in the move *Aliens 2* the marines are not confronted by the aliens immediately. Instead they

are introduced to them step by step; first by the girl, then the motion sensor, then through Face Huggers, and then finally through a big exposition. Many games, movies, television shows, and other media show new enemies by the 'after-effects' of their presence before making a 'proper' introduction. Scripted events are the tool in the designer's pocket that can incorporate this type of foreshadowing into a game.

FLOORPLANS

Floorplans in singleplayer are simpler, to a certain extent, than in multiplayer. Contrary to multiplayer, long corridors and dead-end rooms in singleplayer can be acceptable and are sometimes even interesting since they can be used to enhance theme, style, and/or atmosphere. Unlike multiplayer, singleplayer relies more heavily on enemies, story, scripted events, and atmosphere to accentuate the floorplan. What scripted events happen and where? How well can the AI behavior adapt with different floorplan varieties? What is the atmosphere like? What is the story at this point and how does the floorplan reflect it?

Often, specific needs must be met to match specific events, for example a very large room or a dead-end corridor. As a result, the actual layout is already roughly conceptualized which makes the designer's job a little easier since they can do what they like within the gameplay and story constraints.

Even with this freedom, a few typical layout elements could improve the game experience and/or lessen the designer's workload, like reusing areas. Don't be afraid of reusing existing areas from the level. The work will get done faster, and it adds more depth to the level by reinforcing the theme throughout different locations in the level. Just don't overdo it and be consistent. There are various ways to reuse areas.

• The Preview Method

The preview method allows the player to see areas that currently are not accessible but will be later in the level or game. This allows a designer to reuse some parts of the level and also provide information to the player. This means less work for the designer and it also benefits the game experience. It creates the illusion of a world larger than the immediate play area and can also help the players orient themselves. The player can identify landmarks and understand how the games areas interconnect.

In the screenshot on the right, the player travels over a bridge crossing an underground river and can see a little beach in the distance.

About 15 minutes of gameplay later, the player actually ends up on the beach that was viewable from the bridge.

Image GP 14

Image GP 15

This can also be done in an even simpler way. *Half-Life 2* previewed a large skyscraper in the distance at the start of the game.

By the end of the game the player actually reached and entered the building.

The final location of the game was previewed from the very beginning of the game.

• The Evolution Method

Another way to reuse level elements is the evolution method. The player returns at least once, sometimes more, to the same area in a level. But each time they return, they are presented with an altered environment or change in the gameplay. One example is from the *They Hunger* mod from *Half-Life*. In one of the levels, the player infiltrates a large building. Later, the building catches fire while the player is inside the same area, but now with flames and scorched walls and objects surrounding them. A few levels later, the player is again returned to the same area but at this point the fire has gone out leaving an altered environment that changes the floorplan due to the blackened husks of objects and new holes in walls and floors. The environment is played several times, but each time there's a significant difference and thus does not become boring. Changing the visuals and gameplay each time helps immerse the player in the game world as the changes demonstrate that the world is alive and subject to change – just like in Real Life™.

• The Recycle Method

The last method simply reuses and recycles the area with little to no change. While this may sound like a cheap way of extending gameplay time, there are ways to use it well. One of the simplest methods is to send players to the left (or right) side first in order to perform an action, for example to hit a button that opens a door on the right (or left) side. This causes the player to cross the central area multiple times.

An example of this in a game can be found in *The Darkness*. It used two subway stations as hubs to get to nearly anywhere in the game world, effectively reusing the exact same level a dozen times without degrading the quality of the game. Instead it enhanced the game.

Half-Life expansion packs *Opposing Forces* and *Blueshift* also implemented this method on a much larger scale. In these packs, the player plays either a soldier or a security guard in the same area and during the same time frame as the story and events from the original *Half-Life*. Often the player would walk into the same rooms they saw when playing as Gordon Freeman in the original game, but are instead placed in completely different situations and they view the game story from different perspectives.

Reusing areas can greatly reduce the amount of work on an environment. Effective reuse can be a positive in terms of both the depth of the experience, as well as for the length of the level, and even in terms of the amount of work for the artist or designer. Just be sure to always have something new whenever this method is used. Reduce, and remove any predictability the player may experience by returning to a location they've been before.

One can also build multiple unique variations for a single area. While it may mean more work overall, it is nice to have multiple routes to a goal each of which have their own advantages and disadvantages. This is something that is always possible to implement, even when working on a very linear level or game. Two different exits from a room which eventually lead to the same area is the simplest example of this method. Another example would be a more open floorplan where most rooms have several exits and all connect to the same set of corridors. Games like *Hitman* make very frequent use of this type.

By now, it's probably fairly clear that I'm not exactly a fan of dividing gameplay and visuals between different people and departments. As described previously, I have the same opinion regarding singleplayer floorplans. When a floorplan is designed, it shouldn't only be fun, but also artistically and technically feasible and cohesive. I have had singleplayer floorplans come across my desk that were huge and thus impossible to create in the given time restrictions. I've seen floorplans with giant open areas that were thus impossible to optimize. I've seen floorplans that required skyscrapers in a jungle which rarely makes artistic sense. I've even seen some that threw out all the rules of composition and requested that all the important elements be placed in one small area. I could continue but the point should be clear: basic insight of each element is needed even if one is responsible for creating only one particular aspect of the whole. When making a floorplan, endeavor to make it realistic in terms of workload, technical limitations, and composition – not just in terms of gameplay.

LANDMARKS

The primary purpose of a landmark is to help players navigate through the world. They can be large and small, they can be buildings or rock formations, or they can be small and unique like graffiti on a wall, or even a unique vehicle parked somewhere. The key is that they stand out in some way. They help people remember locations in the overall map and grounds their spatial awareness. Landmarks are important to gameplay because they give the player direction. Getting lost in a level is always frustrating and landmarks can lessen this possibility.

Landmarks also can help players get a feel for the scale of the world and preview their destinations. As mentioned before, the giant skyscraper in *Half-Life 2* is a perfect example of both. Players saw their journey's end right from the beginning and the tower in the distance also gave them a sense of how large the city was. This, in turn, helps the environment appear more credible and realistic. *Oblivion* used its central city's tower in a similar manner; it gave the player a consistent landmark to look for so the player could orient themselves almost wherever they went in the level. Many games make use of smaller landmarks, for example *Bioshock* used many small unique elements to help give each room an individual feeling. They broke each of the areas into sub-areas for the player which was enough for basic navigational needs. We'll return to this more when we examine how composition can guide the player in a certain direction.

INTERACTIVITY

Interactivity makes the world more believable and it can even add fun. Interactivity means that something influences something else. This could be something reacting to the player's actions, or simply something as simple as vegetation reacting to wind or water. A world is not a static object – it lives. Videogames differ from movies and other traditional art forms through their interactivity and animation. Yet those things that live and breathe and move in life will not do so in games unless designers set them up to do so. Allow the player to talk and interact with NPCs, allow them to make changes to the game world and allow NPCs to react to such changes. Make the vegetation sway in the wind, have the characters leave footprints behind, show that the world is changeable or at least demonstrate some sort of activity. Practically anything that can be done to remove a static feel is a change for the better. 'Every action has an equal and opposite reaction' is still an incompletely attained goal in nearly all games and designers have a direct influence on this so endeavor to increase the amount of movement and life.

Scavenger Gameplay is one of the most widely used interactive elements in singleplayer gameplay. Scavenger gameplay is the name I use for the common 'key-quest' where players must enter area A in order to push a button or find a key that will open or unlock area B. This method can quickly become boring or a cheap gameplay ploy so use it sparingly. Avoid being overly simplistic when implementing this type of gameplay. Running into a room, grabbing a key, and running out simply does not challenge a player. Try to make use of secondary elements in conjunction with the scavenger gameplay, for example the player must save an NPC who will then reward the player by opening the door.

A more complex example could be one where the NPC who holds the key is actually part of a mission/quest. Perhaps the NPC falls off a cliff and the player must then retrieve the key. The player would need to find a safe path down, probably fight off some enemies along the way, maybe in a sewer system, and then the player would need to retrieve the key from the NPC's body. This would also allow the designer to reuse the path down as the player travels back up which would be an efficient use of the time invested in creating the area. Try to create unexpected situations for the player to pass through in a variety of ways. Whenever the player feels like they can easily predict what will come next, try to surprise them and make something different happen.

Keys and buttons can take the form of many different shapes and sometimes functions as well. In addition to regular keys, the player could search for a battery, a key-card, an access code, a power switch, a control panel, a person, a tool, explosives, or even manually trigger an environmental event to open the path, like boulders crushing a door for example.

The purpose of scavenger gameplay is simply to encourage the player to explore and achieve something else first. The more creatively this is done, the more the player will appreciate it. Just make sure there's enough variation in this throughout each level and the game. On larger scale, scavenger gameplay could translate into a situation where the player must gain a certain reputation with certain factions or teams by completing enough missions before they can access certain tools, information, or even a new gameplay space. This, in turn, can expose the next stage of the story. Games like *Grand Theft Auto* and most RPG's make use of this.

TRAPS

Traps are commonly used in singleplayer games and can be set up either by using environmental objects, enemies, or both. The most difficult part of creating traps is balancing them appropriately. They must be equal parts challenge, fair, risky, and rewarding. Often designers will place 'instant-death' traps with no warning given to the player, which is very frustrating. Remember: players want to feel as if they have some amount of control over what happens. Forcing them to die without allowing them to understand why, or even see the danger ahead of time and be able to plan around it removes control from the player which in turn causes frustration and irritation.

Communicate the danger to them – warn them. Place dead bodies around the trap area, or show an NPC or teammate getting killed by the trap, or place a suspicious detail in the environment nearby which could be as simple as a sound. Just don't allow the player to learn about it by using their saved game slots. *Oblivion* implemented trap communication quite nicely by clearly showing trigger ropes near the ground in front of the trap area. *Oblivion's* traps were fair because the player could locate the trap without running into it first by noticing the rope, yet the traps were still somewhat hidden and deadly enough to be challenging. A player should understand that when they died from a trap, it was their own fault and that they were in control enough that they could have avoided it if they had not made a mistake. Skill and leveraging the core gameplay makes games challenging, not luck.

ITEMS

Players should ideally always have just enough health/mana/ammo/etc. available. If not, the game becomes frustrating and besides, what's the point of having cool weapons if the player can't use them if they're always out of ammunition? On the other hand, provide too much of an item, and some of the tension and challenge of a game is lost. As with everything else, endeavor to strike a balance. A corollary to this is variation. Encourage the player to make use of the different items, for example weapons, by scattering them, and their ammo, throughout the level. This depends on the game, of course, but if the player carries five weapons with more than enough ammunition for all of them, then they will only use the best weapon and they'll ignore the other four. When this happens, there's almost no point in having created the other four. A designer could even set up specific areas that leverage the strengths of a particular weapon. For example, creating a section where a sniper-type weapon is particularly effective, or an area where a close-combat weapon would be more effective than others.

One common mistake many designers make is to spawn new monsters/enemies immediately after the player grabs some health or armor. Unless specifically called for by the game design, try to avoid doing this. Few players will attempt to grab that 25 point health pickup if by doing so, they expect to lose those 25 health points again when a monster spawns behind them and attacks. This is a pointless gameplay feature that can easily lead to frustration. Giving the player a reward and then punishing them for taking it is a cruel practice.

There are certain exceptions for special types of health and armor, however. If the health pickup is a larger quantity than usual, and the player will walk away with more health than they entered with, then giving them a challenge when they pick it up is generally allowed and, sometimes, encouraged. The difference between the two situations is that in one, the designer is giving the player a certain amount and then immediately taking it back. With the other, the designer rewards the player with something large, and the player can keep some of it, but it's up to them as to exactly how much extra they leave with.

Another use of items is that they can be used to indicate paths through the environment. An item can be placed in such a way that they players receives a hint as to a direction of travel. If the player must choose between a left and a right corridor, and the right corridor has a few health pickups, the player is more likely to be drawn to the right corridor than the left one. Another example would be a situation where the player is up on a ledge with another ledge below them. If an item the player can pickup is within view on the lower ledge, then the player gets the message that it's an ok place to jump down to. If there were no item there, the player could become stuck because they would not know that it's safe to take the jump down.

DIFFICULTY

A level designer's goal is not to frustrate players, but to entertain them by offering them a fun, challenging, memorable experience. Challenge and frustration are two different things and frustration is generally not considered fun by most people.

The modern gamer wants to experience action and adventure. They don't want to be forced to overcome overly difficult challenges that demand weeks of time to solve. As gamers grow older, an industry trend, and start building families, they have less time available and want to progress at a pace that allows them to experience a fair amount in a shorter period of time than what they used to need. If the average Joe comes home after a hard day of work and at the end of the evening has just an hour of free time, then the last thing they want is to get stuck just ten minutes into playing. In such situations they get frustrated instead of having fun and they turn off the game. Remember – games are about entertainment.

As a level designer, the levels created should support different difficulty settings the game offers. Games like *Devil May Cry 3* are ridiculously difficult even on the easiest skill level settings. Different difficult modes allow players to decide how intense and challenging they want the game to be. Don't force your own opinion on difficulty on them.

There are many ways levels can be made easier on low difficulty settings

- Weaker enemies - less health, do less damage to the player
- Fewer enemies
- Better weapons found more often
- More (good) ammo
- More ways to recover health in more places
- More cover

It is important to avoid making levels more difficult by removing save checkpoints. The difficulty of a level should be contained within the gameplay – not in the number of save checkpoints. Removing them will only lead to frustration where the challenge should rest in the skill of the player versus the enemies.

Also, checkpoints can be used to communicate that a significant event will probably happen soon. The player will be more likely to suspect that something big will happen near checkpoints. If the gameplay in a level is built on suspense, checkpoints can easily be figured into the design to enhance or allay tension. Or they can be used in a way that goes against player expectations just to keep the tension that much tighter.

PACING

Pacing is used to describe the ebb and flow of experiences that the player has as they traverse the level. Distribute interesting things throughout the entire level. Make sure to provide enough to keep the player interested, but don't overdo it because there will be nothing left to use at the end of the level which should be the climax – the part where the coolest/most challenging/most awe-inspiring thing happens. If possible, attempt to preview the interesting parts of a level right from the start. For example, when a player enters the level, a cutscene previews events that take place near the end of the level, or maybe some key areas of the level. Wrap it all in a nice package and present it to the player in pieces. Another example could be that, upon entering the level, the player sees a very impressive building in the distance. Instead of traveling right to it, they could pass through lesser areas that are still interesting, like maybe the foundations of the same kind of building, until the end of the level, or suite of levels, where they finally reach that grand building they saw all the way at the beginning. Introduce the cool aspects of a level straight away in order to pique their interest and get them hooked on playing, with the knowledge that more is to come.

Games often do this on massive scales. Plenty of games, like *God of War*, show the player what they will attain, and then they take those features away from the player. As the player progresses, they earn those features back. Previewing cool aspects can help encourage players to continue with the knowledge that they'll get something interesting in return.

Boss fight levels are especially notorious for this approach. The PC edition of *Gears of War* did this very well by revealing the huge Brumak several times through several levels. It kept the player playing because they knew that they would have to face it at some point, they just needed to play a little longer to get there… and a little longer… and a little longer. Motivate the player to continue their progress, just be sure to balance the length of the effort with the reward.

FINAL THOUGHTS

The best result will, in the end, be a combination of all of the above in the right balance. The more of these that are allowed to mix together and influence each other, the more coherent the world will end up being. The whole is greater than the sum of its parts. For example: a player runs through a corridor when suddenly an enemy opens up some water and steam pipes. Water floods the corridor, and the player ends up with reduced movement in waist-high water with their vision obscured by steam. After some time, perhaps the floor collapses under the weight, or maybe just a portion and the player is sucked into a hole. The player then ends up in a completely different area and will thus need to find their way out.

The different aspects described in the above situation are: enemy behavior variation (enemy performs a special, unexpected move), environmental depth and interactivity (world can be damaged and damage has a result), environmental visual changes (the flooded corridor looks different), and last but not least, there are significant changes to the gameplay (gameplay speed, steam obscuring vision). Also, once the floor collapses, the player gains a new task and enters a new area, which is always somewhat of a reward as it encourages more play.

These types of advanced events happen rarely in todays games. Break the mold of the standard 'enemy shoots at player' gameplay and add some much needed life to the level and the gameplay!

One of the most important points in this chapter is to inject variation. Do the unexpected! The player should never be able to completely predict what will happen or otherwise the game will become boring. Repeated gameplay tools, like spawning monsters in a certain way, will only have an impact the first few times the player sees it. After the 20th time, it becomes part of the background. Adding variation to all these systems will create a more cohesive, living environment, keep the player on their toes, and keep them playing longer.