
The dos and don'ts of CS:GO level design



Created by @El_Exodus
Latest update: June 2019

Index

1. Prologue	4
2. Layout	4
2.1 Meeting points/Battlefronts.....	4
2.2 Chokepoints	5
2.3 Staging Areas	6
2.4 Bombsite entrances	6
2.5 Post plant areas	6
2.6 Simplicity > Complexity.....	6
2.7 Unused Space / Areas without serving a purpose.....	7
2.8 Negative space.....	7
2.9 Support various playstyles.....	7
2.10 Allow advanced tactics and teamwork.....	8
2.11 Wingman specific chapter	8
3. Routing	8
3.1 Avoid obstructions.....	8
4. Flow (Chapter written by @oliver_irl).....	8
4.1 Natural guidance	8
4.2 Decision-making	9
4.3 Loops.....	10
5. Navigation/Intuitivity	10
5.1 Landmarks.....	10
5.2 Roof detailing/Alignment hints	10
5.3 Detailing.....	11
5.4 Consistency.....	11
6. Timings	11
6.1 General	11
6.2 Battlefront timing	12
6.3 Avoid wasted time	12
6.4 Rotation time	12
6.5 “Around the world”	12
6.6 Measuring timings	13
7. Risk and Reward	14
7.1 General	14
7.2 Risk and Reward via route design.....	14

7.3 Risk and Reward via sound design.....	14
8. Sightlines	14
8.1 Long sightlines	14
8.2 Tight angles.....	15
8.3 Pixel angles	15
8.4 Vertical sightlines at ramps	16
9. Verticality	17
10. Auditive Design.....	17
10.1 Spatial awareness	17
10.2 Environmental Audio	17
10.3 Sounds of interactable Objects / Triggered sounds / Positional hints	17
10.4 Allow sneaky plays.....	18
11. Cover.....	18
11.1 Avoid Head peeks	18
11.2 Natural Cover.....	18
11.3 Overpowered Cover.....	19
12. Models/Props	20
12.1 Model shape and model collisions	21
13. Scale/Dimensions	23
14. Grid	23
15. Visibility	24
15.1 General	24
15.2 Environmental Lighting.....	24
15.2.1 Colouring	25
16. Spawns.....	25
17. Buy zones.....	26
18. Clipping.....	26
19. Basic Optimization.....	27
20. Presenting your map	27
21. Playtesting	28
22. Dealing with feedback	29
23. Further guides and tutorials	30

1. Prologue

Playing multiplayer games on well-designed levels is usually a great experience while playing on flawed maps often leads to frustration. If you're designing levels, you obviously want people to enjoy the levels you create. However, if you're new to the scene, it's hard to start out without prior experience of what's good and bad. This guide aims to assist you in your design choices by providing 'good measures' in moments of uncertainty during map creation. This guide isn't meant to be a fixed ruleset, rather it's supposed to be a piece of reference material to lead you in the right direction.

Since I joined the mapping community back in 2014, I've witnessed a lot of unique and interesting maps – good ones, bad ones and most of them in between. Almost every level can become a good one, if enough time and the right changes are put into it. Iteration is the key for a good layout.

Hopefully this paper will assist you in making the correct decisions and adjustments to your current and future projects. It's designed to help you succeed in mapping and as a paper of facts and tips to revisit later.

While this guide is aimed at the classic defuse game mode in Counter-Strike: Global Offensive it can still be beneficial for other game modes and games with a similar style.

2. Layout

Before you start sketching your layout, you should know what kind of style you are going for. Do you wish to create a classic 3-lane layout like Dust2 or Mirage, a more asymmetrical map like Inferno, or do you aim for a stacked layout like Nuke? Either of these could provide the backbone for a good layout, although the classic 3-lane layout is probably the easiest to balance.

2.1 Meeting points/Battlefronts

The meeting points in a level are the areas of the first enemy encounters in each round. Since the classic competitive gameplay in CS is based on 5vs5 games, you must shape your layout accordingly. It's recommended to have about three battlefronts – the sweet spot. Two might not give the attacking team enough options to work with, while four might make it too hard for the defending team to stop five rushing attackers.

There are a few console commands which assist you analysing the battlefronts.

"bot_show_battlefront 1" will display the areas where players meet if they rush with their knives out.

To track the battlefront movement and see how the former command identifies the areas,

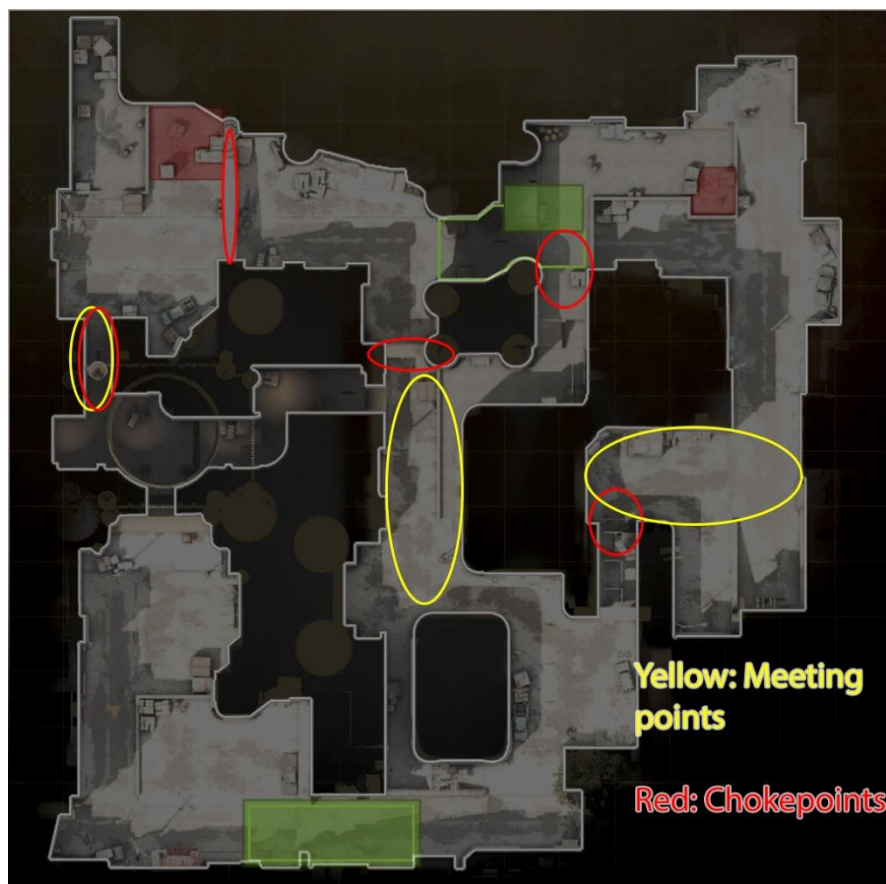
"bot_show_occupy_time 1" will display the moving fronts. Note: **"mp_restartgame 1"** must be used for the latter to work. Also, the performance will be significantly worse with this command in effect.

2.2 Chokepoints

Chokepoints separate different areas in a level. They are usually implemented using tighter paths (connectors/hallways/doorways/tunnels) compared to the rest of the layout. Teams must fight through these bottlenecks to gain control of areas behind them. While you should be very careful regarding the number of battlefronts, chokepoints offer more freedom in their design. The total number of chokepoints is usually higher than the number of meeting points since they are mostly located behind them and routes might split up in between.

For example, Dust 2 mid splits into A short and mid double doors, while the meeting point is right before it. This is a good way to create more major routes without messing up the sensitive gameplay of Counter-Strike.

Remember, all objectives should be placed behind chokepoints – as seen from the attacking side – to ensure a good balance between attackers and defenders.



2.3 Staging Areas

Naturally, the defending team on an objective has the advantage over approaching attackers. To accommodate this imbalance, there are utility grenades in Counter-Strike which allow players to gain a tactical advantage in the process of taking or defending a bombsite.

In order to promote the usage of tactical preparation and execution of strategies, implementing staging areas is an elegant way to achieve this. Staging areas are areas in front of chokepoints – usually with an open ceiling – where players can prepare and throw (utility) grenades like Molotovs, smokes and flash grenades. The attacking team can also use these fairly safe areas to plan their following strategy.

2.4 Bombsite entrances

The areas where the objectives – the bombsites - are located, must be taken by the attacking team. The entrances to these areas have to be crafted carefully to allow balanced yet interesting gameplay.

Similar to the amount of battlefronts in a map, the number of entrances into a bombsite has to be chosen wisely. Having around three of them is always a good starting point, but as Inferno B site with few entrances or Train A site with many entrances show, other approaches can work as well. The more entrances you create, the less powerful each entrance should be. This also means that you'll need to enable more tactical possibilities if you implement fewer entrances.

2.5 Post plant areas

After the C4 has been planted on a bombsite, the roles of Terrorists and Counter-Terrorists shift. Terrorists must now defend the planted bomb for 40 seconds, while Counter-Terrorists are forced to retake the bombsite and defuse the bomb. Since the opposite team is now approaching the objective, the direction of the attack likely as well. Make sure to accommodate this direction change when placing cover on the bombsite. The cover placed so far might only be for the initial attack by the Terrorists and not for the direction change post-plant as well. Therefore, it's important to think about playable positions from a post-plant perspective, too. Such areas can be little cubbies around the bombsite or connected routes which serve to aid the attackers or defenders.

2.6 Simplicity > Complexity

Complexity in layouts is often caused by too many routes, connectors and other options how to move through the map. If a layout offers too many options, players get confused and gameplay suffers as well. This is something you clearly want to avoid.

The most successful maps are simple and easy to learn. Arguably the most popular map - Dust 2 - has the simplest layout out there. Complex maps with many different paths and connectors have the risk of quickly becoming painful to navigate.

Before adding a new route, which might make things more complicated than needed, evaluate if it's benefiting the map. It's recommended to add new routes during development if playtests show something is missing, rather than starting off too complex.

However, the risk of too simple layouts is the high chance to become stall during the map's lifetime. The simplistic Dust 2 sometimes gets this resonance from the player base.

Finding the balance between overly complex and too simplistic, boring layouts can be very time-consuming but will pay off in the final product.

The design strategy of complexity also translates to potential excessive placement of cover/props.

Do as little as possible, and as much as necessary.

2.7 Unused Space / Areas without serving a purpose

A map should be accessible to new players. Reduce the unused playable space to a minimum. This will help guide the players through the map intuitively. For example, if you create a turn in the layout, don't build an inlet on the non-accessible side of it to avoid disorientation.



Also, if you look at Nuke, the long narrow spawn area is a thing you usually want to avoid in maps since it's serving almost no purpose. This kind of space wastes time unnecessarily. However, providing areas on the map where players can survive the bomb explosion or save their gear, are a purpose – keep this in mind.

2.8 Negative space

The term negative space refers to inaccessible areas on a map. You want to design your level in a way, that a large enough portion of your level space fits into this definition. The reasoning behind this is, that accessible areas get a large enough purpose without being just a substitute to an alternate area right next to it. Another reason relates to the sounds of CS. Players make use of sounds to pinpoint enemies' locations. If multiple areas are close to each other, it's hard for players to avoid being heard by potential enemies in the bordering area.

2.9 Support various playstyles

The diversity of different weapons in Counter-Strike is another thing you should consider. A good map supports various playstyles for all the different weapons: Possible indoor areas with short distances and several corners for close range guns (shotguns, MPs), long sightlines for sniper rifles and mostly medium ranged distances since rifles are the most commonly used gun type.

2.10 Allow advanced tactics and teamwork

Since the game heavily relies on team play, your map should encourage and facilitate it. For this, you can enable some boost spots to reach more powerful positions. Furthermore, smoke grenades and flashbangs are an essential element to use when executing strategies. Enable options to throw them – keep a relatively open skybox and build windows to achieve this.

Referring to the earlier paragraph “*Simplicity > Complexity*”: Too many of these will increase complexity and could harm the gameplay.

2.11 Wingman specific chapter

The design of wingman layouts is for the most part very similar to the 5vs5 game modes. However, due to the smaller number of players – 2vs2 – the amount of routes and general size must usually be lower. A good starting point for the routing towards the bombsite is to create 2,5 entrances for the attacking team. While this sounds strange at first glance, having 2 major separate entrances plus one entrance, close to one of the major ones, is usually a nice sweet spot for this game mode.

3. Routing

3.1 Avoid obstructions

Players in Counter-Strike are always focusing on positioning, crosshair placement and tactics. This implies, that basic movement around the level mostly works on an intuitive level without actually looking where players are going. To allow players concentrate on the greater things, movement should be hindered as little as possible. Main travel routes must be free of obstacles and collisions as smooth as possible. Keep floors in these areas smoothly even and move detailing to the sides to keep paths clean.

4. Flow (Chapter written by @oliver_irl)

The kind of flow important in CS:GO level design is about flow of movement and action.

4.1 Natural guidance

Examples of flow of movement is when the player is lead forward and not backwards. You want to move towards the opponent and the objective so the level shouldn't be designed in a labyrinth kind of way, but instead one area should flow naturally into the next.

You want the player to feel like they are in control and give them the opportunity to make decisions on the go, so the overall goal should be to make the flow of movement smooth so that the player can always be in motion.

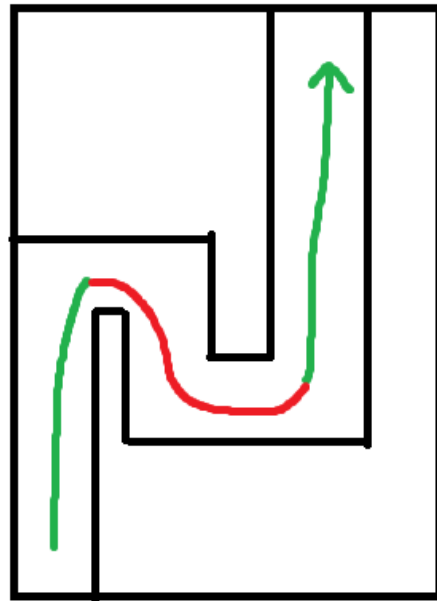
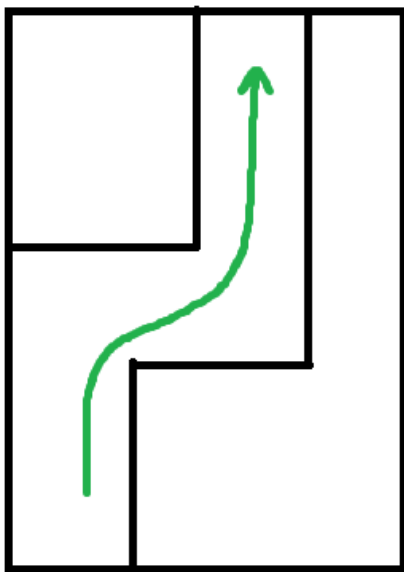
Examples of flow of action is what options the player has in the event of an encounter. In CS:GO you have to think about the map holistically [/as a whole]. Everything is interconnected, so every area can

be an isolated "war zone". If the level has enough cover and options to use utility, then that contributes to good flow.

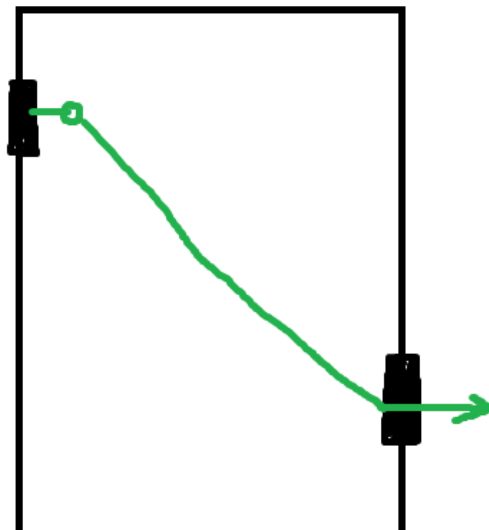
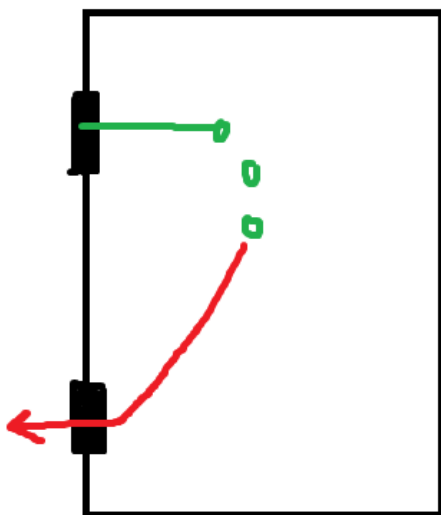
4.2 Decision-making

Flow is about decision-making. Do you let your players play the way they want? Do you feel in control when you enter a bomb-site? You don't really notice when levels have good flow in them. Bad flow can be recognized once certain parts of the map feel uncomfortable for the player and the map doesn't allow the player to make decisions.

You can see that the movement is disrupted in the second example and the player is moving backwards for a moment.

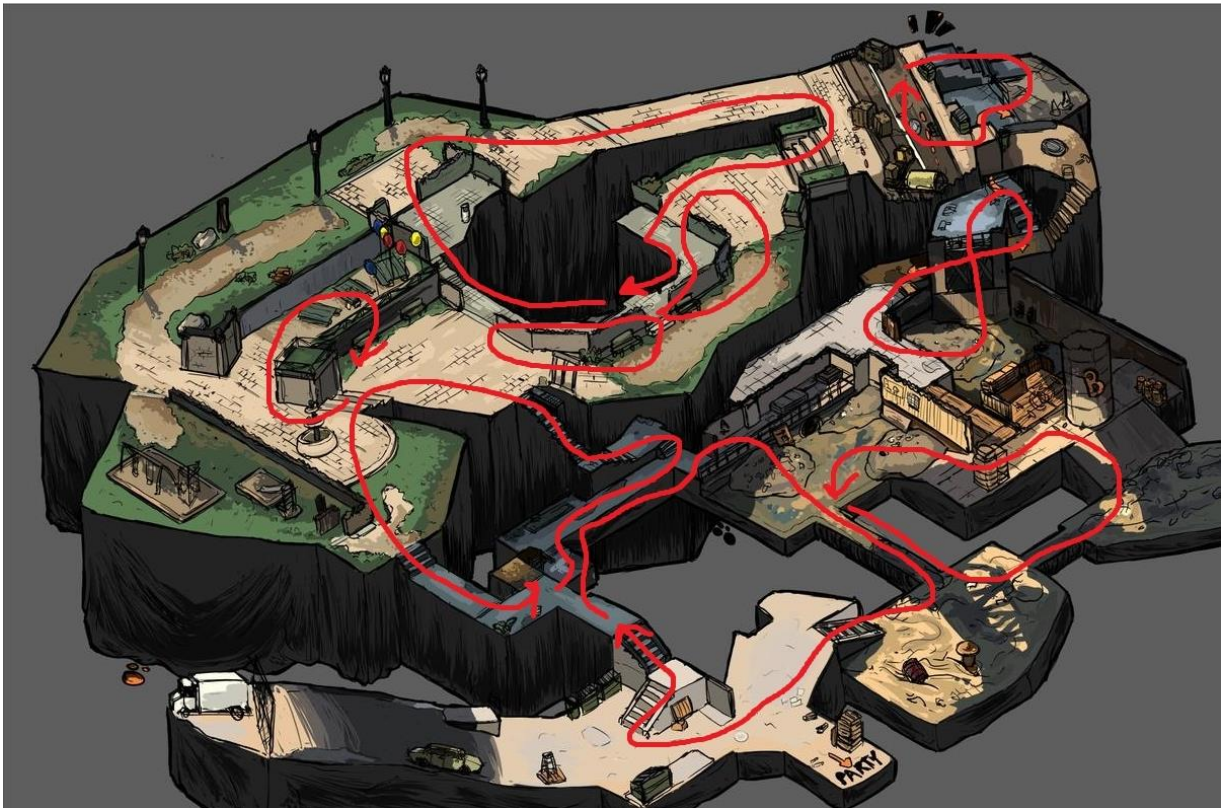


Guiding players naturally in the environment contributes to good flow, and players don't have to stop and think about where to go. In addition to that it keeps the movement going forward.



4.3 Loops

Loops are especially important for CS:GO, since you can use them to get better positioning on your opponent. They are so elegant they work when you want to take a bombsite as a terrorist, or hold the site as a counter-terrorist. Players use them to fall back if you lose an engagement, and loops give players more than one option at any given time.



5. Navigation/Intuitivity

5.1 Landmarks

Subconsciously, players take in the rough look and shape of their surroundings to find their way through an environment more intuitively. Therefore, many maps rely on landmarks. Landmarks are unique, mostly large, structures which are visible from large portions of a map. Having a large focal point like this available makes it easy for players on a new map to get the grasp of a layout quicker than without such a landmark. A great side effect of landmarks is the possibility to align grenade throws by putting their crosshair somewhere on the structure. A prime example for landmarks is the TV tower on Overpass.

5.2 Roof detailing/Alignment hints

Learning how to use utility grenades on many maps can be quite a time intensive task. In order to make the learning process as accessible as possible, make use of detailing above the playable area in a way, that objects help aligning grenade throws. One example how to it, is the placement of antennas on rooftops.

5.3 Detailing

Contrast and detailed areas attract players. Use this knowledge to guide players through a level as much as possible. Highlight and detail accessible doors, corridors and other points of interest. Tint usable doors in a certain colour while leaving inaccessible doors in shades of grey or rather muted colours. Keep the detailing and contrast in non-accessible areas at a low level to avoid disorientated players.

5.4 Consistency

Players should never be confused by all kinds of aspects in level design. Intuitive navigation through gameplay space requires consistency in design decisions. An example for this is the colour coding of interactable elements such as doors. If you decided that an openable door is tinted in a vibrant colour such as red, all openable doors should be tinted with the same colour.



Highlighted accessible door on the community map Thrill

6. Timings

6.1 General

Measuring the timings of a layout is important to check if it is competitively viable or not. The whole Counter-Strike gameplay is based on timers like the round time (1:55) and C4 timer (40 seconds). Having the right timings in a map helps a lot in balancing.

6.2 Battlefront timing

Mentioning an exact number for the timing of the first enemy encounter is almost impossible, since it's varying a lot depending on the actual layout, and different values may work fine. Having this in mind, there's the following: Reaching the chokepoints in about 5-12 seconds into the round is a good value to start with. Obviously, the Mid timing in classic 3-lane layouts will often be much shorter. However, comparing Dust 2 and Mirage, the chokepoint timing in the middle differs a lot. There are other factors besides timing which determine the balance or quality of a middle lane.

Besides any specific number of seconds, the defending team should have a little bit more time on their side of chokepoints, to get into positions before the attackers are able to execute an attack.

6.3 Avoid wasted time

Taking Nuke as another negative example, the large amount of dead space in the spawn areas lead to a lot of wasted time during a whole match. This could have been easily avoided by moving both spawns 3-4 seconds closer towards the core area of the map. At some point during the last years, Valve moved the spawns a bit closer to the core of the map, but there is still plenty of wasted time left.

6.4 Rotation time

The rotation time (bombsite to bombsite) of a layout is probably the most important timing which must be considered. The main reasoning for this is the C4 timer (40 seconds) during after-plant situations. A too short rotation time strengthens the Counter-Terrorist side too much while too long rotation times on the other hand, make it almost impossible for CTs to win a round as soon as the bomb is planted as they won't have enough time reaching the relevant bombsite. If the rotation time is in the range of 10-15 seconds, it should be working for most layouts. Although maps with stacked bombsites, like Nuke, prove that it's working differently as well. For these cases, narrow and/or risky rotation paths with limited tactical options are an option to balance the short rotation time out. If you prefer longer rotation times, try to keep it below 20 seconds as it becomes very hard to retake bombsites with very little time left.

6.5 "Around the world"

The "around the world" timing of a layout is probably the least important of all the timings, but it helps getting the scale of a map right. Depending on which size your map will be, choose a map out of the official map pool most fitting your planned scale and measure the time it takes, to run the longest way possible around the map. If you are happy with the values, continue mapping.

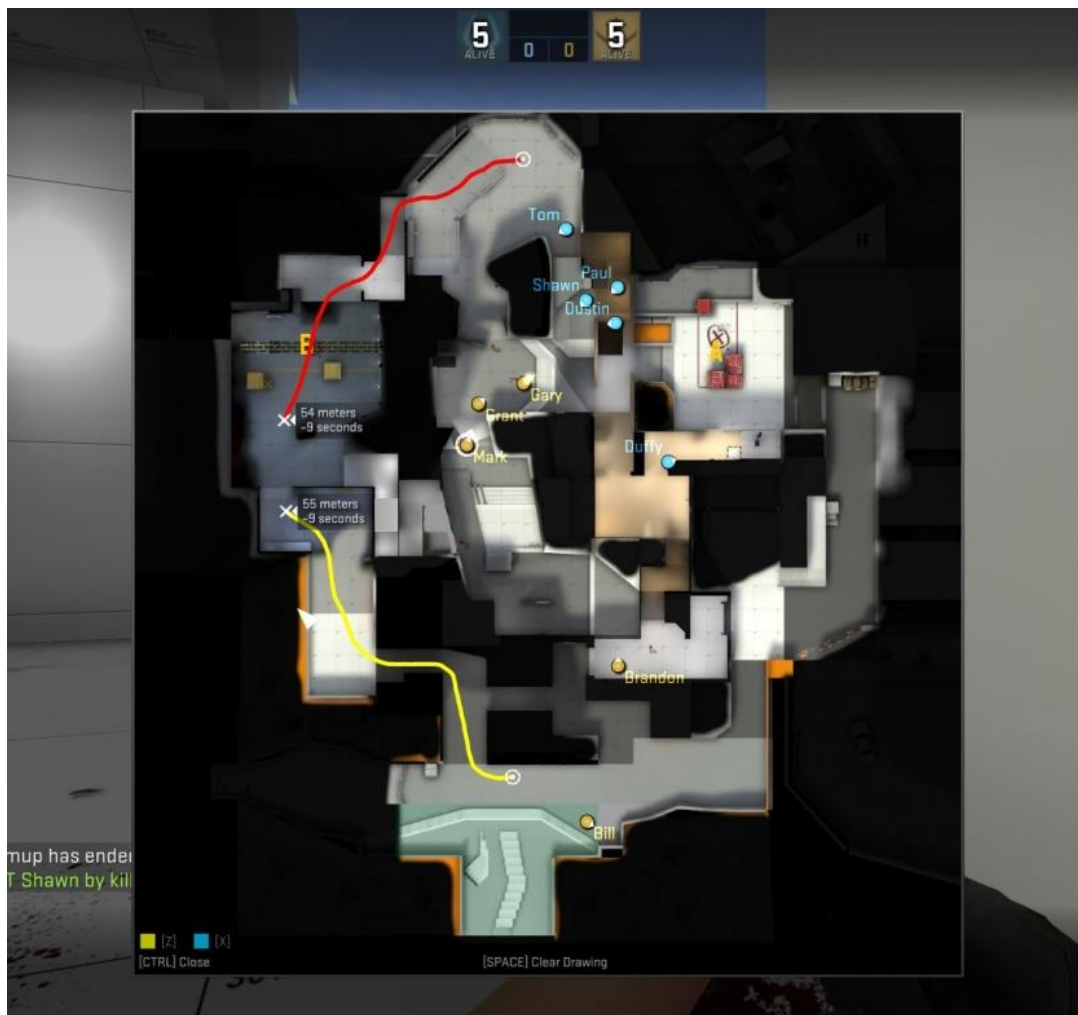
6.6 Measuring timings

A very quick and elegant method to check the timings without being forced to use a stopwatch is hidden in the spectator option. Before it is possible to use this feature, the following console command must be set:

[Feature broken currently, however, still being mentioned in case of a fix.]

- `mapoverview_allow_client_draw 1`

After this command has been activated, open the map by pressing CTRL as a spectator. Hold shift and paint a path with the left mouse button. The resulting path will display the distance and time it takes to walk along this line.



7. Risk and Reward

7.1 General

Rewarding risky plays and positioning is a great way to make gameplay interesting and diverse. The challenge for you as the level designer is to craft positions, routes and bombsites with the target to reward the player if he was able to master a risky situation.

There are two prime examples to showcase the system of risk and reward on the map Cache:

- Positioning: The boost spot on the boxes next to A main is risky to get on, but very rewarding to catch unaware attackers off guard.
- Bomb plant zones: If the bomb is planted on the mid-facing side of the red container on A, it's almost impossible for the CTs to safely defuse it. But since this specific spot is super risky for the Terrorists to plant on, it's a perfect example of showcasing risk and reward trade-offs.

7.2 Risk and Reward via route design

As mentioned in the previous chapter, a massive portion of Counter-Strike's gameplay evolves around timing. It's usually beneficial to get as fast as possible from A to B. In order to not overly strengthen certain quick paths, the method of risk and reward can be applied to the route design as well. Quick routes could involve penalties such as fall damage, while longer options avoid such penalties. An example of this could be a large staircase, where jumping directly through to the bottom saves a few seconds while the player gets hurt along the way.

7.3 Risk and Reward via sound design

To address one aspect in advance, making use of sound design to promote risk and reward on your map is an option as well. Instead of hurting the player taking a quick route, the player could be spared from damage and is instead forced to emit sounds to his surroundings which might make opponents aware of his presence.

More on that topic follows in the chapter "Auditive Design".

8. Sightlines

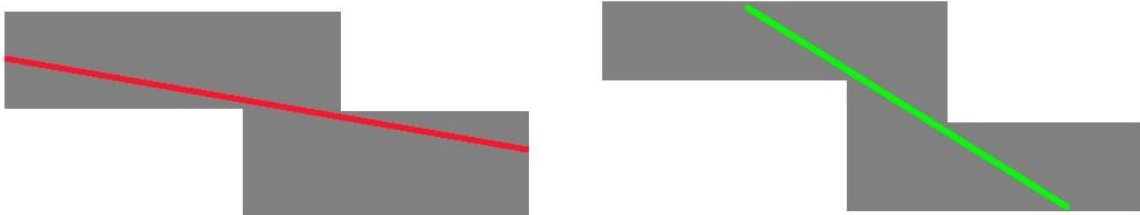
Lots of fights in Counter-Strike take place around corners, therefore you, the mapper, must pay some special attention to the various angles in the level.

8.1 Long sightlines

It's recommended to avoid super long sightlines, where it's only possible to make frags with a sniper rifle. The Dust 2 spawn to spawn sightline is ignoring this, but it is working fine there, because early round picks shouldn't happen with every type of assault rifle. You must own a rifle dedicated for long range battles. The Terrorists also have an option to avoid this sightline and enter the mid through a more central path. The remaining sightline is so long, that you can achieve frags with an assault rifle as well. Since CTs aren't supposed to get active mid control early in the round, they don't need the possibility to frag enemies from spawn to spawn with an assault rifle. That being said, I personally do not recommend to create such a spawn-to-spawn sightline.

8.2 Tight angles

When blocking out a map, it often happens that tight angles are created by accident and enable long and overpowered sightlines. Luckily they are easy to fix by moving the causing corners a bit.



8.3 Pixel angles

Like tight angles, pixel angles are a result of slightly misplaced corners. These types of angles are questionable for multiple reasons including optimization, unintuitive gameplay and unfair advantages. An example for such an angle is in the sightline from the B balcony on Mirage all the way through apartments:



8.4 Vertical sightlines at ramps

When creating ramps or elevation changes, it is important to think about the line of sight between players. If the player on the upper part of a ramp is standing behind cover, he might be able to see the player on the lower part, without being seen by the opponent - if it's done wrong. To show this off more clearly, I found these examples on Dust (1) and Cobblestone. When a Terrorist on Dust is coming straight through the underpass area, the Counter-Terrorist on the upper area is able to see the enemy's feet without being seen himself. On Cobblestone on the other hand, the underpass area is created in a way that the attacking players are side-peeking towards the upper area of the big ramp. This way both parties have the same chances in a firefight without massively unfair advantages.

Don't:



Do:



9. Verticality

Height differences and multiple layers belong to the easiest and most effective techniques to make layouts more interesting. In a lot of other FPS games besides Counter-Strike you don't have to worry much about problems this may cause. In Counter-Strike, however, too much verticality is considered a bad thing. Multiple layered maps make it often hard to distinguish enemy players' positions via radar and/or sound.

As a player, any vertical angles which exceed the $-30^{\circ}/30^{\circ}$ range (no specific values, rather a rule of thumb) are not really enjoyable to play around anymore. Firefights should mostly take place in a horizontal level, where the distance your crosshair travels is consistent. Always keep that in mind when adding different height levels.

10. Auditive Design

10.1 Spatial awareness

In certain circumstances when multiple routes or areas are in audible range of each other, may it be vertically or horizontally, level designers have some options available to assist players with their spatial awareness. Listening to and interpreting auditive cues well, plays a quite important role for the outcome of a match. A simple but effective method to help players locate enemies' footsteps is the usage of dedicated floor materials for certain areas. If, let's pretend, player A finds himself on A site of Nuke and is listening to footsteps outside towards CT spawn, player B could walk either on the concrete ground level floor or on the metal grated upper catwalk. Simply by listening to the sound of the footsteps, player A can pinpoint the location of his opponent which might change the outcome of an upcoming firefight.

10.2 Environmental Audio

Since players have the ability to get much information just by listening to the games' audio, you have to keep in mind that you don't want to take this away from them. Too loud environmental sounds like wind, traffic, car horns, lightning strikes, AC units, etc. can make it very hard for players to distinguish environmental sounds from more important sounds like footsteps, reloading noise and more.

10.3 Sounds of interactable Objects / Triggered sounds / Positional hints

Another option to involve audio into the level design of a map are triggered or area specific sounds. One use case for such sounds is to balance out overpowered routes. Metal detectors or other triggered sounds in positional/tactically powerful locations can raise awareness of players in certain locations. However, making use of triggered sounds must be done carefully. Too loud sounds, a too large audible distance or excessive use of these might cause imbalances and/or unenjoyable gameplay.

10.4 Allow sneaky plays

On some occasions, people like to outsmart their opponents. This includes preventing the opponent from gathering any information about your location. Sneaking (shift walking) players don't make any sounds if their route doesn't involve falling from too high ledges. Falling from heights of 46 units and above causes a dropping sound. Falling crouched (incl. uncrouching mid-air) from 55 units and above will cause a drop sound. Keep this in mind when creating routes around the map.

11. Cover

11.1 Avoid Head peeks

When a player is barely able to look over cover, it is called a head peek. If an opponent is encountering a player behind such cover, barely half of the player's head will be visible to the opponent. As a result, the encounter between these players leads to a frustrating and unfair firefight.

Creating head peek cover is one of the most common mistakes mappers do. The reason for this is simple. The default grid size in Hammer is 64 units and the height for head peek cover is 64 units as well. Gameplay, sightlines and firefights around these are very strange and not enjoyable at all. It's recommended to use below-head cover (~56 units) and above-head cover (~72 units) like on Dust2 A site instead. But not only those classic cubic boxes are enabling them, misplaced ramps and stairs often create head peeks, too. Try keeping them to a minimum.

11.2 Natural Cover

Most Counter-Strike maps utilize crates and boxes to create cover. Unfortunately, some of them rely too much on it, which feels unrealistic and repetitive pretty fast. Whenever it seems possible to integrate cover into the architecture of a map, do it. This does not mean using boxes as cover is a bad thing. It just should be balanced out, so the map is looking like a believable space.

11.3 Overpowered Cover

When adding cover to a map, it's important to not overdo things. Some level designers mistakenly create too many powerful spots without playtesting beforehand to see if there's even the need to do so.

A possibility to limit the strength of a hiding spot is to be not covered towards all possible angles. A good example for this is the Dust 2 A site. Most of the common positions offer cover for 2 of the 3 bomb site entrances. This way the defender has enough cover to work with, but not enough cover to always feel safe. A lot of maps prove that some more powerful cover is working as well though. If you really want to add some powerful cover to your map, there are still possibilities to handicap it. These areas could be crafted like a death trap, without an easy way to leave them - shall they be contested with an incendiary grenade for example. This disadvantage will even out the fact, that players hiding there can't be seen from any of the entrances into the corresponding area. A fitting example for this is the "ninja" corner on Mirage A site.



12. Models/Props

To make a map feel like a less blocky and realistic place, the usage of props is very common and recommended. There are few things which must be kept in mind while placing them though. Small detail props like lamps should always be non-solid in order to allow fluid movement and not collide with any thrown grenades. Another reason to do this is to prevent unintended boost spots. Furthermore, all types of railings should always be non-solid and clipped instead. The reasoning behind is, that it's super frustrating when grenades get blocked by these thin models.

Don't:



Finally, solid props in main gameplay areas should always have a correct hitbox (unlike the tree on an older version of Overpass in the upper park).

Don't:



Do:



12.1 Model shape and model collisions

Some props are also very bad for smooth movement with their default collision model. If we compare the truck model on the A site of Inferno's old version with the hay wagon model on the new Inferno, there clearly has been a massive improvement. Although the truck model offered a mostly accurate hitbox (which is good) movement on it has been super clunky and frustrating. If done correctly, a few clip brushes should have been added, to enable a smoother movement. In addition to that, the truck was very bad for gameplay because of strange sightlines through it and next to it. Avoid models like this in main gameplay areas.

Don't:





Do:



13. Scale/Dimensions

To get the scale right when doing the layout, it's handy to know roughly how big the map should be. If you're unsure where to start with the scale at all, load up the .vmf file of a similar sized map in Hammer and draw one big brush covering the whole playable area of said map. The resulting brush can be pasted into your personal map.vmf. This brush will help you not exceeding the maximum desired size of your layout while blocking out. The playable area of Dust 2 for example, is roughly within 4000x4000 units.

Supplementary to the overall scale of a map, there are plenty of other dimensions which will help you create a working layout. Note: These are all rule of thumb values to start with. You must adapt your map depending on feedback and the architecture you want to build. It's recommended to keep hallways relatively wide, so movement doesn't feel too restricted and teammates are not blocking each other too often.

- Corridor width: 128 units – 256 units
- Corridor height: (never below 128 units) 144 – 192 units (depending on the architecture)
- Average indoor wall depth: 8-16 units
- Average outdoor wall depth: 16-32 units
- Vent-size crouch only: about 60x60 units; Vent-size standing: about 80x80 units
- Stairs: 8x12 units; Never steeper than 45°
- Doors: 48x108 units; Doorframes: 56x112 units (Preferably double doors for fluid movement, however, double doors need to open both doors at the same time)
- Player dimensions: Standing: 32x72 units; Crouching: 32x54 units
- Jump height: 54 units; Crouch jump height: 64 units
- Silent drop height: Up to 46 units for standing drops; Up to 55 units for mid-air uncrouched ones
- Maximum walkable obstacle height: 18 units (avoid high steps like this)
- Smoke size: 288 units diameter

14. Grid

In the early stages of creating a map, grid sizes between 8 and 32 units should be used preferably. Reserve the smaller grids (1, 2 and 4) for detailing and finalizing brushwork. It's way easier to work in Hammer having used the larger grid before.

As soon as you start detailing, don't be afraid of the 1-unit grid anymore. A lot of detailing can be achieved using this grid. However, mesh based detailing is usually preferred for visual reasons.

Independent from the actual grid size, it's heavily recommended sticking to the grid whenever possible.

15. Visibility

15.1 General

Counter-Strike is a very competitive game where every player needs to see easily what's going on. That implies that player models must stand out from the environment. Avoid noisy textures, dark corners and colours similar to the ones used on player models in main gameplay areas. The same applies to foliage which makes it very hard to spot other players. Keep noisy detailing above head level to ensure visibility is as good as possible.

15.2 Environmental Lighting

To further help visibility, good **light_environment** entity settings are crucial. The default entity values are:

- Brightness: 255 255 255 **200**
- Ambient: 255 255 255 **20**

Especially the Ambient value is something that should be changed for maximum visibility. I'd recommend these brightness values for daylight settings to start with (Depending on the chosen setting):

- Brightness: 255 255 255 **250**
- Ambient: 255 255 255 **150**

The ambient light value works as following: Instead of calculating the brightness on a surface based on the direct line of sight to the sun direction, the ambient value calculates the light a surface receives from the surface amount of the sky(box) it can "see". The larger the exposed area of the sky is to the receiving surface, the more lit the respective texture will be. This implies that no matter how bright your ambient light value is set, fully enclosed rooms will always stay dark. If you want to fake a believable ambient light for enclosed areas, make use of fake lights. Fake lights are light entities with a constant light value usually set to really low brightness values.

The official maps tend to use much higher brightness values than the ones mentioned in this guide so far. Be encouraged to experiment with the settings to achieve the best values for your needs.

Example *light_environment* values from official maps:

Dust2:

- Brightness: 254 230 197 575
- Ambient: 211 226 248 245

Nuke:

- Brightness: 255 240 206 710
- Ambient: 101 139 182 550

Inferno:

- Brightness: 255 238 198 640
- Ambient: 114 149 188 420

Overpass:

- Brightness: 253 217 162 620
- Ambient: 181 201 232 220

15.2.1 Colouring

To set the desired mood on your map, you likely won't use the default white colour of the entity but change it to a more fitting alternative. The values you are going for heavily depend on your chosen setting. There is only one thing to keep in mind: Be very careful with the colour intensity. It's recommended to be very gentle with the offset from white as colours are quickly perceived as too extreme in-game.

16. Spawns

For competitive gameplay you'd just need 5 spawns per team. In order to make it possible for community servers to use your map, there should be 16 spawns per team instead. This amount of spawns might mess with the timings of your map if you haven't placed your spawns very carefully though. To bypass this issue, it's possible to prioritize them. Simply change the "Spawn Priority" value of the chosen spawn entities. A lower number means higher spawn priority.

It is recommended to stay organized and consistent with the values. The default value is "0" and I'd suggest to not use the default value. The reasoning behind using values of greater than "0" are temporary placed spawn entities in a level. These temporary entities will have the default priority value and therefore always be used before your actual spawns. Any left-over spawns will be identified right off the bat using this method. Use the following values for the different types of spawn entities.

- Competitive: 5 spawns with a value of "1"
- Casual: 5 additional spawns with a value of "2"
- For any community servers with more than 20 player slots: "3"

Looking at Dust 2, the Counter-Terrorist spawns are always the same (prioritized) while the Terrorist spawns are randomized which causes every round to be a bit different in terms of timings.

Another thing which not enough level designers do, is rotating the spawn entities, so spawned players face the correct direction right from the start. Also, to avoid having invalid spawns, it is important to always place your spawns a bit elevated from the ground (16 units are fine). This is something even professional mappers forget in rare cases.

17. Buy zones

There is not much to say about buy zones except for one thing: The buy zone trigger brushes should be large enough, so players are able to buy on the fly. Many players are using buy binds to buy guns and accessories after they started walking in the beginning of a round. The zone shouldn't end right after the spawn entities to allow this. If they are a bit larger, it's not game breaking, but a lot more comfortable for players.

18. Clipping

As mentioned earlier, you want to ensure smooth movement as much as you can. The usage of clip brushes is crucial to achieve this. First of all, always clip stairs. If this isn't done, the players will not enjoy walking on them, since the clunky movement on unclipped stairs messes with the precision of the crosshair. It's also significant to clip small corners and brushes like shown in the screenshot below. The reason for this is blind movement. In an in-game situation where lots of utility grenades like smoke grenades and flash grenades are being used, it is vital for players not getting stuck on small corners and edges when they are unable to see where they're moving.

Besides that, there are more reasons why you should clip thoroughly. Clip brushes block the C4 and therefore can avoid bomb grief spots. Also, game breaking boost spots are history if clipping is done right.

Furthermore, it should be avoided using the Player Clip texture since it's messing with the generation of the nav-mesh and won't affect the actual purpose of a clip texture. Instead use the normal, red, clip texture.

A very useful bind for collision testing on your local server is the following. These commands will not only toggle clip brushes on and off, it will also display collision models of props. Testing maps like this will allow you to find game breaking bugs easily before any playtests.

- `bind "HOTKEY" "toggle r_drawclipbrushes 0 2; toggle vcollide_wireframe 0 1"`

Do:



19. Basic Optimization

In the very early stages of prototyping, optimization is not really an important thing. Until the very basic shape of a layout is created, it's ok to work with no proper skybox, because changes are way faster and easier to apply. This can quickly be achieved by using the cordon tool. However, as soon as the basic brushwork is completed, it's good to start caring about it. Set small and non LOS (=line of sight) blocking brushes as func_detail and start creating a proper skybox. Another rather simple optimizing technique is to disable collisions on props further outside the playable area. Doing these things will not only improve performance but also reduce the compiling times of a map significantly. A well optimized map can run well on a low-end system while poorly optimized maps often have trouble on medium to high-end systems. A detailed guide on optimization is linked down below since this is not the main goal of this guide.

20. Presenting your map

In order to get real feedback, a map has to be presented properly. It is not enough to post *"Check out my map, need feedback ASAP pls. Here's the Workshop link"* in a forum. Your map presentation must arouse interest. Make some screenshots of your map and chose the one you think looks the best overall. This is the one you should embed into your post. Explain what you want to achieve with your map, what you have already done and maybe even show one or two references you are planning to make use of. This way the reader can imagine what the map could look like eventually. Boring pictures of an untextured layout are often meaningless to an outsider. Once the interest of a reader has been aroused (hopefully), it's time to show the layout overview. This is not something you want to miss, since basic feedback is often already possible simply by looking at a radar picture. The radar should contain both spawn locations and both bombsites - including names. In the official Valve developer Wiki, there are radar resource files so you can edit the radar presentation on an advanced level.

- https://developer.valvesoftware.com/wiki/Creating_a_working_mini-map#Resource



For creating the showcase screenshots, I collected some console commands which should be used when creating them. Also, maximize your video settings for the best possible representation of your map.

Map screenshots:

- sv_cheats 1
- r_drawviewmodel 0
- cl_drawhud 0
- net_graph 0
- noclip

Radar screenshot – old method (in addition to the commands above):

- cl_leveloverview 4 (Values between 3 and 6 usually work fine. Set it to 0 to disable it)
- r_novis 1

Radar – new method:

<https://github.com/Terri00/CS-GO-Auto-Radar>

TAR (made by Terri) is a tool for an automated radar generation. All you have to do for it, is assigning elements of your map to visgroups. After setting up TAR in the compiler, a radar will be created every time you compile your map with it enabled. This method will ensure that your radars are as accurate and as readable as possible.

To wrap the map presentation up, additional screenshots could be posted inside of a spoiler so interested readers have more footage to look at if they want to.

21. Playtesting

There exist several playtest services for the CS:GO mapping community. Most notable are the following two options:

- Mapcore Playtesting:

<https://www.mapcore.org/topic/20518-mapcore-csgo-playtesting-50>

- Source Engine Discord Playtesting:

<https://www.tophattwaffle.com/playtesting/>

In order to get the best possible out of playtests, your map must be prepared properly. These are a few requirements all maps should fulfill:

- A working radar
- A working .nav mesh
- Suffixes for .bsp names to avoid version conflicts
- 10+ spawns per team
- Packed custom content
- A working game mode
- Clipping for smooth movement and to prevent game-breaking boosts
- Basic optimization
- Roughly working timings

Playtest in the right setting.
Playtest with everyone.
Don't explain anything.
Always playtest the whole experience.
Make those 1-minute changes.
Filter feedback, always.

22. Dealing with feedback

Mapping newcomers often crave for feedback, but don't really know how to deal with it. What you secretly expect, are people saying that your layout is awesome and could be the next Dust 2. Unfortunately, this will most likely never happen. Sometimes feedback will be harsh, but you shouldn't let yourself be discouraged by that. If people are harsh with their feedback, there must be some reason for it and only shows the urgency of changes and that things can't stay as they are. Counter-Strike is a competitive game and therefore people might become emotional very quickly. If you ask these people to explain their feedback a bit more detailed, most of them will respond nicely and help you fix the flaws a layout may have. Don't respond that you feel mistreated. It's in the nature of CS that players get annoyed by poor design decisions. You, the mapper, must learn to deal with feedback like this.

"Feedback" à la "Valve, add this pls" is pretty much useless. Sure, it's nice to read, but this is no useful feedback at all. Personally, I'd rather see someone complaining that the map's balance is "crap", than just telling me "good map". Level design is very iterative and therefore every mapper should be happy when people showcase the flaws a layout may have. Accept feedback and consider changes. Don't be ignorant with a mindset, that your layout is already perfect. If all you want to see are compliments, don't ask for feedback.

The above being said obviously only applies, if you actually did receive feedback. This is one of the reasons I created this guide. Aspiring mappers should have some guidelines to work with, while missing feedback from other players.

23. Further guides and tutorials

Source Engine Optimization Roadmap (will2k): <https://www.mapcore.org/topic/20087-source-engine-optimization-roadmap/>

Source Lighting Technical Analysis: Part One (leplubodeslapin):
<https://www.mapcore.org/articles/development/source-lighting-technical-analysis-part-one-r65/>

Source Lighting Technical Analysis: Part Two (leplubodeslapin):
<https://www.mapcore.org/articles/development/source-lighting-technical-analysis-part-two-r66/>

How to Develop a Map That Works (Valve):
https://developer.valvesoftware.com/wiki/How_To_Develop_A_Map_That_Works

Quick Guide to Competitive CSGO Design (will2k): <http://gamebanana.com/articles/74>

How are maps chosen for Operations? (Valve):
<http://steamcommunity.com/games/CSGO/announcements/detail/230003535507901582>

CS:GO 6 Principles of Choke Point Level Design (World of Level Design):
<http://www.worldofleveldesign.com/categories/csgo-tutorials/csgo-principles-choke-point-leveldesign.php>

Collection of resources for level design (jackophant):
<http://www.mapcore.org/topic/18911-attn-new-map-makers/>

Level Design Compendium - Trello Board (@RazorOfArtorias)
<https://trello.com/b/AM3ZOmAd/level-design-compendium>

TopHATTwaffle's tutorial collection: <https://www.tophattwaffle.com/tutorials/>

TopHATTwaffle's CS:GO SDK bootcamp series:
https://www.youtube.com/playlist?list=PL-454Fe3dQH0WCzAsmydsr24NFaFrNC_h

GDC Talk about CS:GO level design by Volcano and FMPONE
<https://www.youtube.com/watch?v=Y5RZP52jYy4>