A Counter-Strike 1.6 map with J.A.C.K.

2026-11-03

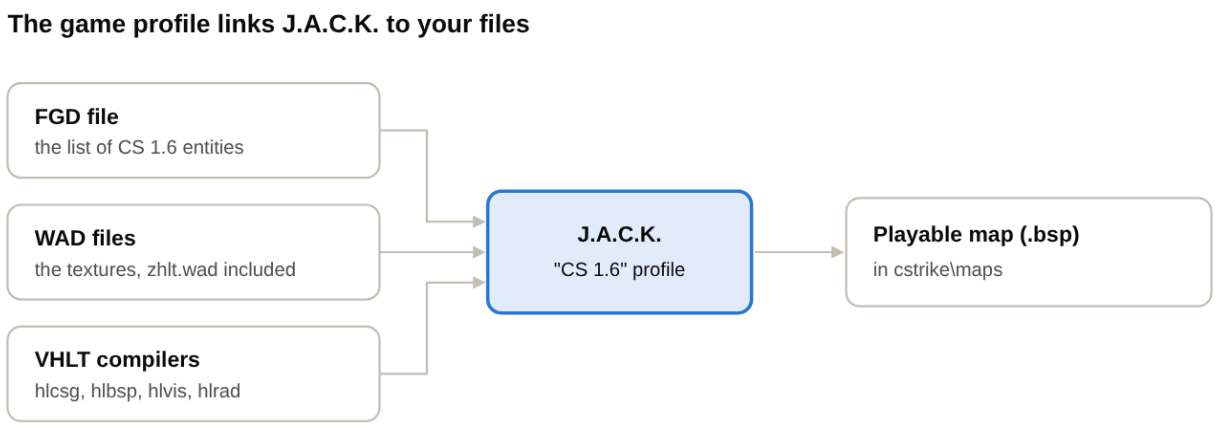
This course teaches you how to create a Counter-Strike 1.6 map with the J.A.C.K. editor, from installation to compilation. It follows the outline of the French tutorial "Une map avec Hammer !" (OpenClassrooms), updated for J.A.C.K. and the VHLT compilers. Read it in tab order: Part 1 (basic functions), J.A.C.K. Reference, Part 2 (entities and effects), Part 3 (compilation and optimisation), Part 4 (appendices).

Part 1: J.A.C.K. basics

Get started with J.A.C.K. and learn to handle brushes. This is where beginners should start.

1. Installation and setup

To map for Counter-Strike 1.6 you need three things: the J.A.C.K. editor, compilers (VHLT), and two kinds of files that ship with the game or the editor, the FGD and the WADs. You will install them, then tie them together in a **game profile**.



setup · 3 file sources, 1 profile, 1 compiled map

The game profile is the hub of everything: it tells J.A.C.K. which entities to offer (FGD), which textures to show (WAD), and which programs to run to build the playable map.

Download and install

1. **J.A.C.K.** comes as a free version (official site jack.hlfx.ru) and a paid version on Steam ($14.99). It runs on Windows, Linux and macOS.
2. **VHLT** (Vluzacn's Half-Life Tools): version v34, released in August 2015, is the latest (VDC: VHLT). Download it from GameBanana and unzip it to a simple folder such as `C:\VHLT`. It contains `zhlt.wad`, required for the tool textures.
3. **The FGD**: J.A.C.K. ships with updated FGDs (VDC: J.A.C.K.). For Counter-Strike, use `halflife-cs.fgd`, or its improved version Counter-Strike.fgd.
4. **The WADs** are already in the game, in the `valve` and `cstrike` folders.

> **Warning:** the Steam version of J.A.C.K. does not include the GoldSrc compilers, and the free version includes an old version. Either way, install VHLT separately (VDC: J.A.C.K.).

> **Note:** with Steam, the game is usually in `C:\Program Files (x86)\Steam\steamapps\common\Half-Life`. The `cstrike` folder is Counter-Strike's, the `valve` folder is Half-Life's.

Configure J.A.C.K.

Open **Tools > Options**, then the **Game Profiles** tab ("Game Configurations" in Hammer). The steps follow the TWHL setup guide.

Create the profile

1. Click **Edit**, then **Add**.
2. Type `CS 1.6` and confirm.

"Game" sub-tab

1. In **Game Data files**, click **Add** and pick the Counter-Strike FGD.
2. **Default PointEntity class**: `info_player_start`. **Default SolidEntity class**: `func_wall`. These are the entities offered by default, which saves time.
3. Ignore "Texture format" and "Map type".

"Directories" sub-tab

- **Mod Directory**: `Half-Life\cstrike`.
- **Base Game Directory**: `Half-Life\valve`.

"Build Programs" sub-tab

- **Game Executable**: `Half-Life\hl.exe`. Add `-game cstrike` after the name so the game starts straight into Counter-Strike.
- **CSG, BSP, VIS, RAD**: the four VHLT programs (`hlcsg.exe`, `hlbsp.exe`, `hlvis.exe`, `hlrad.exe`, or their 64-bit versions).
- **Source Maps Directory**: `Half-Life\cstrike\maps`. This is where the compiled map is copied.

> **Warning:** if the compilers and the maps folder are on two different drives, compiling can fail. Keep them on the same drive (TWHL).

"Textures" sub-tab

Click **Add WAD** and add:

- `halflife.wad` and `liquids.wad` (`valve` folder), useful for water textures and some tool textures;
- the `cstrike` WADs you need (`cstrike.wad`, `cs_dust.wad`…), **except** `cached.wad`;
- `zhlt.wad`, shipped with VHLT.

> **Note:** unlike Hammer, J.A.C.K. keeps a separate WAD list for each profile. You can therefore have a Half-Life profile and a CS profile without reconfiguring everything each time.

Check that everything works

1. Confirm with **OK**, then **File > New** and choose the `CS 1.6` profile.
2. Open the texture browser (**Browse** in the right panel): your WAD textures should appear.
3. In the entity list, look for `armoury_entity`: if it is there, the Counter-Strike FGD is loaded.

Sources: VDC: J.A.C.K., TWHL: Setup Guide, VDC: VHLT, Steam: J.A.C.K..

2. The basics of any map

Before building anything, you need to understand three things: how the engine sees your map, how the editor shows it to you, and what it is made of.

The GoldSrc engine

Counter-Strike 1.6 runs on the **GoldSrc** engine, the Half-Life engine. It is a BSP engine: the map is split into convex volumes, which lets the game work out very quickly what is visible. This imposes a few rules you will meet throughout the course.

- **Maximum size**: 8192 × 8192 × 8192 units, i.e. 4096 units in each direction from the centre (VDC: GoldSrc Dimensions). J.A.C.K. accepts much larger maps, but the game will not load them.
- **The unit**: everything is measured in editor units. One unit is roughly one inch (2.54 cm). This is only an approximation: mainly remember that the player is 72 units tall.

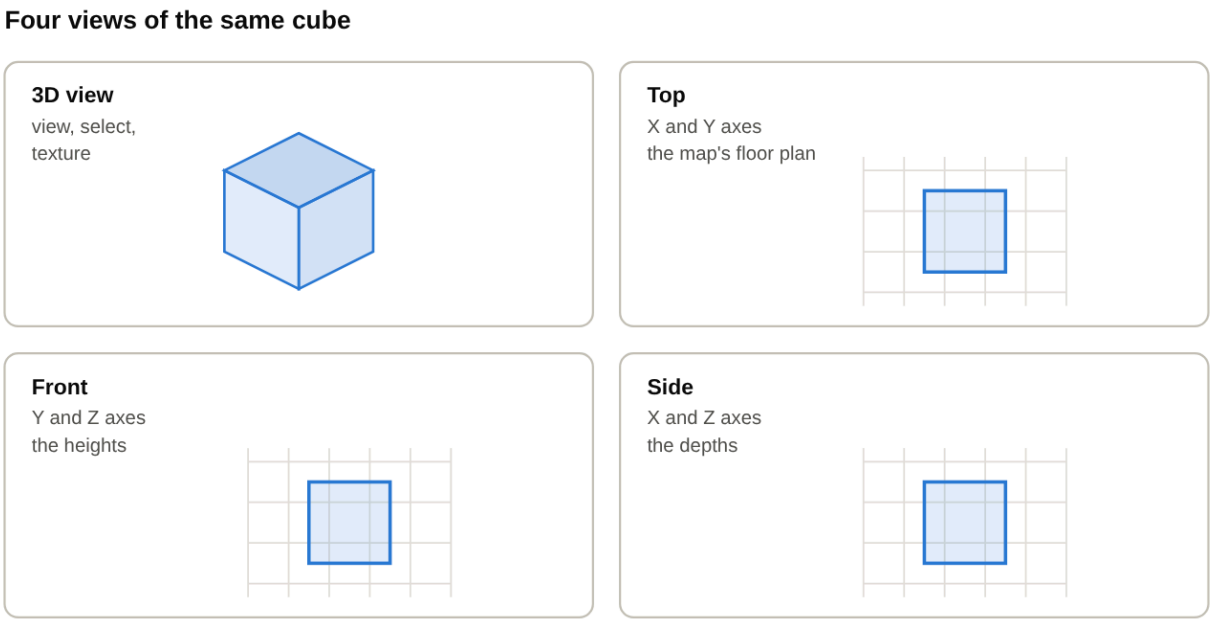
> **Note:** the useful dimensions (step, jump and passage heights) are collected in the "Lengths" appendix, Part 4.

The work environment

The J.A.C.K. screen is made of:

- **four views** in the centre (detailed below);
- **the left toolbar**: selection, brush and entity creation, texturing, clipping, vertices;
- **the top bar**: file, undo, grid, texture lock, compile;
- **the right panel**: current texture, brush shape, entity to place, **To Entity** and **To World** buttons, visibility groups;
- **the status bar** at the bottom: cursor position, selection size and, added by J.A.C.K., the position of the selection centre (VDC: J.A.C.K.).

The 2D views and the 3D view



the 4 J.A.C.K. views · one and the same cube

The three 2D views show the map flat, each along two axes. This is where you draw and move brushes, pixel-precise thanks to the grid. The 3D view is used to check the result, select faces and texture.

> **Note:** right-click a view's name (top left of each view) to change its type. For the 3D view, choose **3D Textured** to see textures, **3D Flat** for solid colours, or **Wireframe** for edges only.

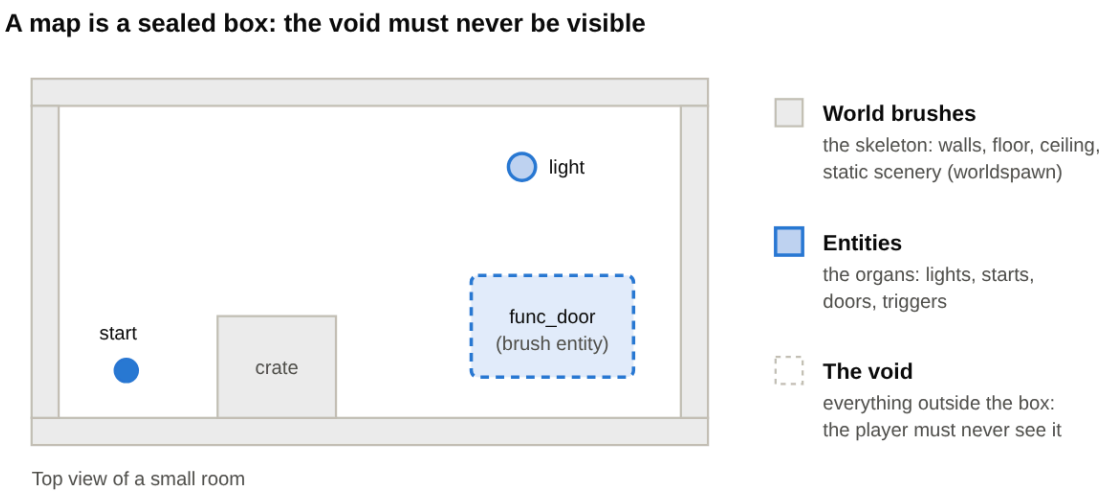
J.A.C.K. adds features to the views that Hammer lacks: a background image in the 2D views (handy for tracing a plan), cameras you can save and recall, and real-time display of animated and transparent textures and of the sky (VDC: J.A.C.K.).

Moving around

- **In a 2D view**: the mouse wheel zooms; hold the space bar + left-click to pan.
- **In the 3D view**: press **Z** for free-look; the mouse aims and W, A, S, D (or the arrow keys) move. Press **Z** again to get the cursor back.

> **Common mistake:** in free-look the mouse no longer selects anything. If no click responds in the 3D view, press Z.

What a map is made of



parts of a map · brushes, entities, void

A map is made of two families of objects:

- **Brushes**: convex volumes that form the skeleton of the map. As long as they are not tied to an entity they belong to the "world" (worldspawn): they are fixed, solid and cast shadows.
- **Entities**: anything that has behaviour. Some are points (a light, a player start), others are brushes given a role (a door, a buy zone).

And there is what is nothing: **the void**. Everything outside the brushes is void, and the player must never be able to see it. Your map must therefore be a fully sealed box. If a hole remains, compilation detects a "leak" and the map will be badly lit and slow; this is covered in "Compilation (B: debugging)", Part 3.

> **Warning:** the box must be sealed by world brushes. A door, a breakable wall or a window (entities) do not seal the void.

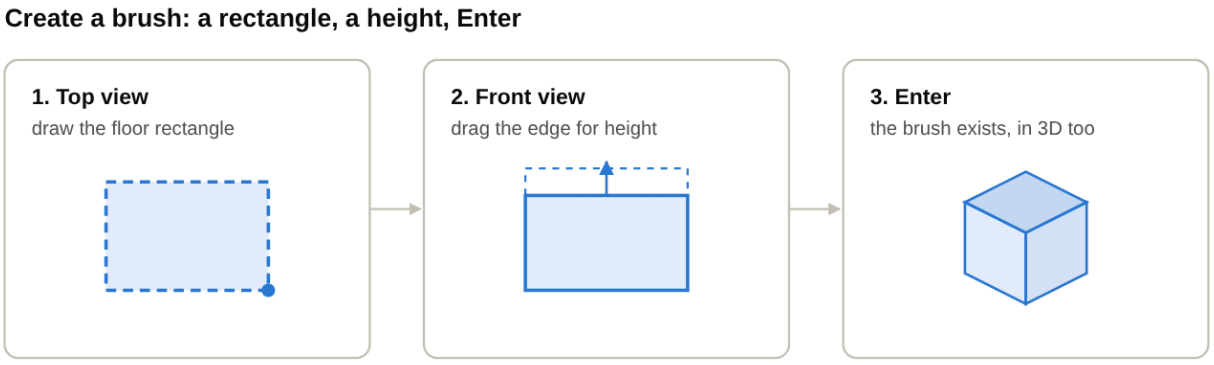
Sources: VDC: GoldSrc Dimensions, VDC: J.A.C.K., VDC: Hammer Hotkey Reference.

3. Brushes (A: creation)

Brushes are the skeleton of your map. This chapter teaches you to create, select and duplicate them, without worrying about making a playable map yet.

Create a brush

Open a blank map with **File > New** and choose your `CS 1.6` profile. The four views are empty: that is normal.



creating a brush · 3 steps

1. Pick the **Block** tool in the left bar (Shift+B). The cursor becomes a cross.
2. In the **Top** view, click and drag to draw a 3 × 3 square rectangle.
3. In the **Front** view, drag the top edge to give the brush a height of 3 squares.
4. Check in the **Side** view that the shape is right.
5. With the cursor still in a 2D view, press **Enter**.

> **Common mistake:** if you press Enter while the mouse is in the 3D view, the brush is not created. Go back to a 2D view.

To cancel before confirming, press **Esc**. To delete a brush you already created, select it and press **Del**.

Select a brush

Take the **Selection** tool (Shift+S).

- **In the 3D view**, click the brush.
- **In a 2D view**, click one of its edges or the small cross at its centre. Anywhere else selects nothing.
- **Several brushes**: hold **Ctrl** while clicking.
- **A whole area**: in a 2D view, drag a box around the brushes, then press **Enter**.
- **Overlapping objects**: **Page Up / Page Down** cycle through the objects under the cursor (VDC: Hotkeys).

> **Note:** J.A.C.K. can also select by dragging in the 3D view (VDC: J.A.C.K.).

Cloning

Two ways to duplicate a brush:

- **Ctrl+C** then **Ctrl+V**, as everywhere;
- faster: hold **Shift** while dragging the brush with the mouse. One copy stays in place, the other follows the cursor.

J.A.C.K. keeps the links between entities when you clone or copy a group: if a door and its button are copied together, the copied button opens the copied door (VDC: J.A.C.K.).

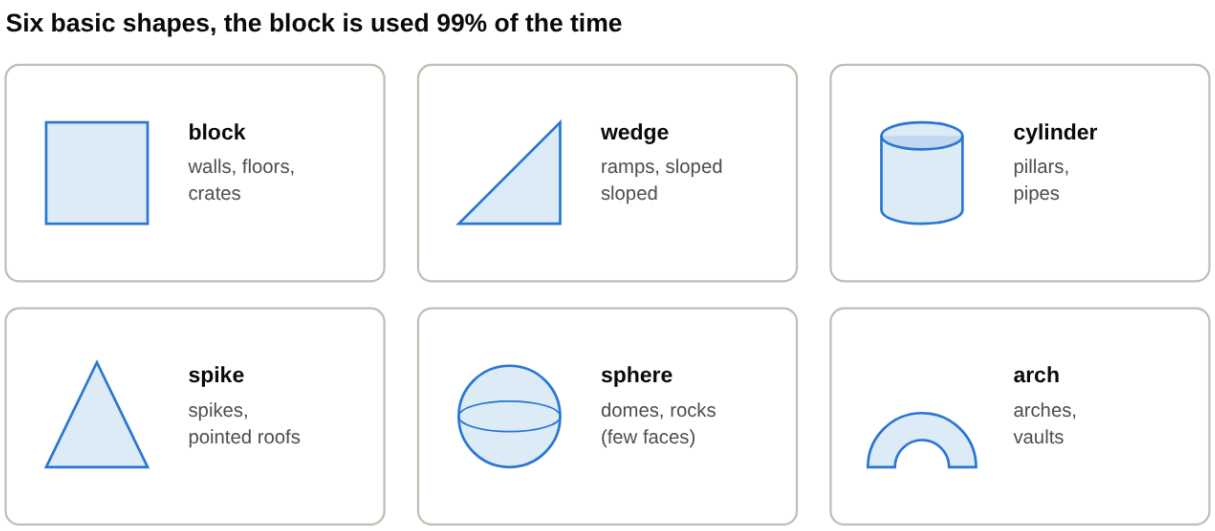
Zoom and grid precision

The grid acts as a magnet: everything you draw snaps to its lines.

- **[** makes the grid finer (16, 8, 4, 2, 1 units), **]** makes it coarser.
- The status bar shows the current step, e.g. "Snap: On Grid: 16".
- The **mouse wheel** zooms. Zooming does not change the grid step: they are two independent settings.

> **Tip:** generally work with a grid of 8 or 16, and go down to 1 only for small details. Off-grid brushes are the leading cause of micro-gaps and leaks.

The different brush shapes



brush shapes · 6 primitives

The shape is chosen in the right panel, under **Objects** (or "Primitives" depending on the version). For cylinders, spheres and arches, the **Faces** field sets the number of sides.

> **Warning:** the more faces a brush has, the more the engine must draw. An 8-sided pillar is usually enough; go up to 12 or 16 only for a large, prominent cylinder.

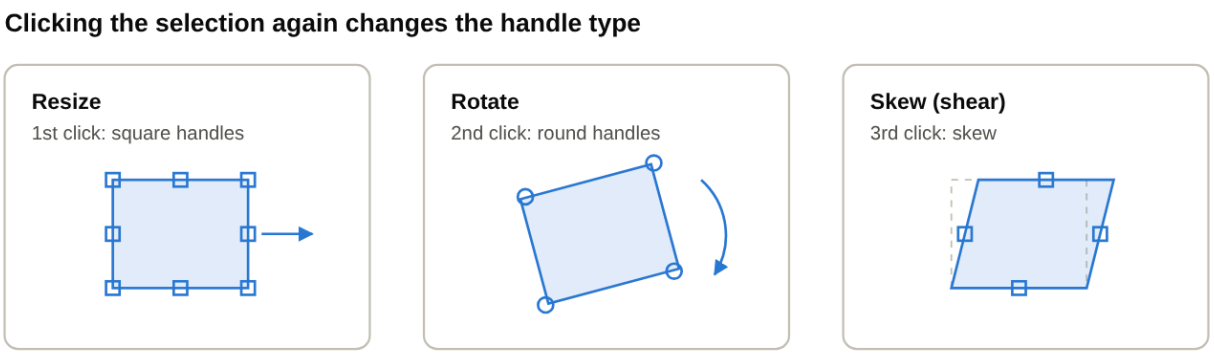
Sources: VDC: Hammer Hotkey Reference, VDC: J.A.C.K..

4. Brushes (B: changing the structure)

You can create brushes. Time to transform them: rotate, stretch, cut, hollow and move their vertices.

Selection handles

A selected brush shows handles in the 2D views. **Click the brush again** to switch from one handle type to the next.



Selection tool · 3 handle types

- **Resize**: drag a square handle. Corner handles act on two axes at once.
- **Rotate**: drag a round handle. Rotation snaps in 15° steps; hold **Shift** to rotate freely (VDC: Hotkeys).
- **Skew**: drag a handle in the middle of an edge; the brush deforms into a parallelogram.

> **Note:** for exact values (rotate exactly 45°, scale to 150%), use **Tools > Transform** (Ctrl+M).

> **Warning:** after a free rotation, a brush's vertices rarely land on the grid. Remember **Snap to Grid** (Ctrl+B) to realign them, otherwise you create tiny gaps between brushes.

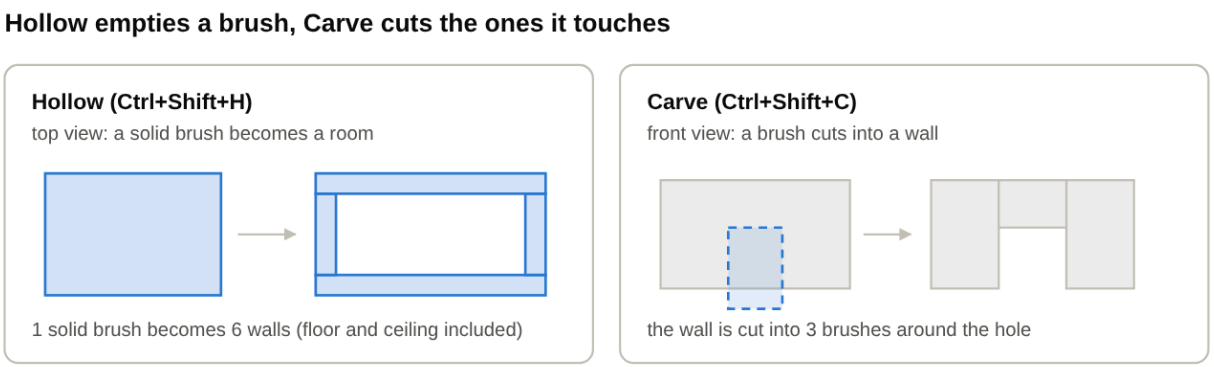
Move precisely

With a brush selected, the **arrow keys** in a 2D view move it by one grid step. This is the best way to butt two brushes together without overlap.

Group and ungroup

To handle several brushes as one object (a table, a staircase), select them and press **Ctrl+G**. **Ctrl+U** separates them. A group exists only in the editor: it has no effect in game.

Hollow and Carve



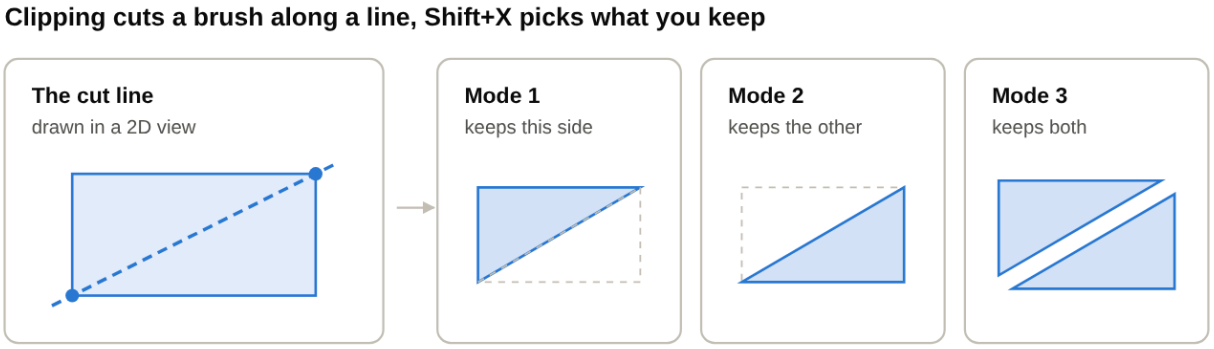
Hollow and Carve · before and after

Hollow (Ctrl+Shift+H in J.A.C.K., Ctrl+H in Hammer) turns a solid brush into a hollow box, with a wall thickness you choose. It is the quickest way to make a sealed room: create a brush the size of the room, then Hollow with a thickness of 16.

Carve (Ctrl+Shift+C) uses the selected brush as a cookie cutter: every brush it touches is cut around it. Then just delete the cutter brush.

> **Warning:** Carve cuts into many small, often badly formed brushes, especially with a cylinder or a sphere. Only use it with simple brushes, and prefer Clipping or building by hand for a door or a window.

Clipping



Clipping tool · 3 modes

The **Clipping** tool (Shift+X) cuts a brush cleanly in two along a plane.

1. Select the brush.
2. Take the Clipping tool.
3. In a 2D view, click and drag to draw the cut line.
4. Press **Shift+X** to cycle the modes: keep one side, keep the other, keep both.
5. Press **Enter** to confirm.

Example: for a ramp, create a 256 × 64 × 64 brush and, in the Front view, draw a diagonal from the bottom-left corner to the top-right corner. Keep the bottom piece: you get a perfect ramp without touching the Vertex tool.

Vertex Manipulation (morphing)

The **Vertex** tool (Shift+V) shows the vertices of the selected brush. Drag them to get shapes no primitive offers: rocks, twisted ramps, spikes.

- The solid dots are the **vertices**.
- The small dots in the middle of edges move **a whole edge**.
- If two vertices overlap, the editor offers to merge them.

> **Common mistake:** a brush must always stay **convex** (no dents) and each face must stay **planar**. If you move a single vertex of a square face, the face twists and the

> brush becomes invalid. J.A.C.K. can then **triangulate** the face to split it into two flat triangles (VDC: J.A.C.K.).

> **Note:** J.A.C.K. can lock the texture axes while you move vertices, so the texture does not slide (VDC: J.A.C.K.).

Sources: VDC: Hammer Hotkey Reference, VDC: J.A.C.K..

5. Textures (A: using textures)

A brush with no chosen texture receives the current texture. This chapter shows how to choose a texture, apply it, align it, and which textures have a special role.

Choose a texture

In the right panel, click **Browse**. The browser shows every texture in your profile's WADs.

- The **Filter** field shows only textures whose name contains what you type (`door`, `crate`, `wall`...).
- The **Only used textures** box shows only textures already in the map.
- J.A.C.K. sorts textures into **tabs** (VDC: J.A.C.K.).

Double-click a texture: it becomes the current texture.

Apply a texture

Before creating the brush: choose the texture, then create the brush. All its faces get it.

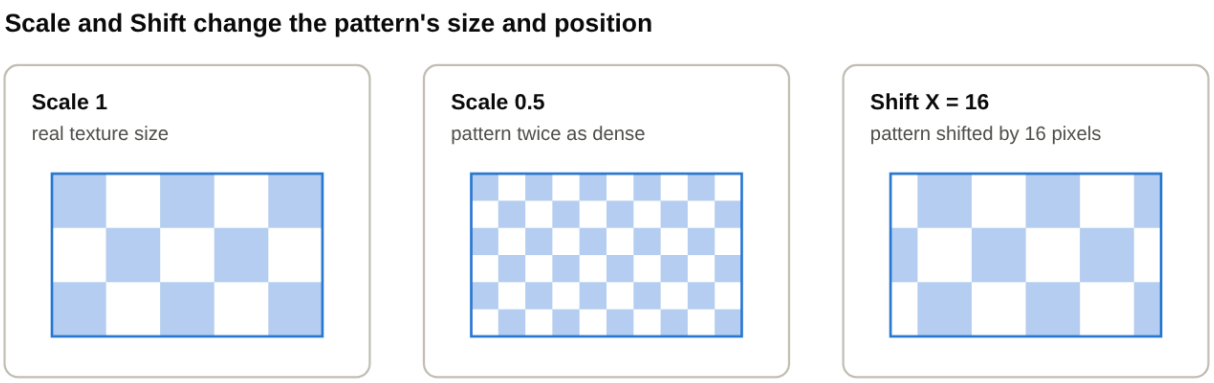
After creation:

- the whole brush: select it and press **Shift+T** (apply current texture);
- a single face: take the **Texture Application** tool (Shift+A), click the face in the 3D view, then choose the texture.

With the Texture Application tool, in the 3D view (VDC: Hotkeys):

- **left-click** selects a face, **Ctrl+click** adds more, **Shift+click** takes all faces of the brush;
- **right-click** applies the current texture to the face under the cursor;
- **Alt+right-click** applies the texture and alignment of the selected face.

Align a texture



Texture Application window · effect of Scale and Shift

The Texture Application window sets how the texture sits on the selected faces:

- **Scale X / Y**: the size. 1 = real size, 0.5 = half as big (so more repeats), 2 = twice as big.
- **Shift X / Y**: the offset in pixels, to line a pattern up with an edge.
- **Rotation**: the angle of the texture on the face.
- **Justify**: aligns the texture left, right, top, bottom, centre, or **Fit** to stretch it exactly over the face. J.A.C.K. can fit several faces at once.
- **World / Face**: in World mode the texture follows the world axes, which keeps it continuous from brush to brush; in Face mode it follows the face, which suits slopes better.
- **Treat as one**: several selected faces are treated as one for Justify.

Example: a 64 × 112-unit door with a 64 × 128-pixel door texture. Select the front face and click **Fit**: the scale becomes 1 × 0.875 and the door fits the brush exactly.

> **Note:** the **Texture Lock** button in the top bar makes the texture follow when you move a brush. When off, the texture stays fixed in the world and "slides" over the brush.

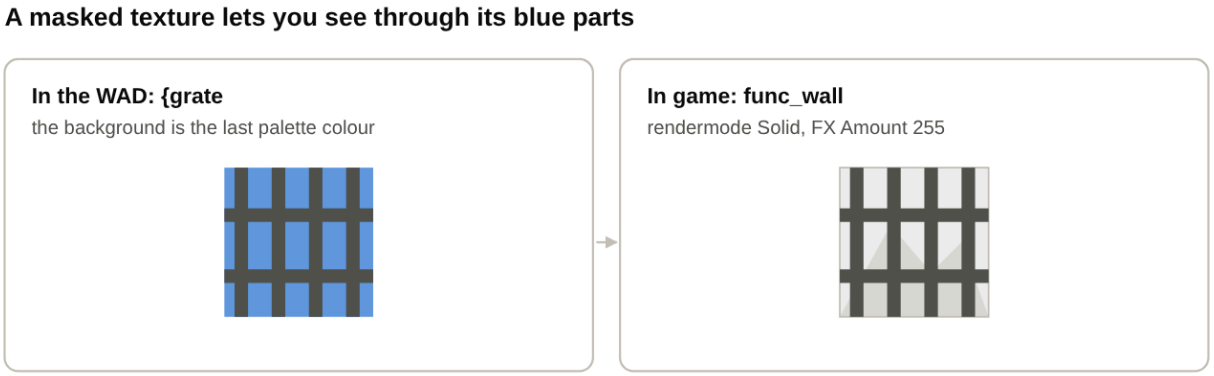
> **Note: Replace** (Tools menu) replaces one texture with another across the whole map; J.A.C.K. can adjust the scale if the two textures differ in size (VDC: J.A.C.K.).

Special texture names

The start of a texture's name changes its behaviour in game (VDC: Texture prefixes).

Prefix	Effect	Typical use
`{`	Masked texture: the last palette colour (usually pure blue) becomes transparent, if the entity has a render mode other than Normal	Grates, fences, foliage
`!`	Liquid: ripples, you can swim in it	Water
`!slime`, `!lava`	Harmful liquid (lava: double damage)	Toxic pool

Prefix	Effect	Typical use
`!cur_0`, `!cur_90`…	Current that pushes in one direction	River
`+0` to `+9`	Animation of up to 10 frames, about 10 frames per second	Screens, blinking neon
`+A` to `+J`	Second set: a button or func_wall toggles between `+0` and `+A` when triggered	Button that lights up
`-0` to `-9`	Random variant picked for each face	Floors that do not repeat
`scroll`	Scrolls, at the speed of a func_conveyor	Conveyor belt
`sky`	Shows the sky (skybox)	Outdoor ceiling



masked texture · in the WAD and in game

For a masked texture to be transparent, put it on an entity (a `func_wall` or a `func_illusionary`) and set **Render Mode = Solid** and **FX Amount = 255** (TWHL: Render Properties). On a world brush the blue stays visible.

Tool textures

Some textures are never displayed: they give instructions to the compiler. They are in `zhlt.wad` and `halflife.wad` (TWHL: Tool Textures).

Texture	Visible	Solid	Role
NULL	No	Yes	Removes the face from rendering. For faces the player never sees
CLIP	No	Players	Invisible wall: blocks players, lets bullets through
ORIGIN	No	No	Sets the rotation axis of a rotating entity
AAATRIGGER	No	No	Conventional texture for triggers
SKY	Sky	Yes	Shows the map's skybox
HINT + SKIP	No	No	Visibility optimisation (see Part 3)
BEVEL	No	Yes	Fixes snagging on outer corners

Texture	Visible	Solid	Role
NOCLIP	No	No	Like NULL, with no collision at all

> **Warning:** do not mix a tool texture with normal textures on the same brush, nor two tool textures together. The exceptions are NULL (with normal textures) and the HINT + SKIP pair (VDC: Tool textures).

Example: a staircase with small steps makes the player's view bob. Lay a ramp-shaped CLIP brush over it from the first to the last step: the player glides smoothly, but the steps stay visible and bullets still hit them.

Decals

Decals are images laid over walls: marks, posters, stains, graffiti. They come from `decals.wad`.

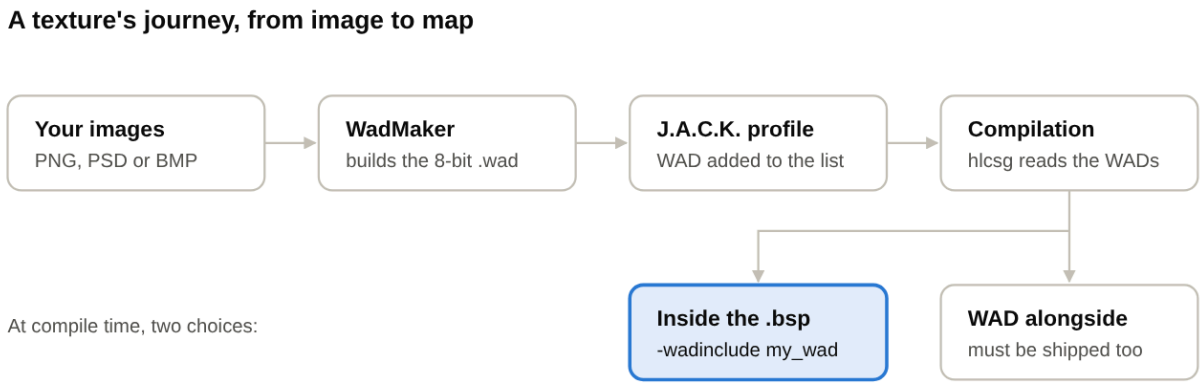
1. Choose a decal in the texture browser.
2. Take the **Decal** tool (Shift+D).
3. Click a wall in the 3D view: J.A.C.K. creates an `infodecal` entity there.

> **Note:** J.A.C.K. displays decals more faithfully than Hammer (VDC: J.A.C.K.). A decal must touch a world surface: placed in empty space, it does not show.

Sources: VDC: Texture prefixes, TWHL: Tool Textures, VDC: Tool textures (GoldSrc), TWHL: Render Properties, VDC: Hammer Hotkey Reference, VDC: J.A.C.K..

6. Textures (B: making your own)

The game's textures are no longer enough? You can create your own, store them in your own WAD file and use them in J.A.C.K. like any other.



custom texture · from image to .bsp

The rules of a GoldSrc texture

The engine is demanding (the303: GoldSrc Map Texture Tutorial):

- **Dimensions**: multiples of 16 (64 × 64, 128 × 128, 256 × 64…). Powers of 2 are the most practical.
- **Maximum size**: 512 × 512. Use it sparingly: players have to download your textures.
- **Colours**: 8-bit, i.e. a 256-colour palette per texture.
- **Name**: 15 characters at most, no spaces.

The tools

Tool	What it does	For whom
WadMaker	Turns a folder of images into a WAD. Builds the 256-colour palette by itself	Recommended: fast, accepts PNG, Photoshop and Krita
Wally	The classic WAD editor: view, rename, retouch	For managing a WAD by hand
HL Texture Tools	WAD and sprite viewer and editor	Alternative to Wally

> **Note:** WadMaker is a command-line tool for Windows and Linux. It can also extract textures from an existing WAD or .bsp (VDC: WadMaker).

Build your WAD with WadMaker

1. Create a folder, e.g. `my_wad`, and put your images in it. **Each file name becomes the texture name**: follow the prefixes from the previous chapter (`{grate.png`, `+0screen.png`, `+1screen.png`…).
2. Drag the folder onto `WadMaker.exe`. A `my_wad.wad` file is created next to it.
3. If you change an image, do it again: the WAD is rebuilt.
4. For special settings (dithering, transparency colour), a `wadmaker.config` text file in the folder lets you set them texture by texture.

Masked textures

A texture whose name starts with `{` uses the **last colour of its palette** (index 255) as the transparent colour (VDC: Texture prefixes). By convention, these areas are painted pure blue (0, 0, 255). With WadMaker, transparent areas of a PNG are handled automatically.

> **Reminder:** in game, transparency only appears on an entity with **Render Mode = Solid** and **FX Amount = 255** (chapter 5).

Use your WAD

1. In **Tools > Options > Game Profiles > Textures**, click **Add WAD** and choose `my_wad.wad`.
2. Your textures appear in the browser like the others.
3. When compiling, decide how the game will find them:

- **embedded in the .bsp** with hlcsg's `-wadinclude my_wad` parameter: the map is bigger, but works on its own;
- **WAD alongside**: you must ship `my_wad.wad` with the map, in the `cstrike` folder, or players will get missing textures.

> **Tip:** for a map meant for servers, embed your custom textures in the .bsp. Do not do it for the game's WADs (`cstrike.wad`, `halflife.wad`): every player already has them.

Texture sounds

Footstep and bullet-impact sounds depend on the texture. The game reads them from `sound/materials.txt`, one line per texture: a letter then the name (TWHL: materials.txt).

Letter	Material	Letter	Material
M	Metal	T	Tile
V	Vent duct	G	Grate
D	Dirt	W	Wood
S	Liquid	P	Computer
Y	Glass	N	Snow (CS only)
X	Grass (CS only)		

Example: the line `W CRATE01` gives the `CRATE01` texture a wood sound.

> **Warning:** Counter-Strike reads the full name for footsteps, but only the first 12 characters for bullet impacts (TWHL). And `materials.txt` is shared by the whole game: changing it on a server changes the sounds for every map. For a custom map, the simplest approach is to name your textures like existing textures of the right material, or accept the default sound (concrete).

Sources: the303: GoldSrc Map Texture Tutorial, VDC: WadMaker and SpriteMaker, VDC: Texture prefixes, TWHL: Sound Effects for Textures.

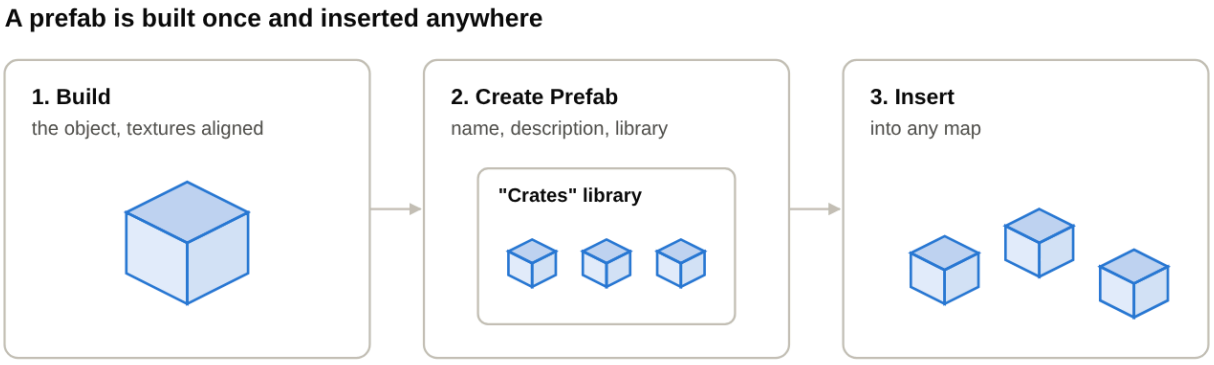
7. Prefabs

After creating enough brushes you will start repeating yourself: a crate, a street lamp, a table come back in every map. **Prefabs** (prefabricated objects) save you from rebuilding them.

> **Question:** what is a prefab?

A prefab is a group of objects (brushes and entities) saved in a library, which you insert in one click into any map.

- **Pro**: complex objects in seconds, always identical.
- **Con**: a map made of downloaded prefabs loses all originality. Use them to save time, not to replace your own work.



prefabs · build, save, insert

Insert a prefab

J.A.C.K. uses Hammer's prefab system.

1. In the right panel, where you choose the brush shape, open the **categories** list: besides "Primitives" it contains the prefab libraries.
2. Choose a library, then an object in the list below.
3. Click **Insert original prefab**: the object appears in the map as a group.
4. Move and rotate it like any group.

> **Note:** a prefab's name sometimes indicates its orientation ("east", "north"…). You can rotate it after inserting anyway.

Create a prefab

Example: an open crate the player can hide in.

1. Create a 64 × 64 × 64 brush with a crate texture, and align the texture.
2. **Hollow** it (Ctrl+Shift+H) with a thickness of 4.
3. Turn on **Ignore Groups** in the top bar, then select and delete the front wall.
4. Align the textures inside.
5. Select the five remaining brushes and click **Create Prefab** in the right panel.
6. Give a name and a description, choose the library, then **OK**.

The prefab library

The **Tools > Prefab Factory** menu opens the prefab manager: create a library (Add Library), add, rename, delete or export objects.

Download prefabs

GoldSrc community sites (TWHL, GameBanana) offer prefabs and example maps you can copy objects from. Another very simple method: open an example map in J.A.C.K., copy the object (Ctrl+C) and paste it into your map (Ctrl+V).

> **Common mistake:** a downloaded prefab may contain invalid brushes, textures missing from your WADs or Half-Life-only entities. After inserting it, run **Map > Check**

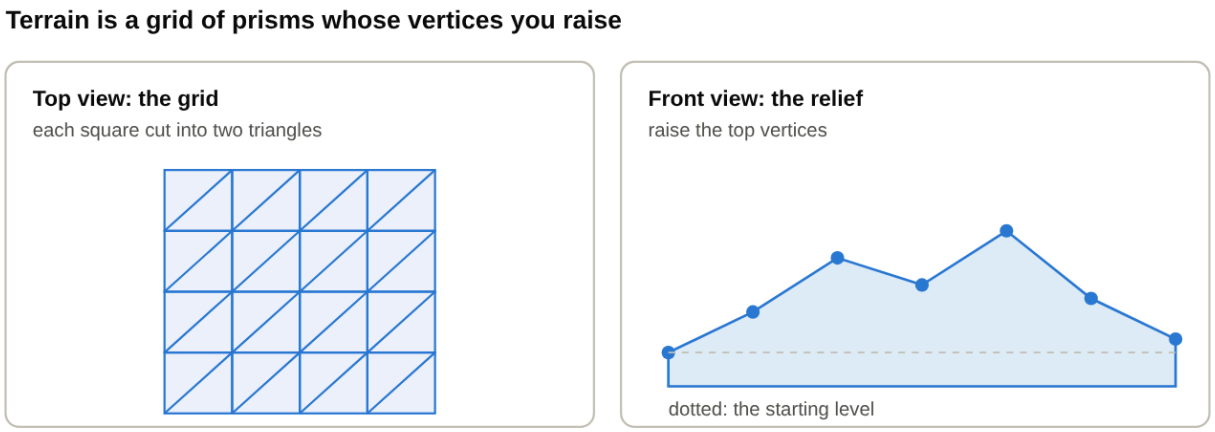
> **for Problems** (Alt+P) before compiling.

Source: the original tutorial "Une map avec Hammer !" (OpenClassrooms, 2011), chapter "Les prefabs"; J.A.C.K. uses the same method.

8. Natural scenery: shaping terrain

Hills, dunes, river banks: natural terrain is not made of straight brushes. GoldSrc has no dedicated terrain system; you build it from brushes whose vertices you move.

> **Note:** the old generators (GenSurf, Terrain Generator) mentioned in old tutorials are no longer maintained. With J.A.C.K., the Vertex tool is enough for most terrain. And importing Source-format maps (VMF) does not carry over "displacements", which do not exist in GoldSrc (ModDB: J.A.C.K. 1.1.1058).



terrain · grid seen from above and relief seen from the front

Prepare the grid

1. Create a brush 128 × 128 units and 32 high: this is one terrain "tile".
2. With the **Clipping** tool (Shift+X), cut it diagonally in the Top view, in "keep both" mode. You get two triangular prisms.
3. Select both prisms and clone them (Shift + drag) to fill a 4 × 4 grid of tiles, with no gaps between them.

> **Why triangles?** A face with three vertices is always planar. If you raise one vertex of a square tile, its top face twists and the brush becomes invalid. With triangles you can raise any vertex.

Shape the relief

1. Select **the whole grid**, then take the **Vertex** tool (Shift+V).
2. In the Front or Side view, select a top vertex (drag a selection box around it): the overlapping vertices of neighbouring tiles are taken together, so the surface stays joined.

3. Raise or lower it with the mouse or the arrow keys.
4. Repeat vertex by vertex to draw hills and hollows.

> **Warning:** keep the grid on (8 or 16) while shaping. Off-grid vertices create cracks between tiles, and therefore leaks.

Texture the terrain

Select all the top faces with the Texture Application tool (Ctrl+click), apply a grass, sand or dirt texture, then **Align: World** so the pattern runs continuously from tile to tile. Put the **NULL** texture on the bottom faces, which the player will never see.

> **Tip:** terrain quickly gets expensive in polygons. Prefer a few large tiles (128 or 256) to small 32 tiles: the difference in relief is barely visible, the difference in smoothness very much so (see "Optimisation", Part 3).

Sources: VDC: J.A.C.K., ModDB: J.A.C.K. 1.1.1058, TWHL: Tool Textures.

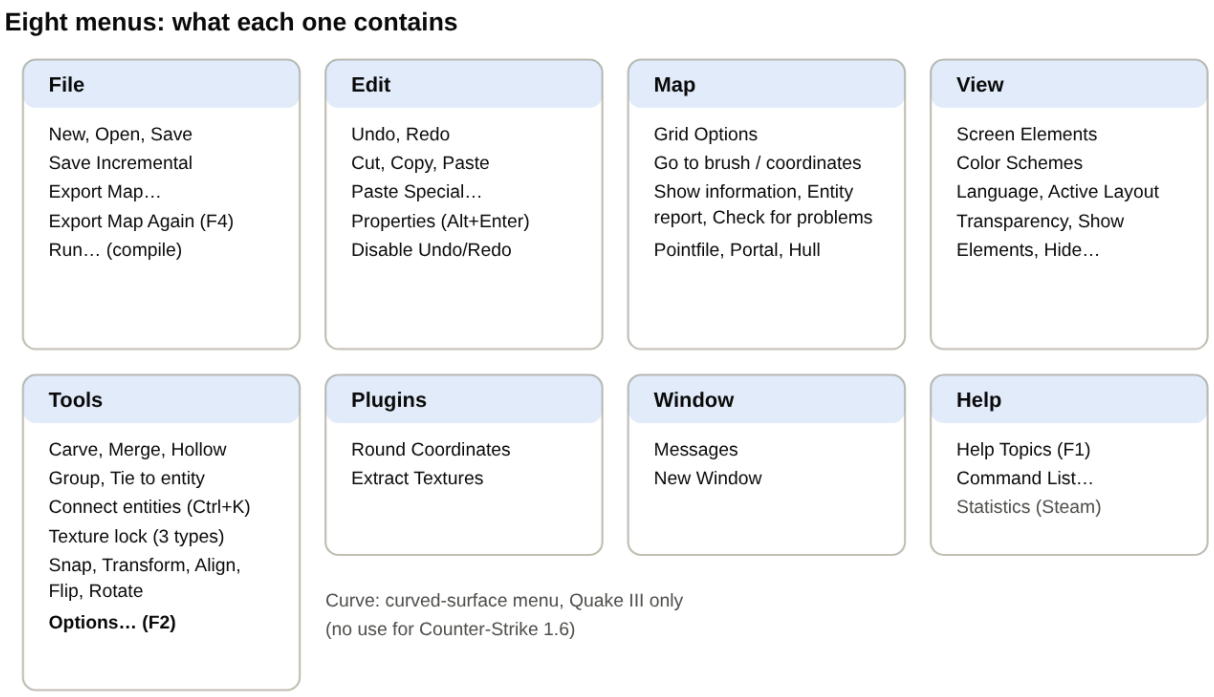
J.A.C.K. Reference: menus, toolbars and options

The catalogue of the editor's commands and settings, based on its official manual.

This tab complements Part 1. It is based on the **J.A.C.K. Reference Manual** (version 1.1) published by the editor's authors (official site, 2016; Steam version, 2018) and on the features page of the official site. Command names are as they appear in the editor; J.A.C.K. is also available in Russian, Ukrainian, Belarusian and Polish (View > Language).

> **Note:** some parts of the manual (toolbars and the full shortcut list) could not be read online. What is given here is confirmed; the full shortcut list is in the editor, under **Help > Command List…**.

1. The menus



menu bar · 8 menus and their main commands

The descriptions below follow chapter 5.1 of the manual. Commands common to all software (New, Open, Save, Undo, Copy…) are not detailed.

File

Command	Shortcut	Role
Save Incremental		Saves, adding a version number to the file name

Command	Shortcut	Role
Export Map…		Exports the map as .map, the only format the compilers read (they do not read .jmf)
Export Map Again	F4	Re-exports to the same place and name as last time
Run…		Opens the compile window, then launches the game on the map

Edit

Command	Shortcut	Role
Properties	Alt+Enter	Properties of the world or of the selected entity
Paste Special…		Multiple paste with offset and rotation (section 4)
Disable / Enable Undo/Redo		Clears the undo history to free memory

Map

Command	Shortcut	Role
Grid Options > Snap to Grid		Snap to grid
Grid Options > Show Grid		Shows the grid in the 2D views
Grid Options > Show 3D Grid		Shows a grid on brushes in the 3D view
Grid Options > Grid Settings		Grid size (faster with [and])
Go to brush number…	Ctrl+Shift+G	Jumps to a brush or entity by its number
Go to coordinates…		Places the camera at "X Y Z" coordinates
Show information…		Map statistics
Entity report…		Entity list and filter (section 4)
Check for problems	Alt+P	Finds and fixes errors (section 4)
Map properties…		Worldspawn properties (sky, WADs…)
Load / Unload Pointfile		Shows a leak trace (.pts or .lin)
Load / Unload Portal file		Shows the VIS portals (.prt)
Load / Unload Hull file		Shows a collision hull (.p0 to .p3), for collision problems

View

Command	Role
Screen Elements	Shows or hides the toolbars
Color Schemes	Colour schemes for the interface and views, preset or custom (see section 3, "Colours")

Command	Role
Language	Interface language, without restarting
Active Layout	Type of the active view: X/Y, Y/Z, X/Z, or wireframe, flat, textured, shaded textured rendering
Autosize 4 Views	Gives the four views the same size
Center 2D/3D Views on Selection	Centres all views on the selection
Transparency	Transparency of special textures and/or entities
Show Elements	target/targetname links, wireframe models in 2D views, realistic sky, texture animation, special-texture faces, UV grid
Hide Items / Hide Paths	Hides item entities and path entities
Hide Unknown Entities	Hides entities missing from the FGD
Hide Entity Names	Hides class and name labels
Hide Selected / Unselected Objects	Hides the selection, or everything else
Show Hidden Objects	Shows everything hidden again

Tools

Command	Shortcut	Role
Carve	Ctrl+Shift+C	Subtracts the selected shape from the surrounding brushes
Merge	Ctrl+Shift+M	Merges into one brushes that together form a convex volume
Make Hollow…	Ctrl+Shift+H	Hollows the brush (thickness to choose)
Group / Ungroup		Binds objects together
Make Detail / Structural		Detail brush, if the game profile allows it
Tie to entity / Move to world		Turns into an entity, or returns to the world
Connect entities	Ctrl+K	Creates the target → targetname link between two selected entities
Texture application…	Shift+A	Surface Properties window
Replace textures…		Find and replace textures
Texture lock (move+rotate)		The texture follows moves and rotations
Texture lock (scale)		The texture follows resizing
Texture lock (UV coordinates)		The texture stays pinned to the vertices (UV-Lock)

Command	Shortcut	Role
Align to view		Projects the texture from the camera, like a slide projector
Snap selected to grid		Snaps one corner of the selection to the grid; repeating changes corner (8 corners)
Snap selected to grid individually		Snaps each object separately, point entities by their origin
Transform		Rotation, scale or move with exact values
Align objects		Aligns objects to one side of the selection
Flip objects		Flips horizontally, vertically or both, depending on the last active 2D view
Rotate objects		Rotates clockwise or counter-clockwise, depending on the last active 2D view
Automatic configuration…	Ctrl+F2	Creates game profiles automatically (Steam version)
Options…	F2	J.A.C.K. configuration (section 3)

Plugins, Window, Help

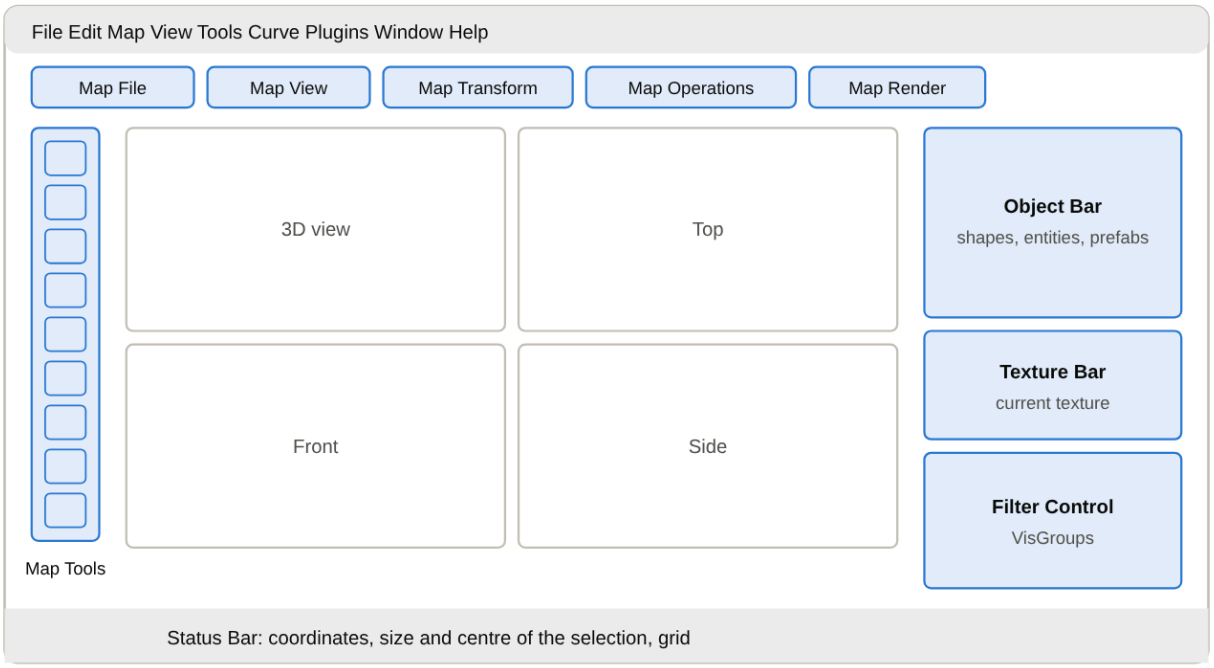
Menu	Command	Role
Plugins	Round Coordinates	Rounds coordinates on export, like Hammer
Plugins	Extract Textures	Extracts textures from a .bsp (mainly useful for Quake)
Window	Messages	Panel of errors, warnings and information, exportable as text
Window	New Window	Extra window (independent windows mode)
Help	Help Topics (F1)	Opens the manual
Help	Command List…	The list of all shortcuts
Help	Statistics…	Steam statistics (Steam version)

Source: J.A.C.K. Reference Manual, chapter 5.1.

2. Toolbars and panels

The manual describes ten bars (chapter 5.3). They can be moved, shown or hidden with **View > Screen Elements**: the diagram shows an indicative layout, not necessarily the one on your screen. **Tools > Options > General > Use large map toolbar icons** enlarges their icons.

The J.A.C.K. screen: where each bar is



J.A.C.K. screen · indicative toolbar layout

Bar	What it contains (per the manual)
Map Tools	The tools from Part 1, chapter 3: selection, magnifier, camera, entity, brush, texture application, decal, clipping (click again to change mode), vertices, and the Path tool
Map File	The file commands
Map View	The display commands, including **Load Window State**, which reloads a saved window layout
Map Transform	The transform commands
Map Operations	**Merge** (Ctrl+Shift+M), **Carve** (Ctrl+Shift+C), and two toggles: immediate box selection in 2D views (no Enter), and box selection by centre handle only
Map Render	The render modes, including **Show/hide models in 2D** (wireframe models in the 2D views)
Object Bar	Brush shapes (Primitives) and their settings, prefab libraries, the list of entities to place, **To Entity** and **To World**
Texture Bar	The current texture and the **Browse** button
Filter Control	VisGroups (details below)
Status Bar	Cursor coordinates, selection size, position of its centre, grid

For Map File, Map View and Map Transform, the button-by-button details could not be read in the online manual: hover over each icon in the editor to see its name.

Filter Control: VisGroups

VisGroups let you hide or show parts of the map while editing. An object can belong to several VisGroups.

Action	Effect
Double-click a name	Hides or shows the group (hand icon)
Assign	Adds the selected objects to the chosen group
Mark	Highlights the group's objects
Edit	Changes the group's name and colour
Remove	Deletes the group, not its objects
Purge	Deletes empty groups
Drag one group onto another	Merges the two, under the second one's name

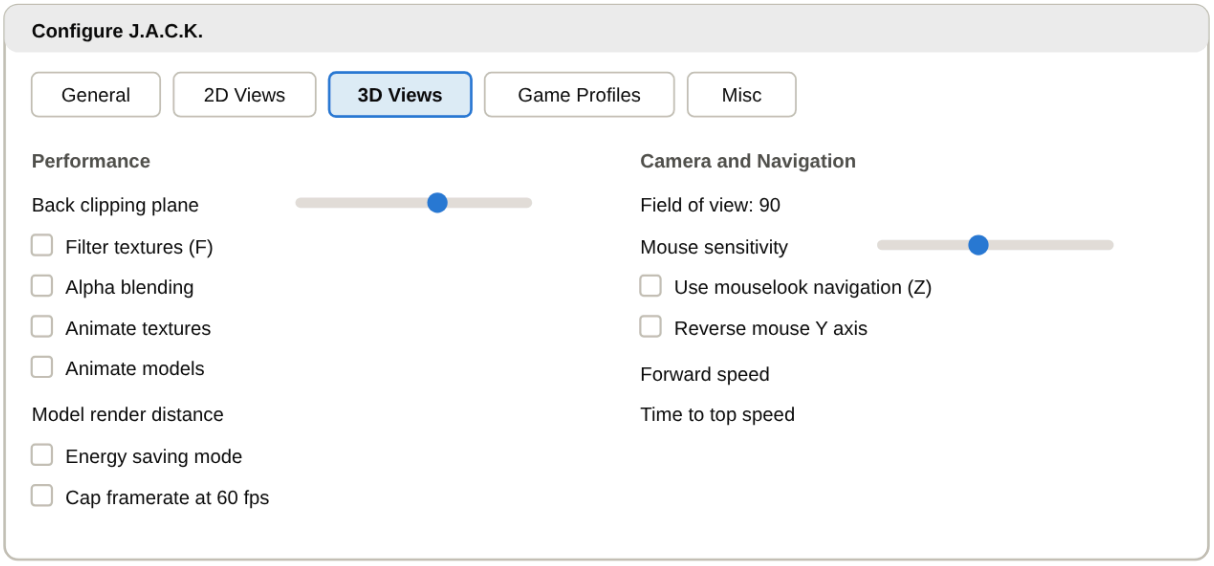
To create a group: select, Alt+Enter, VisGroup tab, **Edit groups…** then **New Group**.

Sources: J.A.C.K. Reference Manual, chapters 3.8, 4.1 and 5.3, VDC: Hammer Hotkey Reference.

3. Options: Tools > Options (F2)

The **Configure J.A.C.K.** window has five tabs: **General**, **2D Views**, **3D Views**, **Game Profiles** and **Misc**. Only Game Profiles is essential at first; the others can stay at their defaults (manual, chapters 2.1 and 5.2).

F2 opens Configure J.A.C.K.: five tabs, here 3D Views



Configure J.A.C.K. · 3D Views tab

General tab

Option	Effect
Independent window configurations	Free window layout; add windows from the Window menu
Load default window positions with maps	Reloads the last layout when opening a map (otherwise: Load Window State button)
Maximum world size (units)	Size of the build space; also a compiler limit (8192 for Half-Life)
Cache game data resources on demand	Loads a profile's resources only when first used (recommended)
Load resources from archives	Reads models and sprites from .pak/.pk3 archives
Use large map toolbar icons	Large icons in the toolbars
Transform angles of point entities when rotating	Rotating a selection also rotates point entities' angles
Snap selection to grid when dragging	Moves snap to the grid (otherwise per pixel)
Don't snap primitive points to integer grid	Vertices of new shapes are not rounded
Stretch arches to fit original bounding rectangle	Arches fill the whole drawn rectangle
Preserve internal connections in cloned entities	target/targetname links stay internal to the copies
Restore cut objects if not pasted	An object cut (Ctrl+X) and never pasted reappears
Precise selection of models	Selects on the model's exact shape, not its box
ALT+drag to scale models	Scale a model with Alt + drag (game-dependent)
ESC to clear selection	Esc clears the selection
Memory usage	Memory / texture quality slider; 4th notch from the left recommended
Undo levels	Number of undo steps (50 by default)

2D Views tab

Option	Effect
Crosshair cursor	Crosshair cursor over brushes
Default to fixed-degree rotation	Rotation in fixed steps (15°) instead of 1°
Display scrollbars	Scrollbars (otherwise: Space + drag)
Draw vertices	Shows brush vertices
Draw camera	Shows the 3D camera (a blue eye)
Draw models	Draws models instead of their box
Draw sizing info	Shows dimensions next to the selection

Option	Effect
Center on camera after movement in 3D	Re-centres the 2D views on the camera after moving in 3D
Use VisGroup colors for object lines	Brush lines in their VisGroup's colour
Keep VisGroup when clone-dragging	Clones keep the original's VisGroup
Arrow keys nudge selected object/vertex	Arrow keys move the selection
Reorient primitives on creation in the active 2D view	Orients new shapes according to the view they are drawn in
Automatic infinite selection in 2D windows (no ENTER)	Box selection works without Enter
Selection box selects by center handles only	The box only takes objects whose centre is inside
Grid: Size	Default grid for new maps, from 1 to 512
Grid: Highlight every 8 grid lines	A stronger line every N lines (N adjustable)
Grid: Hide small grid	Hides the grid when it is under 4 pixels
Grid: Dotted grid	Dotted grid instead of lines

3D Views tab

Option	Effect
Back clipping plane	Distance beyond which nothing is drawn; lower = smoother
Filter textures	Texture smoothing; **F** key in the 3D view
Alpha blending	Shows transparency
Animate textures	Animates textures and effects
Animate models	Animates models
Model render distance	Beyond it, models are replaced by their box
Energy saving mode	Uses less power, movement sometimes less smooth
Cap framerate at 60 fps	Limits to 60 frames per second
Multisample antialiasing (MSAA)	Anti-aliasing
Field of view	Camera view angle: 90 by default, 80 to 120 recommended
Mouse sensitivity	Mouse sensitivity
Use mouselook navigation	Free-look (Z key, then W/A/S/D)
Reverse mouse Y axis	Inverts up and down
Forward speed / Time to top speed	Maximum speed and time to reach it

Game Profiles tab

Each profile has four sub-tabs (Part 1, chapter 1):

Sub-tab	Settings
Game	FGD files, map type, default point and brush entities
Directories	Game executable, Base Game Directory, Mod Directory, Source Maps Directory, Model Viewer
Textures	Texture files (Add Files…), texture brightness, default scale (1.0), palette file
Build Programs	Paths and parameters for CSG, BSP, VIS and LIGHT (RAD)

Misc tab

Option	Effect
Backup existing files on save	Keeps a copy on every save (.jmf becomes .jmx)
Enable autosave	Automatic save, every 5 minutes by default (adjustable)
Keep the last N autosaves	Number of saves kept (0 = unlimited)
Autosave modified files only	No save if nothing changed
Beep on successful autosave	Beep on each save
Use directory of the current file	Saves next to the map (otherwise: folder of your choice)
Steam Cloud (Steam version)	Online storage of maps, saves and backups, synced at startup

Colours (3D view background, grid, selection…)

Colours are not in the Options window but in **View > Color Schemes**: J.A.C.K. offers ready-made colour schemes and a command to define your own, with the colours of the views and windows (manual, 5.1; official site). This is where you change, for example, the views' background colour.

> **Note:** the 2D views can also show a **background image** (BMP or PNG), with adjustable offset, scale, brightness, smoothing and colour inversion; these settings are saved in the .jmf (official site). Handy for reproducing a plan.

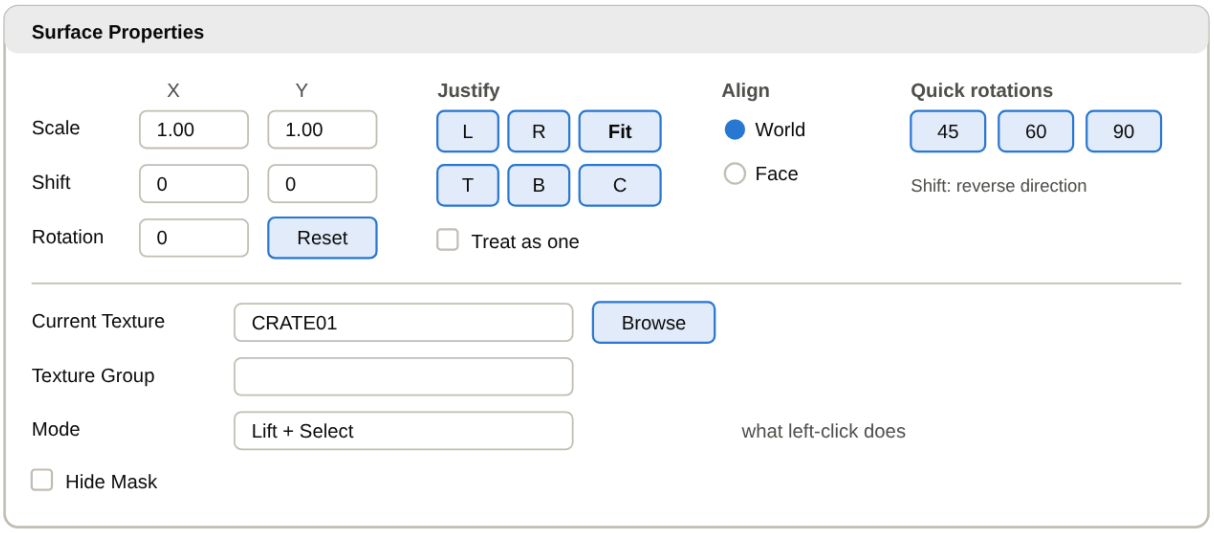
> **Note:** some advanced settings, such as showing values in SmartEdit mode, are in the `VDKSettings.ini` file (official site).

Sources: J.A.C.K. Reference Manual, J.A.C.K.: Features.

4. Other windows

Surface Properties (Shift+A)

Shift+A opens Surface Properties: every setting for a face



Surface Properties · fields and buttons

Control	Effect
Scale X / Y	Texture size; a negative value mirrors it
Shift X / Y	Offset
Rotation, Reset	Angle; Reset sets scale, offset and rotation back to zero
Justify: L, R, Fit, T, B, C	Align left, right, fit, top, bottom, centre
Treat as one	Treats the selected faces as one
Align: World / Face	Projection by world or by face
45, 60, 90	Quick rotations; with Shift, the other way
Texture Group, Current Texture	Texture group and current texture
Hide Mask	Hides the red tint of selected faces
Mode	What left-click does: Lift + Select, Lift, Select, Apply (texture only), Apply (texture + values)

J.A.C.K. adds the modes **NULL to Selected**, **NULL to Unselected** and **Apply (texture + values + axes)** (official site).

Object Properties (Alt+Enter)

- **Class Info**: the attributes, the **SmartEdit** button, the Yaw box and its compass, a **Help** button, and buttons to browse sprites, sounds and models.
- **Flags**: the checkboxes; their numeric value shows on hover.
- **VisGroup**: VisGroup membership, and access to the group manager.

Paste Special (Edit)

Field	Effect
Number of copies	Number of copies
Start at center of original	Copies start from the centre of the original
Keep internal connections	target/targetname links stay internal to each copy
Group copies	Groups the copies
Offset (X, Y, Z) and Rotation (X, Y, Z)	Offset and rotation between two copies; the "<" buttons copy a dimension of the selection

Replace Textures (Tools)

- **Find / Replace**: textures to look for and to put, with preview.
- **Replace in**: the whole map or only marked objects, hidden objects included or not.
- **Action**: Replace exact matches, Replace partial matches, Substitute partial matches, or Do not replace textures (only mark the brushes found).
- **Modify texture scale by factor**: adjusts the scale when the new texture is a different size.

Hollow (Ctrl+Shift+H)

A single field, the wall thickness. A **negative** value hollows outwards. Confirmation is asked if several brushes are selected.

Check for Problems (Alt+P)

Lists the map's errors (for example: no `info_player_start`). Buttons: **Go to error** (centres the views on it), **Fix** (fixes this one), **Fix all (of type)** (all of the same type), **Fix all errors**.

Entity Report (Map)

Lists entities with their class, name and target. Buttons **Mark**, **Go to**, **Delete**, **Properties**, **Hide**, **Unhide** (hidden entities are in italics). Filters: type (all, brush, point), hidden entities, search by key/value (with **Exact**) and by class.

Small windows

Window	Use
Go to Brush (Ctrl+Shift+G)	Entity and brush number; tick "Search visible brushes only" if the map was exported with "Save visible objects only"
Go to Coordinates	Coordinates without commas, e.g. `304 -384 0`
Map Information	Map statistics
Entity Help	Entity help, generated from the FGD
Default Room Properties	When creating a map, a first room: size, wall, floor and ceiling textures, "Apply NULL to backfaces" option. Shift + click on New reopens it

Window	Use
Transformation	Move, rotate or scale with exact values
Automatic Configuration (Ctrl+F2, Steam)	Creates profiles for installed games; "Remove all existing game configurations" starts from scratch (irreversible)

Sources: J.A.C.K. Reference Manual, chapters 3.4 and 5.2, J.A.C.K.: Features.

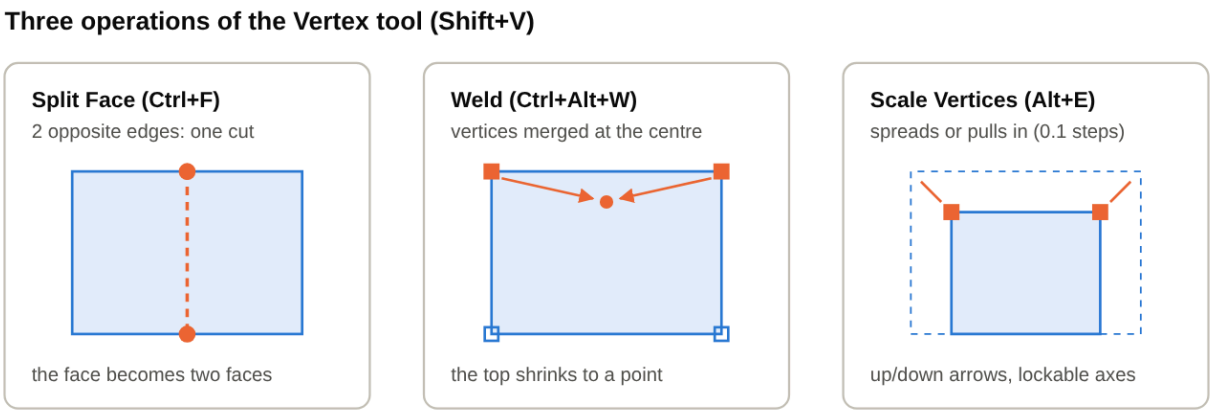
5. Advanced tools

Cameras and navigation in the 3D view

Action	Command
Camera tool	Shift+C
Create a camera	Shift + drag in a 2D view (the point = the eye, the line = the direction)
Next / previous camera	Page Down / Page Up (camera tool active)
Delete the current camera	Del
Move eye and direction together	Ctrl + drag
Free-look	Z, then mouse + W, A, S, D
Turn the view on the spot	Drag with the left button
Up, down, sideways	Drag with the right button
Strafe sideways	Drag with both buttons
Forward, backward	Shift + drag with the right button

J.A.C.K. saves cameras in the map file (.jmf, .rmf, .vmf) (official site).

The Vertex tool



Vertex tool · split, weld, scale

- Pressing **Shift+V** repeatedly cycles three modes: vertices and edges (default), vertices only, edges only.
- **Alt+click** avoids deselecting the brush.
- To undo (Ctrl+Z), first leave the Vertex tool.
- J.A.C.K. **prevents invalid shapes** (non-convex): moving gets harder as you approach one.
- **Split Face** (Ctrl+F) splits a face between two selected opposite edges; **Weld Vertices** (Ctrl+Alt+W) merges the selected vertices at their centre; **Scale Vertices** (Alt+E) spreads or pulls the vertices together, in 0.1 steps with the arrow keys, with boxes to lock certain axes.
- The context menu (right-click) also offers **Triangulate**, which splits a twisted face into flat triangles.

Clipping, Merge and Carve

- **Clipping** (Shift+X): Shift+X or another click on the button changes mode; Ctrl + drag moves the whole cut line; Enter cuts; the **O** key in a 2D view shows the size of the pieces.
- **Merge** (Ctrl+Shift+M): merges several brushes into one, if together they form a convex volume.
- **Carve** (Ctrl+Shift+C): do the small square holes first, then the round ones; avoid carving with many-sided shapes (arches, spheres).

The Arch tool

Choose the **arch** shape in the Object Bar and draw a rectangle: a settings window opens.

Setting	Effect
Wall width	Thickness of each piece
Number of sides	From 3 to 100 pieces
Arc	From 8 to 360 degrees; the **Circle** button sets 360
Start angle	Starting angle; 0 = top, clockwise
Add height	Height added to each piece: gives a spiral staircase
Group parts	Groups the pieces
Fit selection	Fits the pieces to the whole rectangle
Triangulate	Triangular prisms instead of quads
Smooth slope	Continuous ramp instead of steps (with Add height)

Example: a spiral staircase. Arc = 360, Number of sides = 16, Add height = 16, Group parts ticked.

The Path tool (path_corner paths)

Action	Command
First point	Shift + click in a 2D view; a window asks for the name, direction (one way, circular, ping-pong) and entity (usually `path_corner`)
Add a point at the end	Shift + click on an existing point
Insert a point before	Select a point, Insert
Delete points	Del; Shift+Del deletes the whole path
Convert to linked entities	Ctrl+Shift+E (cannot be undone)

It is the fastest way to lay out the route of a `func_train` (Part 2, chapter 16).

Models

- With an entity that takes a model, the "..." button in its properties browses .mdl files; body groups and skins show in the preview.
- Right-click a model: **View model...** opens it in the viewer set in the profile, **Reload model** reloads it from disk.
- With the **ALT+drag to scale models** option, Alt + drag scales a model, if the game supports it.

Link two entities in one gesture

Select the calling entity, then the called one, and press **Ctrl+K** (Tools > Connect entities): J.A.C.K. fills the first one's Target with the second one's Name.

Sources: J.A.C.K. Reference Manual, chapters 3 and 4, J.A.C.K.: Features.

Part 2: Entities and special effects

Bring your map to life: lights, doors, triggers, effects and game modes.

1. Entities

Brushes form the skeleton of the map; entities are its organs. Anything that moves, lights up, makes noise or reacts to the player is an entity.

Point entities and brush entities

	Point entity	**Brush entity**
What it is	An object placed at a precise point, with no volume	One or more brushes given a role
How to create it	**Entity** tool (Shift+E), click in a 2D view, Enter	Select the brushes, then **To Entity** (Ctrl+T)
Examples	`light`, `info_player_start`, `ambient_generic`	`func_door`, `func_buyzone`, `trigger_once`
To remove it	Del	**To World** (Ctrl+W) returns the brushes to the world

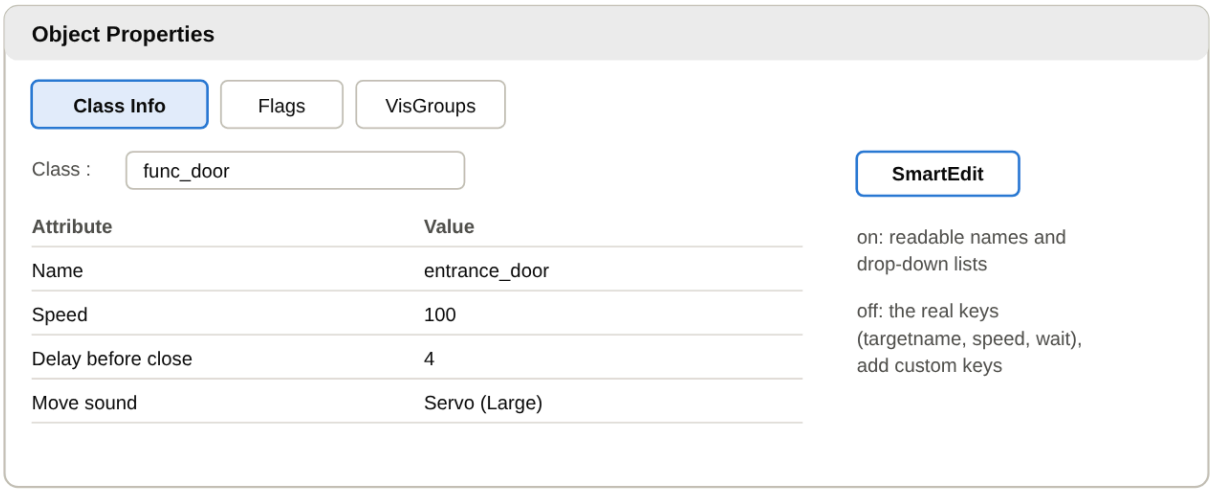
> **Note:** in J.A.C.K., right-clicking in a 2D view opens an entity creation menu, faster than the list in the right panel (VDC: J.A.C.K.).

> **Warning:** a brush entity is not a world brush. It does not seal the map (the void is still void behind a door) and, by default, it casts no shadow.

Entity properties

Select the entity and press **Alt+Enter**.

Alt+Enter opens the properties: class, keyvalues and flags



entity properties · Object Properties window

- **Class**: the entity type. You can change it here: a `func_wall` becomes a `func_door` without rebuilding the brushes.
- **Attributes** (keyvalues): each entity has its own, defined by the FGD.
- **Flags**: checkboxes that change the behaviour ("Use only", "Starts open"…).
- **SmartEdit**: when on, the editor shows readable names and drop-down lists. When off, it shows the real keys and lets you add some by hand, which is needed for `multi_manager` or compiler keys (`zhlt_…`).

J.A.C.K. improves SmartEdit mode with a preview of the chosen values (VDC: J.A.C.K.).

Attributes you will see everywhere

Attribute	Key	Role
Name	`targetname`	The entity's name, so another one can call it
Target	`target`	The name of the entity to call
Delay before trigger	`delay`	Wait in seconds before calling the target
Kill target	`killtarget`	Entity to delete when the call happens
Master	`master`	Entity that allows or blocks this one (chapter 5)
Pitch Yaw Roll	`angles`	The orientation
Render FX, Render Mode, FX Amount, FX Color	`renderfx`, `rendermode`, `renderamt`, `rendercolor`	The appearance ("Render properties" appendix, Part 4)

The **Name / Target** pair is the heart of a map's logic: it is the subject of chapters 4 and 5.

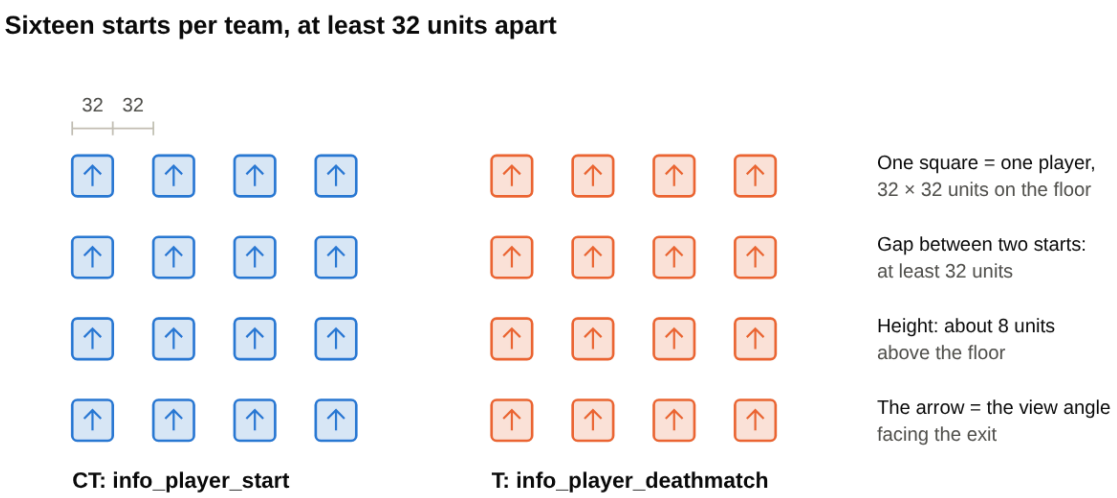
Sources: VDC: List of Counter-Strike entities, VDC: J.A.C.K..

2. Player start positions

Entities: `info_player_start`, `info_player_deathmatch` · **Type:** point entity · **Difficulty:** easy

Without a start point, nobody can spawn in your map. In Counter-Strike, each team has its own (TWHL: Introduction to Counter-Strike mapping):

- `info_player_start`: **Counter-Terrorist** start (CT);
- `info_player_deathmatch`: **Terrorist** start (T).



spawn area · 16 CT and 16 T seen from above

Place the starts

1. Take the **Entity** tool, choose `info_player_start` and click in the Top view where you want it.
2. In the Front view, place it about 8 units above the floor.
3. In its properties, set the angle (the small **Angles** dial) so players face the exit of the area.
4. Clone it (Shift + drag) to make a grid, leaving at least 32 units between two starts.
5. Do the same with `info_player_deathmatch` on the other side of the map.

How many?

About **16 per team**, 32 in total: that is the most the game can handle (TWHL). If a server gets more players than a team has starts, the extra players cannot spawn.

> **Common mistake:** starts that are too close, or that touch a wall or the ceiling, spawn players stuck, or killing each other as they spawn. Leave room, and check that no start pokes through a brush.

> **Note:** without a `func_buyzone` entity, a small buy zone is created automatically around each start. To decide for yourself, see chapter 20, "Counter-Strike game modes".

Test

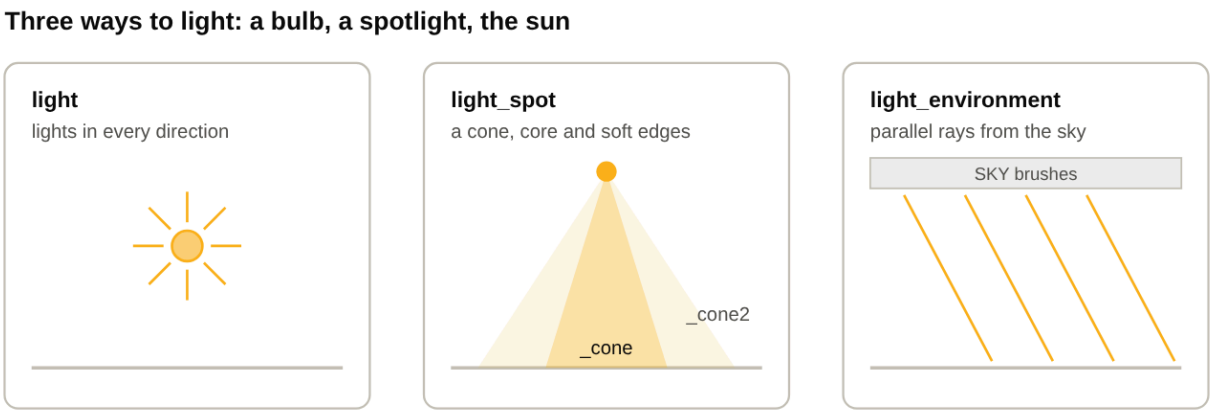
Run the map with bots ("Bots and waypoints" chapter, Part 4) and reload it full several times: it is the only way to see whether all starts are safe.

Source: TWHL: Introduction to Counter-Strike mapping.

3. Lights!

Entities: `light`, `light_spot`, `light_environment` · **Type:** point entity · **Difficulty:** easy

A map without light is completely black (or completely bright, if it leaks). Lighting is computed once and for all at compile time, by the RAD program: it does not change in game, except for switchable lights.



lighting · light, light_spot, light_environment

A lamp: `light`

Place the entity below a ceiling, near an object that justifies the light (neon, lamp).

- **Brightness**: four numbers, `R G B Intensity`. For example `255 255 200 250` gives a warm white light of intensity 250. Intensity can exceed 255.
- **Appearance** (style): flicker, pulse, candle, faulty neon…
- **Name**: a named light becomes **switchable**: a button or trigger can turn it on and off.
- **Initially dark** flag: the light starts off.

The sun: `light_environment`

It lights every surface that can see the sky, with parallel rays, from brushes textured **SKY**. You need **only one per map**.

- **Pitch**: the tilt of the rays, from -90 (straight down) to 0 (horizontal).
- **Angles** (yaw): the direction the sun comes from.
- **Brightness**: the sun's colour and intensity, in the same format as `light`.

> **Warning:** without a visible SKY brush, `light_environment` lights nothing. It is the classic mistake behind a pitch-black outdoor map.

Spotlights: `light_spot`

A projector that lights a cone (TWHL: light_spot).

- **Pitch**: from -90 (down) to 90 (up).
- **Inner Radius** (`_cone`): the angle of the brightest core.
- **Outer Radius** (`_cone2`): the angle where the light fades out. Above 90 the value has no effect.
- **Target**: aims at an entity (preferably an `info_null`) instead of setting the angle by hand.
- **Is Sky**: set to Yes, the spot behaves like a `light_environment`.

Light-emitting textures

Some textures emit light themselves: neons, screens, ceiling panels. The list is in the `lights.rad` file, placed next to the compilers and read by RAD: each line pairs a texture name with a colour and an intensity (TWHL: Compiling 101). If this file is empty or missing, those textures light nothing.

Switching a light on and off

Example of a switch:

1. Place a `light` and name it `room_lamp`.
2. Create a small brush on the wall and make it a `func_button` (Ctrl+T).
3. In the button: **Target** = `room_lamp`, **Delay before reset** = 1, **Don't move** flag ticked.
4. Compile: each press of the button turns the lamp on or off.

> **Warning:** a map can contain at most **31 differently named lights**, and a single face can receive only 4 light styles (TWHL: light_spot). Give all the lamps in one room the same name if they should switch together.

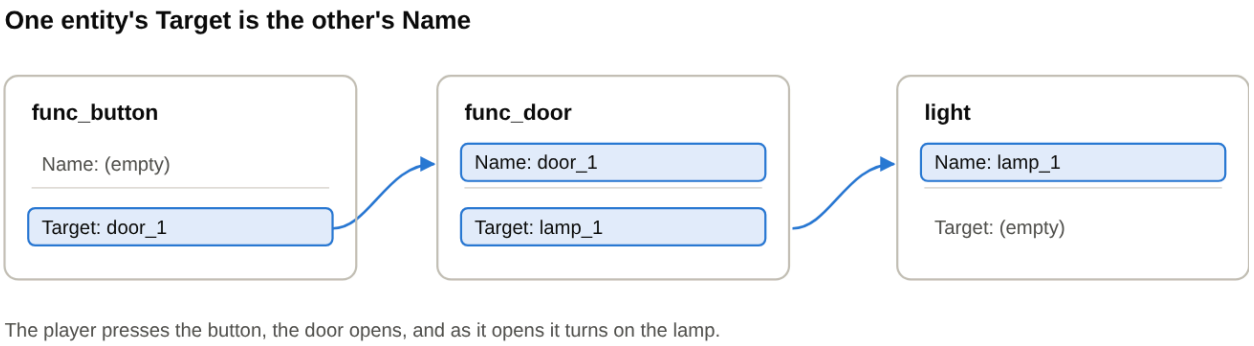
Sources: TWHL: light_spot, TWHL: Compiling 101.

4. Triggers (A: simple)

Entities: `trigger_once`, `trigger_multiple`, `trigger_auto`, `func_button`, `func_rot_button` · **Type:** brush and point entities · **Difficulty:** easy

A door that opens, a light that turns on, an explosion: it all starts with a trigger that "calls" another entity.

Entity calls



entity calls · Target to Name, in a chain

One entity calls another through two fields:

- **Name** (`targetname`): the name of the called entity;
- **Target** (`target`): the name of the entity to call.

When the trigger fires, it looks for every entity whose **Name** equals its **Target**, and activates them. A door opens (or closes), a named light turns on (or off), a sound plays. Several entities can share the same name: they will all react together.

Three more fields complete the call:

- **Delay before trigger** (`delay`): wait, in seconds, before calling the target;
- **Kill target** (`killtarget`): name of an entity to **delete** when the call happens;
- **Master**: name of a `multisource` that allows or blocks the entity (next chapter).

> **Tip:** use meaningful names with no spaces: `bomb_a_door`, `corridor_lamp`. With 50 entities, `door1` and `door2` quickly become a puzzle. J.A.C.K. draws the links between entities in the views, which helps you keep track.

On first pass: `trigger_once`

An invisible volume that fires its target the first time a player enters it, then disappears.

1. Create a brush and texture it with **AAATRIGGER**.
2. Ctrl+T, class `trigger_once`.
3. **Target**: the name of the entity to fire.

The AAATRIGGER texture is a convention: triggers are invisible in game whatever their texture, but it lets you spot them in the editor (VDC: Tool textures).

On every pass: `trigger_multiple`

Same as `trigger_once`, but it re-arms. The **Delay before reset** attribute (`wait`) sets the time, in seconds, before it can fire again.

When the map loads: `trigger_auto`

A point entity that fires its target by itself when the map starts, after the given delay. Useful to start ambient music, light up scenery or start a train.

Buttons: `func_button`

A brush the player activates by pressing the **Use** key (E), or by touching it if the flag allows.

Attribute or flag	Role
Target	The called entity
Delay before reset	Seconds before the button returns. -1 = stays pressed forever
Sounds	The button's sound
Speed, Lip	Press speed and distance
Don't move flag	The button does not move in (wall switch)
Toggle flag	One press to activate, a second to deactivate
Sparks flag	Throws sparks
Touch Activates flag	Touching the button is enough

To make a button change appearance when activated, give one of its faces a `+0…` texture that has a `+A…` version: the button switches from one to the other (VDC: Texture prefixes, TWHL: Doors).

Rotating buttons: `func_rot_button`

A lever or crank that turns instead of pressing in. Like any rotating entity it needs an **ORIGIN** brush, tied with it, at the axis. The **Distance** attribute gives the rotation angle in degrees.

> **Complete example:** a button that opens a door. Make the door (`func_door`, Name = `door_1`), then the button (`func_button`, Target = `door_1`, Delay before reset = 3, Don't move flag). Doors are covered in detail in chapter 8.

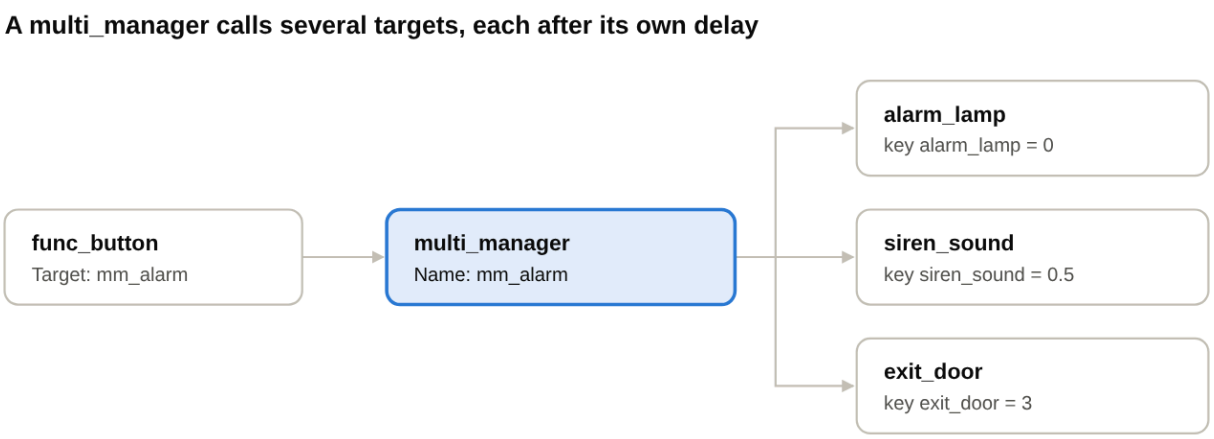
Sources: TWHL: Doors, VDC: Texture prefixes, VDC: Tool textures (GoldSrc).

5. Triggers (B: complex)

Entities: `multi_manager`, `multisource`, `trigger_relay`, `trigger_changetarget`, `trigger_counter` · **Type:** point entities (except `trigger_counter`, a brush entity) · **Difficulty:** medium

A simple trigger calls one target at a time. For a real sequence (an alarm that turns on lamps, sounds a siren, then opens a door), you need logic entities.

Call several entities in order: `multi_manager`



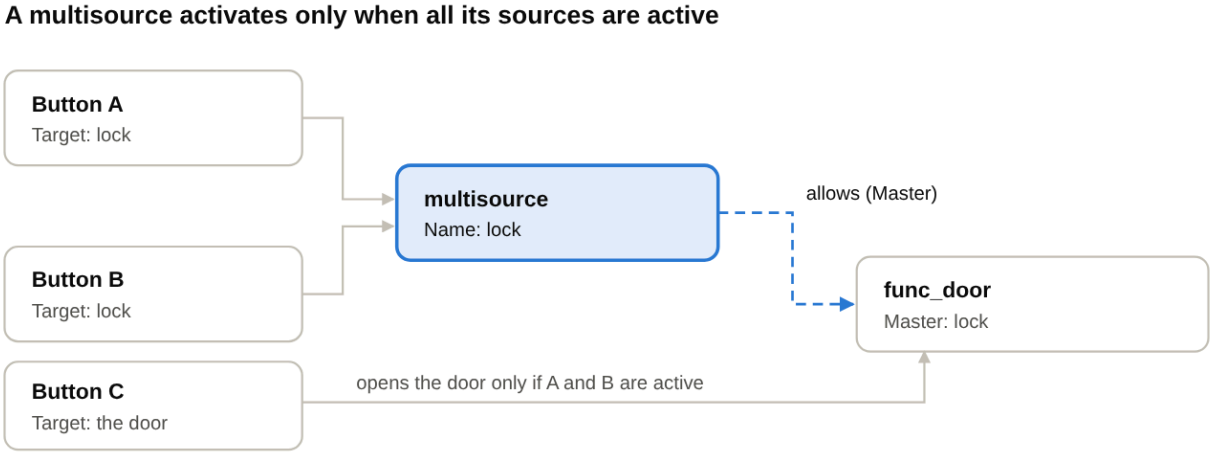
multi_manager · 1 call, 3 staggered targets

The `multi_manager` calls up to **16 targets**, each after its own delay (TWHL: multi_manager).

1. Place a `multi_manager` and give it a **Name** (`mm_alarm`).
2. Point your trigger at it (**Target** = `mm_alarm`).
3. In its properties, **turn SmartEdit off**.
4. Click **Add**: the **key** is the target's name, the **value** is the delay in seconds (`siren_sound` = `0.5`).
5. Repeat for each target.

> **Note:** to call the same target twice, add `#1`, `#2`… to the end of the key: `lamp`, `lamp#1`, `lamp#2`. The game ignores this suffix, which only keeps the keys distinct (TWHL). Example: `lamp` = 0, `lamp#1` = 1, `lamp#2` = 2 makes a lamp blink.

Lock an entity: `multisource`



multisource · two conditions to open

The `multisource` is a logical "AND": it is only active when **all** triggers aiming at it are active (TWHL: Multisource).

It is mostly used as a **Master**: an entity whose Master field holds the multisource's name only works while the multisource is active.

> **Warning:** the multisource does not fire the door itself, it allows it. And if a button resets, the multisource becomes inactive again: the door locks again. Set reset delays long enough for the player to act.

Other logic entities

Entity	Role	Key attributes
`trigger_relay`	Relay: receives a call and passes it on, with a delay and a forced state (On, Off or Toggle)	Trigger state; **Remove on fire** flag so it works only once
`trigger_changetarget`	Changes another entity's **Target** during play	Target = the entity to change, New target = its new target (TWHL)
`trigger_counter`	Fires its target only after being activated a set number of times	Count before activation

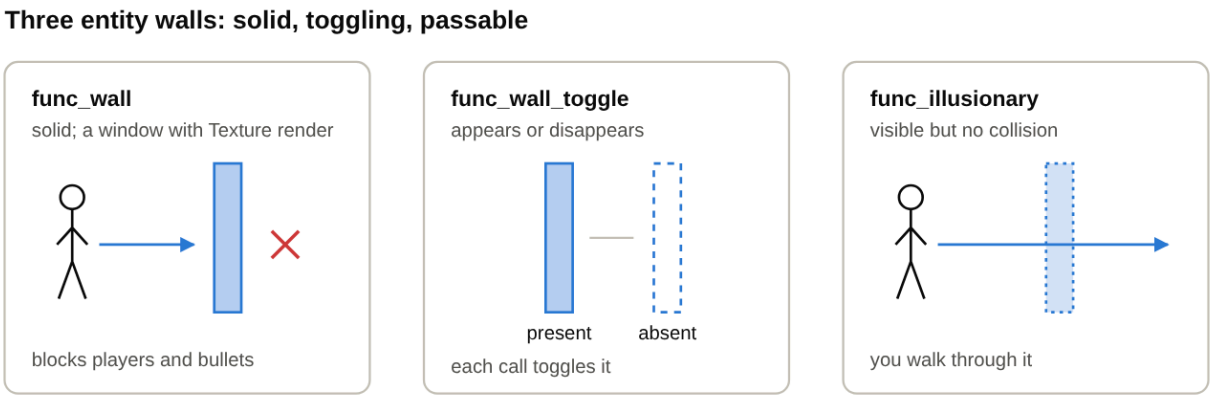
Example with `trigger_counter`: three targets to destroy (`func_breakable`), each with Target = `counter`. The `trigger_counter` named `counter`, with a count of 3, opens the door when the third one breaks.

Sources: TWHL: multi_manager, TWHL: Multisource, TWHL: trigger_changetarget.

6. Walls

Entities: `func_wall`, `func_wall_toggle`, `func_illusionary` · **Type:** brush entity · **Difficulty:** easy

A world brush is always opaque, solid and still. Turned into an entity, a wall can become transparent, appear and disappear, or let things pass through.



entity walls · func_wall, func_wall_toggle, func_illusionary

Transparency

All these entities have the render properties (**Render Mode**, **FX Amount**…). The two settings to know (TWHL: func_wall, TWHL: Render Properties):

- **Window**: Render Mode = **Texture**, FX Amount around **100**. The lower the value, the more transparent (0 = invisible, 255 = opaque).
- **Grate or fence** (`{` texture): Render Mode = **Solid**, FX Amount = **255**. The blue parts of the texture disappear.

Simple walls: `func_wall`

A fixed, solid brush that is nonetheless an entity. It is used for:

- windows and grates (settings above);
- **small details** (pipes, light fittings, ledges): as entities they do not cut the world brushes around them, which avoids needless faces and light seams (TWHL: func_wall);
- walls whose `+0` / `+A` texture must switch when triggered.

> **Trick:** a CLIP-textured face on a `func_wall` lets projectiles pass through that side (TWHL).

Toggle walls: `func_wall_toggle`

A wall that disappears or reappears each time it is called. Give it a **Name**, and have a button or trigger aim at it. The **Starts Invisible** flag makes it start absent.

Example: a secret passage. A `func_wall_toggle` named `passage` blocks a corridor; a hidden button with Target = `passage` makes it disappear.

Illusionary walls: `func_illusionary`

Visible, but with **no collision at all**: neither players nor bullets hit it. It is used for foliage, curtains, cobwebs, or a fake wall hiding a passage.

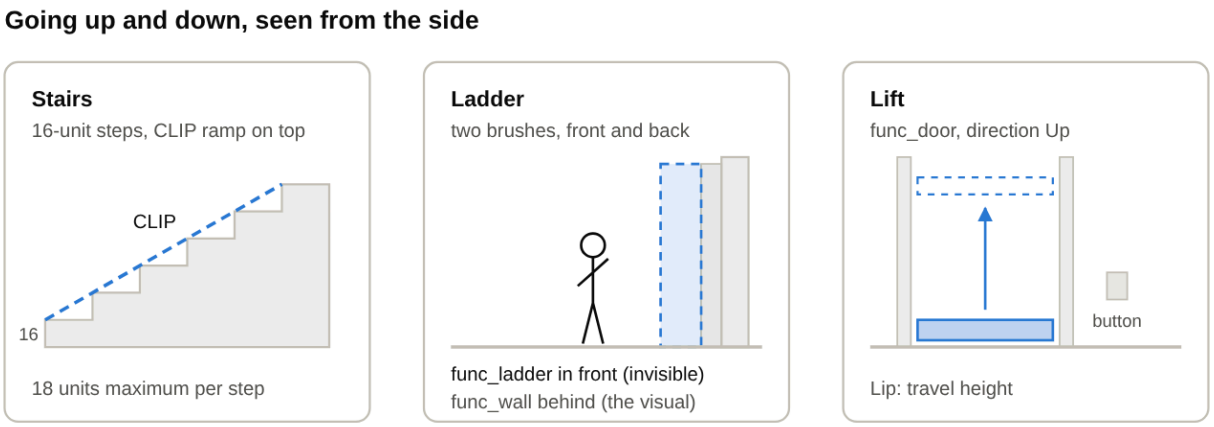
> **Warning:** like any entity, a `func_illusionary` does not seal the map. Behind a fake wall there must always be a real world brush to stop the void.

Sources: TWHL: func_wall, TWHL: Render Properties.

7. Going up and down

Entities: `func_ladder`, `func_door`, `func_plat` · **Type:** brush entity · **Difficulty:** easy to medium

Three ways to change floors: stairs, ladders, lifts.



vertical movement · stairs, ladder, lift

Stairs

Stairs are made only of world brushes, but they must respect the engine: a player climbs a step of **at most 18 units** on their own (VDC: GoldSrc Dimensions). Higher than that, they must jump.

- Steps **16 units high** and 16 to 32 deep: the most common value.
- 8-unit steps: more realistic, but the player's view jumps on each step.

> **Trick:** lay a ramp-shaped **CLIP** brush over the stairs, from the nose of the first step to that of the last. The player glides smoothly, bullets still hit the real steps.

Ladders: `func_ladder`

The `func_ladder` is an invisible volume the player climbs in. The ladder's look is another brush (TWHL: Ladders).

1. Create the visible ladder brush, with a ladder texture (32 units wide for a 32 texture).
2. If the texture is masked (`{ladder…`), make it a `func_wall` with Render Mode = **Solid** and FX Amount = **255**.
3. Copy that brush and place the copy just in front.
4. Make the copy a `func_ladder`, textured **AAATRIGGER** to spot it.

> **Warning:** do not use `CLIPHULL` textures around a ladder: it behaves oddly and the player climbs too fast (TWHL: func_ladder).

> **Note:** a `func_ladder` can be sloped (a slanted mesh) or horizontal (the player then moves sideways) (TWHL).

Lifts

With a `func_door` (the most flexible):

1. Create the platform and make it a `func_door`.
2. **Angle** (or Move direction) = **Up**: the "door" rises instead of sliding sideways.

3. **Lip**: the distance subtracted from the travel. By default a door moves by its own height; for an 8-unit platform that must rise 128, set Lip = -120.
4. Give it a **Name**, tick the **Toggle** flag, and place one button at the top and one at the bottom, with Target = that name.

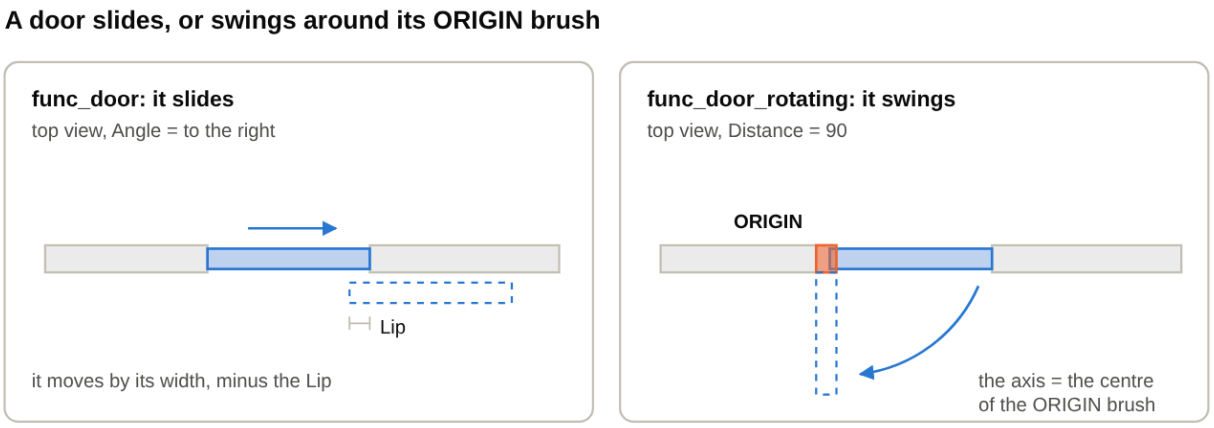
With a `func_plat`: an automatic platform that rises when stepped on and comes back down after a few seconds. The **Travel altitude** setting sets the height. `func_platrot` does the same while rotating.

> **Warning:** if a player blocks a lift, it goes back the other way, unless you set damage (**Damage inflicted when blocked**). To avoid accidents, keep damage at 0 and leave enough room above the platform.

Sources: VDC: GoldSrc Dimensions, TWHL: Ladders, TWHL: func_ladder.

8. Doors

Essential to almost every map, doors come in two families: those that slide and those that swing.



doors · sliding and rotating, seen from above

Sliding doors

Entity: `func_door` · **Type:** brush entity · **Difficulty:** fairly easy

A door that opens by sliding sideways, up or down.

1. Create a brush with a door texture, the same size as the texture.
2. Select it and click **To Entity** (Ctrl+T).
3. Choose the `func_door` class.

Useful attributes (TWHL: Doors):

Attribute	Role
Angle	The opening direction: the small dial, or **Up** / **Down** in the list
Name	To open it with a button or trigger

Attribute	Role
Speed	Opening speed
Master	A `multisource` that must be active for the door to open
Move sound / Stop sound	The sound during and at the end of the movement
Delay before close	Seconds before closing automatically. **-1** = stays open
Lip	What remains visible once open. 0 = the door disappears completely into the wall
Damage inflicted when blocked	Damage dealt to whoever blocks it
Health (shoot open)	Opens when shot

Flag	Role
Starts Open	Starts open
Passable	Can be walked through without blocking
Toggle	One call opens, the next call closes
Use Only	Opens only with the Use key, not by walking up to it

> **Note:** a door with no Name opens when a player touches it. With a Name, it waits to be called (or used, if the Use Only flag is ticked).

Rotating doors

Entity: `func_door_rotating` · **Type:** brush entity · **Difficulty:** medium

1. Create the door brush and texture it.
2. Create a small **ORIGIN**-textured brush at the hinges: its centre will be the rotation axis.
3. Select **both** (Ctrl+click), then Ctrl+T and `func_door_rotating`.
4. **Distance** = 90 (the opening angle in degrees).

Flags specific to this entity (TWHL: func_door_rotating): **Reverse Direction** (turns the other way), **One-way** (opens in one direction only), **X axis** / **Y axis** (by default the door turns around the vertical Z axis).

> **Common mistake:** forgetting to tie the ORIGIN brush to the door. It then rotates around the centre of the map and vanishes into the scenery (TWHL: Doors).

> **Counter-Strike quirk:** a rotating door still moving at the end of a round may keep turning into the next round, until its cycle ends (TWHL).

Momentary doors

Entities: `momentary_door`, `momentary_rot_button` · **Difficulty:** medium

A door that opens as long as the player turns a valve, and stops when they let go. The valve (`momentary_rot_button`, with an ORIGIN brush) has the `momentary_door`'s name as its Target.

The button and the door

To open a door with a button: the door has a **Name** (`door_1`), the button has **Target** = `door_1` and the **Don't move** flag. For a double door, create two `func_door`s opening in opposite directions and give them the **same Name**: a single button opens both.

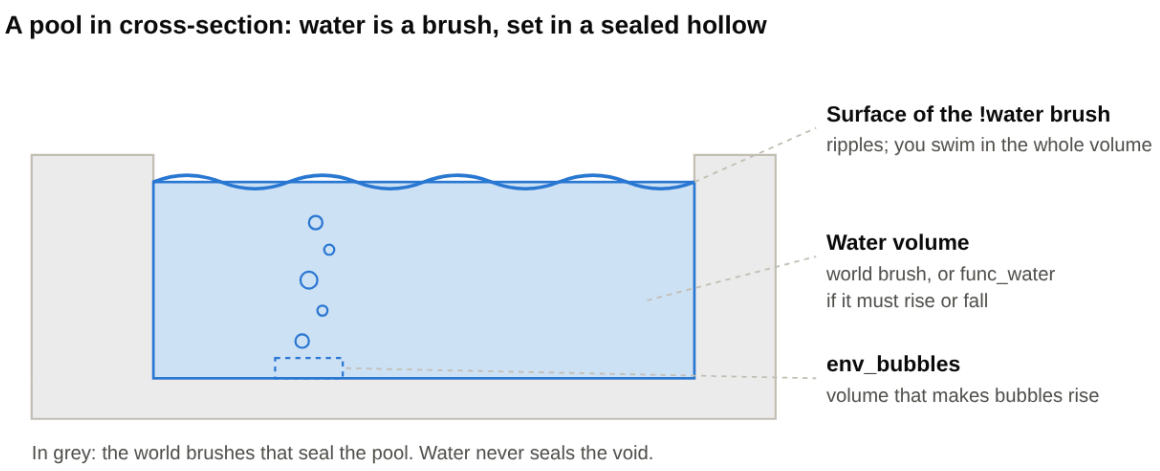
The glass door trick

A glass door is nothing special: it is a door whose rendering you set. Use a glass texture, Render Mode = **Texture**, FX Amount around **100**. For a door with an opaque frame and a pane, make two overlapping `func_door`s with the same Name: one for the frame, the other for the transparent pane.

Sources: TWHL: Doors, TWHL: func_door_rotating, TWHL: Render Properties.

9. Water and bubbles

Entities: `func_water`, `env_bubbles` · **Type:** brush entity · **Difficulty:** easy



pool · cross-section with water and bubbles

Water

In GoldSrc, water does not require an entity: it is a **brush textured with a texture starting with `!`** (`!water`, `!waterblue`…). That prefix alone makes the volume liquid (VDC: Texture prefixes).

1. Dig the pool into the world brushes.
2. Create a brush that fills the pool up to the water level.
3. Apply a `!` texture to all its faces.

Method	What you get
World brush with `!water`	Swimmable water, rippling surface
`func_water`	The same water, but as an entity: only the top face is drawn, and the water can move like a door (level rising or falling when called)
`!slime`, `!lava`	Harmful liquid (lava deals double damage), world brush only
`!cur_0`, `!cur_90`…	Current that pushes the player in one direction, world brush only

> **Warning:** water does not seal the map. The pool must be fully surrounded by world brushes, including on the sides that rise above the surface.

> **Counter-Strike quirk:** a bomb planted in water does not explode and does not end the round. Never put a bomb site in or near water (TWHL: Introduction to CS mapping).

Bubbles: `env_bubbles`

A brush entity that makes bubbles rise throughout its volume. Set the **density** (number of bubbles), the **frequency** and the rising **speed**. With a Name, it can be switched on and off.

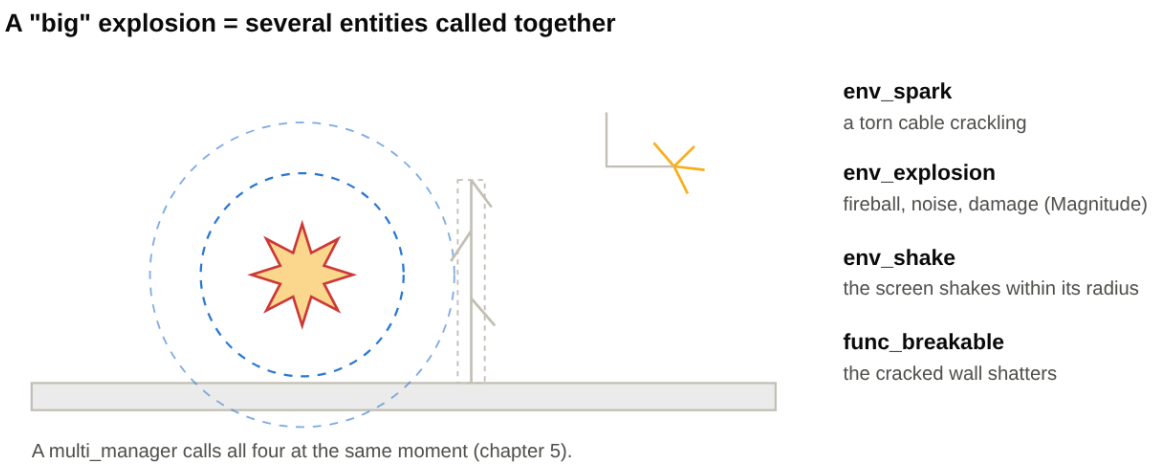
Example: a small `env_bubbles` brush at the bottom of the pool, under a vent grate, to fake an air inlet.

Sources: VDC: Texture prefixes, TWHL: Introduction to Counter-Strike mapping.

10. Spectacular effects

Entities: `env_spark`, `env_shake`, `env_explosion`, `func_mortar_field` · **Type:** point entities (except `func_mortar_field`) · **Difficulty:** easy

A good explosion is never a single entity: it is a fireball, a screen shake, debris and sparks called at the same time.



effects · explosion, shake, debris, sparks

Sparks: `env_spark`

Sparks and their crackle, at irregular intervals. **Max Delay** sets the longest wait between two bursts. With a Name and the **Toggle** and **Start ON** flags, it can be switched on and off. Place it where the effect makes sense: a torn cable, an electrical box.

Shakes: `env_shake`

Shakes the screen of nearby players.

Attribute	Role
Amplitude	Strength of the shake, from 0 to 16
Frequency	Jolts per second; around 40 gives a good result
Duration	Duration in seconds
Effect radius	Radius of effect
GlobalShake flag	The whole map shakes, whatever the radius

Values from the entity's VDC page, written for Source: use them as a starting point and check in game.

Explosions: `env_explosion`

An explosion at the entity's position when it is called (VDC: env_explosion (GoldSrc)).

- **Magnitude**: the power, and therefore the damage.
- Flags: **No Damage** (1), **Repeatable** (2), **No Fireball** (4), **No Smoke** (8), **No Decal** (16), **No Sparks** (32).

> **Warning:** without the **Repeatable** flag, the explosion happens only once and the entity disappears. Its damage also hits players: test the Magnitude in game before publishing the map. Tick No Damage for a purely visual effect.

Bombardments: `func_mortar_field`

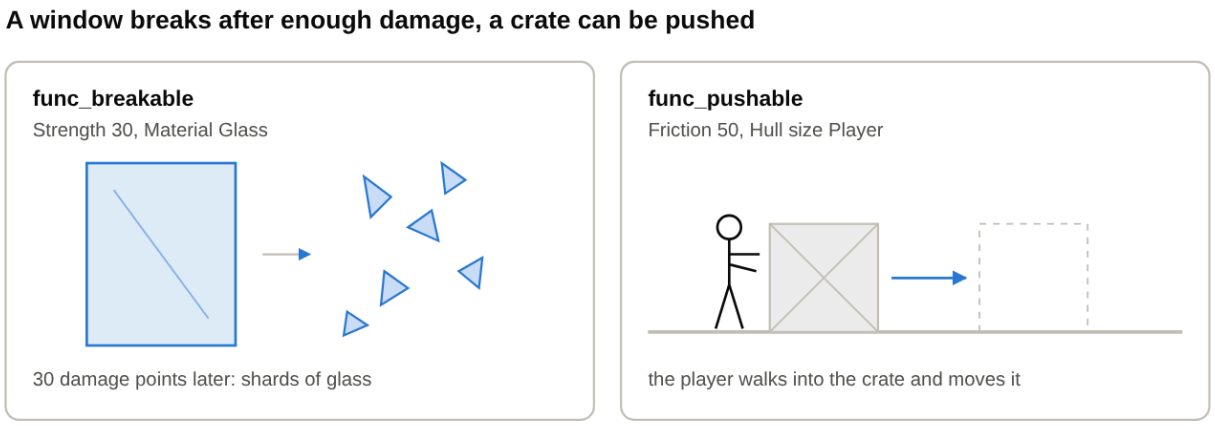
A brush entity that drops shells at random over its area when called. The number of shells and their spread are set in its properties.

Example: an exploding wall

1. Build a cracked section of wall and make it a `func_breakable` named `broken_wall` (next chapter), **Only Trigger** flag.
2. In front of it, place an `env_explosion` named `boom`, Magnitude 100, No Damage flag.
3. Add an `env_shake` named `quake`, Amplitude 8, Duration 1, radius 512.
4. A `multi_manager` with the keys `boom` = 0, `quake` = 0, `broken_wall` = 0.1 fires them together.
5. A button or `trigger_once` has the `multi_manager` as its Target.

Sources: VDC: env_explosion (GoldSrc), VDC: env_shake, VDC: List of Counter-Strike entities.

11. Breaking and pushing



breakable and pushable objects

Breakable objects

Entity: `func_breakable` · **Type:** brush entity · **Difficulty:** easy

Windows, crates, vent grates, weak walls: anything that breaks is a `func_breakable` (VDC: func_breakable (GoldSrc)).

Attribute	Role
Strength	Damage it takes before breaking. **0** = unbreakable
Material Type	The sound and look of the debris
Gibs Direction	Random debris (0) or thrown in the direction of the shot (1)
Spawn on Break	An item that appears in its place
Explode Magnitude	An explosion when it breaks. 0 = none
Gib Model	A custom model for the debris

Materials: **Glass** (0), **Wood** (1), **Metal** (2), **Flesh** (3), **Cinder Block** (4), **Ceiling Tile** (5), **Computer** (6), **Unbreakable Glass** (7), **Rocks** (8).

Flag	Role
Only Trigger (1)	Breaks only when called (Name), not from bullets
Touch (2)	Breaks when touched
Pressure (4)	Breaks when stood on (collapsing floor)
Instant Crowbar (256)	Breaks on the first melee hit

> **Counter-Strike quirk:** in CS 1.6, a grenade that hits a `func_breakable` can break it, whatever its material. According to mappers, the object needs a Render Mode other than Normal for some behaviours to work; Solid with FX Amount 255 keeps a normal look (TWHL comments: func_breakable).

> **Note:** the `func_breakable` sends a call to its **Target** when it breaks. That is how the `trigger_counter` in chapter 5 works.

Pushable objects

Entity: `func_pushable` · **Type:** brush entity · **Difficulty:** medium

A crate the player moves by walking into it (TWHL: func_pushable).

Attribute	Role
Hull size	The collision hull: Point (16 × 16 × 16), Player (32 × 32 × 72), Big (64 × 64 × 72), Player duck (32 × 32 × 36)
Friction	Resistance, from 0 to 400 (400 = heaviest)
Buoyancy	How it floats in water
Strength	Damage before breaking, if the **Breakable** flag is ticked

> **Warning:** a `func_pushable`'s collision is an axis-aligned box, not the exact shape of its brushes. A crate smaller than 32 × 32 behaves like a 32 crate, a larger one like a 64 crate. Make square crates of 32 or 64 to avoid surprises (TWHL).

Sources: VDC: func_breakable (GoldSrc), TWHL: func_breakable, TWHL: func_pushable.

12. Sound effects

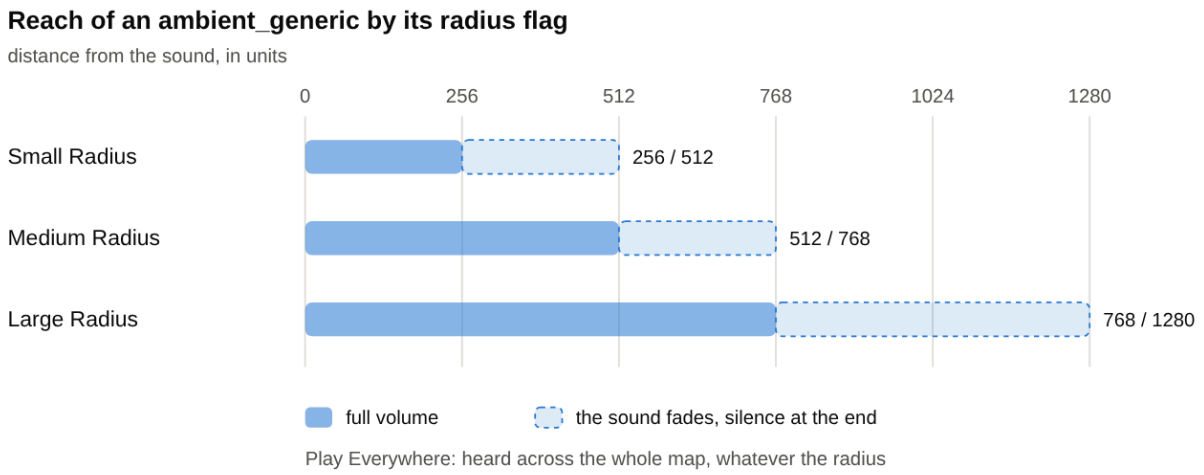
Entities: `ambient_generic`, `env_sound` · **Type:** point entity · **Difficulty:** easy

A machine hum, the wind, an alarm: sound does a lot for a map's atmosphere.

Play a sound: `ambient_generic`

Attribute	Role
WAV Name	The sound's path from the `sound` folder (`ambience/drips.wav`)
Volume	From 0 (silent) to 10 (maximum)
Pitch	The sound's pitch. 100 = normal
Fade in / Fade out	Gradual rise and fall of the volume
Dynamic Presets	Ready-made effects (machine, oscillation…) that override the other settings

Values and flags from TWHL: ambient_generic.



ambient_generic · reach of the 3 radii (TWHL)

Flag	Effect
Play Everywhere (1)	Heard across the whole map
Small / Medium / Large Radius (2, 4, 8)	The sound's reach (diagram above). If several are ticked, the smallest wins
Start Silent (16)	Does not play at start: it must be called
Not Toggled (32)	Replays on each call instead of switching on and off; for short sounds

> **Note:** a sound does not loop because you ask the entity to: it loops only if the .wav file contains **cue points** (loop markers). Without them, it plays once (TWHL).

Example: an alarm set off by a button. An `ambient_generic` named `alarm`, sound `ambience/alarm1.wav`, flags **Start Silent** and **Medium Radius**. The button has `alarm` as its Target: one press starts the alarm, a second stops it.

Change the acoustics: `env_sound`

When the player enters its radius (and can see it), the acoustics change: echo, reverb (VDC: env_sound (GoldSrc)).

Room Type	Effect
0	Normal, no effect
1	Generic
2 to 4	Metal (small, medium, large)
5 to 7	Tunnel
8 to 10	Chamber
11 to 13	Bright room
14 to 16	Underwater
17 to 19	Concrete
20 to 22	Very large space
23 to 25	Cave
26 to 28	Weird effects

> **Tip:** place an `env_sound` at the entrance of each type of place (a tunnel, a hangar) and another set to 0 at the exit, to return to normal sound.

Use your own sounds

Store your files in a subfolder of `cstrike/sound` (e.g. `sound/my_map/`) and put that path in WAV Name. As with textures, these files must be shipped with the map: see chapter 19, "Sprites and models", for the package.

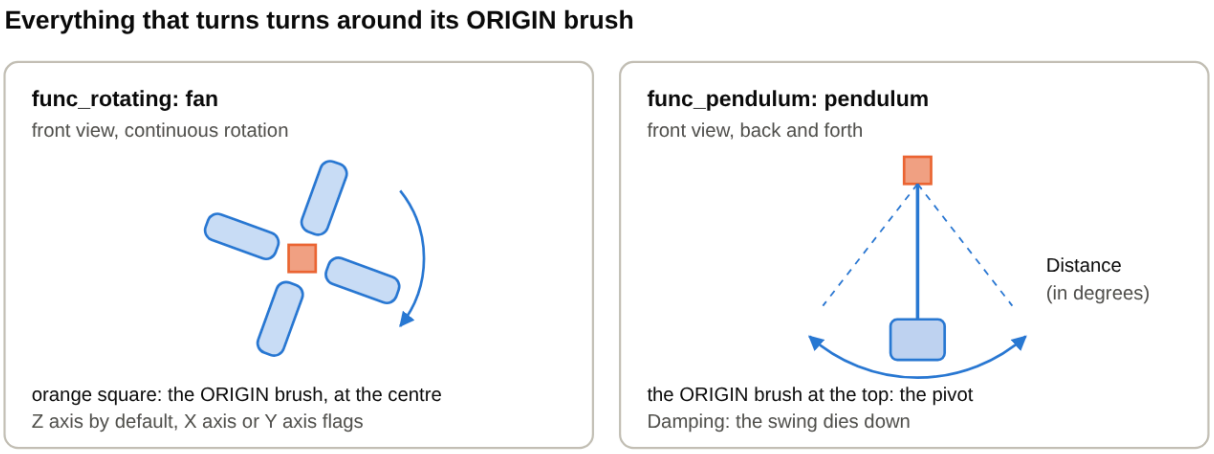
Footstep and impact sounds tied to textures are covered in chapter 6 of Part 1.

Sources: TWHL: ambient_generic, VDC: env_sound (GoldSrc).

13. Rotations

Entities: `func_rotating`, `func_pendulum` · **Type:** brush entity · **Difficulty:** medium

Fans, propellers, wheels, pendulums: like rotating doors, these entities need an **ORIGIN** brush tied with them, placed on the rotation axis.



rotations · fan and pendulum

Rotating objects: `func_rotating`

An object that spins continuously (VDC: func_rotating (GoldSrc)).

1. Build the object (the blades of a fan).
2. Add a small **ORIGIN** brush at its centre.
3. Select everything, Ctrl+T, `func_rotating`.

Attribute	Role
Rotation Speed	Maximum speed
Friction	From 0 to 100%: the lower the value, the longer it takes to start and stop (with the Acc/Dcc flag)
Damage when blocked	Damage dealt to whoever blocks it
Fan Sounds / WAV Name	The sound of the movement
Volume	From 0 to 10

Flag	Role
Start ON (1)	Spins from the start
Reverse Direction (2)	Spins the other way
X Axis (4) / Y Axis (8)	Changes the axis; by default it spins around the vertical axis
Acc/Dcc (16)	Speeds up and slows down gradually
Fan Pain (32)	Hurts on contact
Not Solid (64)	Can be walked through

> **Example:** a ceiling fan in a room. Blades as a `func_rotating`, flags **Start ON** and **Not Solid**, Rotation Speed 200. To turn it on with a switch, remove Start ON, give it a Name, and have a `func_button` target it.

Swinging objects: `func_pendulum`

An object that swings from side to side: chandelier, sign, pendulum (TWHL: func_pendulum).

Attribute	Role
Speed	Swing speed
Distance	The amplitude, in degrees. A negative value reverses the direction
Damping	From 0 to 1000: the swing dies down until it stops mid-travel
Damage when blocked	Damage dealt to whoever blocks it

Flags: **Start on** (1), **Passable** (8), **Auto-return** (16, returns to its start position when called), **X-axis** (64) and **Y-axis** (128).

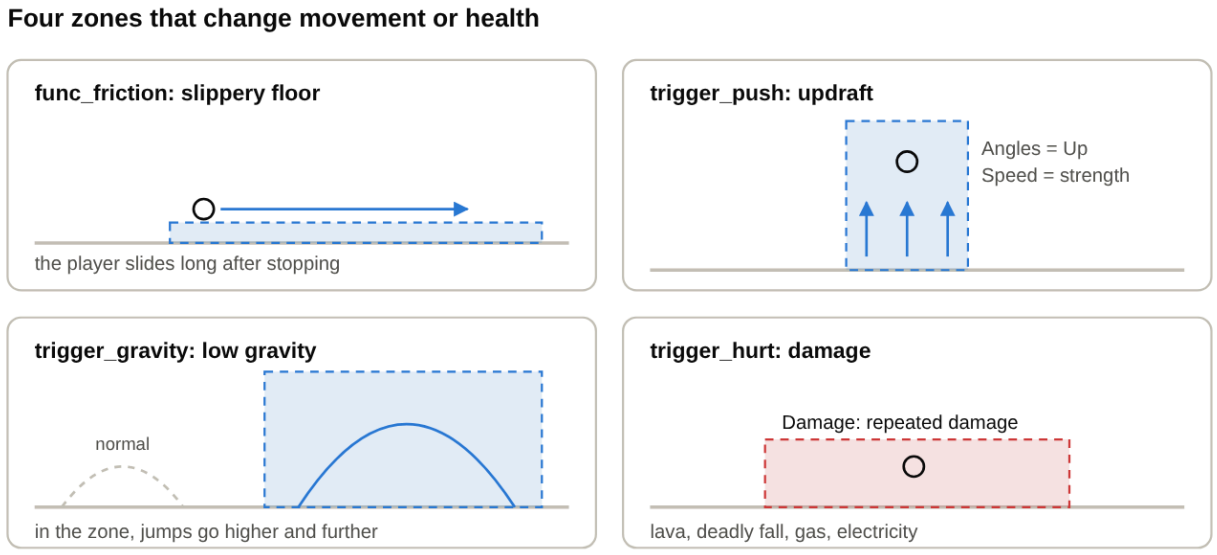
> **Trick:** placed very far from the object, the ORIGIN brush turns the swing into an almost straight back-and-forth: a small rotation around a distant point moves the object a lot. TWHL gives the formula to work out the angle (TWHL).

Sources: VDC: func_rotating (GoldSrc), TWHL: func_pendulum.

14. Hazardous environments

Entities: `func_friction`, `trigger_push`, `trigger_gravity`, `trigger_hurt` · **Type:** brush entity · **Difficulty:** easy

These four entities are volumes that change what happens to the player when they enter.



special zones · friction, push, gravity, damage

Slippery zone: `func_friction`

A brush laid on (or forming) the floor that reduces grip: ice, oil. The **Percentage of standard** attribute (`modifier`) sets the friction: 100 = normal, lower = more slippery. It is

listed among the Counter-Strike entities (VDC: List of CS entities).

Wind zone: `trigger_push`

Pushes everything that touches it, in the direction of its **Angles** (Up for a vertical updraft) at **Speed** (VDC: trigger_push (GoldSrc)).

Flag	Effect
Once Only (1)	Pushes only once
Start Off (2)	Starts off: it must be called to activate
Pushables (4)	Also pushes `func_pushable`s

Example: a vertical air shaft. A `trigger_push` the height of the shaft, Angles = Up, Speed 400, lifts the player to the top.

Change gravity: `trigger_gravity`

Changes the gravity of the player passing through. The **Gravity** attribute is a multiplier: 1 = normal gravity, 0.5 = half as strong.

> **Warning:** the gravity change stays on the player after they leave the zone. Place a second `trigger_gravity` set to 1 at the exit to restore normal gravity.

Hurt: `trigger_hurt`

Deals damage to whoever is inside (VDC: trigger_hurt (GoldSrc)).

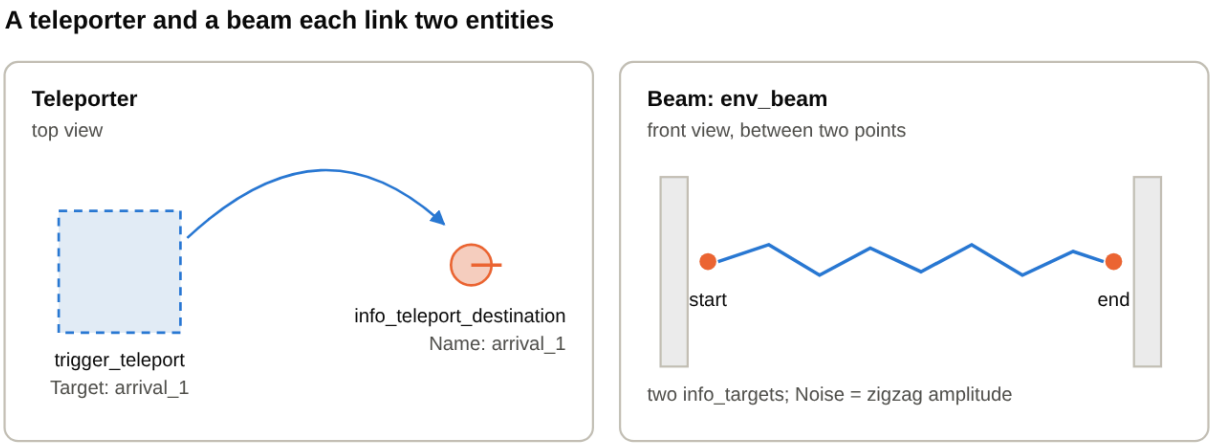
Attribute	Role
Damage	The damage dealt, repeatedly for as long as you stay inside
Damage Type	The type: Burn (8), Shock (256), Poison (131072), Radiation (262144)… It changes the icon and the on-screen effect
Name	To enable and disable it

Flags: **Target Once** (1), **Start Off** (2), **No Clients** (8, does not affect players), **FireClientOnly** (16), **TouchClientOnly** (32).

Example: a bottomless pit. A `trigger_hurt` at the bottom, Damage 1000: a falling player dies on arrival, without waiting for fall damage.

Sources: VDC: trigger_push (GoldSrc), VDC: trigger_hurt (GoldSrc), VDC: List of Counter-Strike entities.

15. Teleportation and visual effects



teleporter and beam · two linked entities

Teleportation

Entities: `trigger_teleport`, `info_teleport_destination` · **Type:** brush entity + point entity · **Difficulty:** easy

You need two entities: the departure volume and the arrival point (TWHL: Teleporters).

1. Create a brush where the player should vanish, textured AAATRIGGER, and make it a `trigger_teleport`.
2. Give it a **Target**: `arrival_1`.
3. Place an `info_teleport_destination` at the arrival point and name it `arrival_1`. Its **angle** gives the direction the player will face.

> **Warning:** place the destination slightly above the floor and away from walls and ceiling, or the player arrives stuck in the scenery.

> **Note:** if your FGD does not offer `info_teleport_destination`, an `info_target` works the same way (TWHL).

Button-activated teleporter: give the `trigger_teleport` a **Master** = `tele_master`, create a `multisource` named `tele_master`, and a `func_button` with Target = `tele_master` and Delay before reset = 5. The teleporter works for 5 seconds after the press (TWHL).

Sprites: `env_sprite`

An animated image (.spr file) placed in the map: flame, lamp glow, smoke (TWHL: env_sprite).

Attribute	Role
Sprite Name	The file path (`sprites/glow01.spr`)
Framerate	Animation frames per second
Scale	The size

Attribute	Role
Render Mode	**Glow** (grows and fades with distance), **Additive** (black becomes transparent), **Texture**…
FX Amount / FX Color	Transparency and tint

Flags: **Start on** (visible from the start) and **Play Once** (the animation plays once per call).

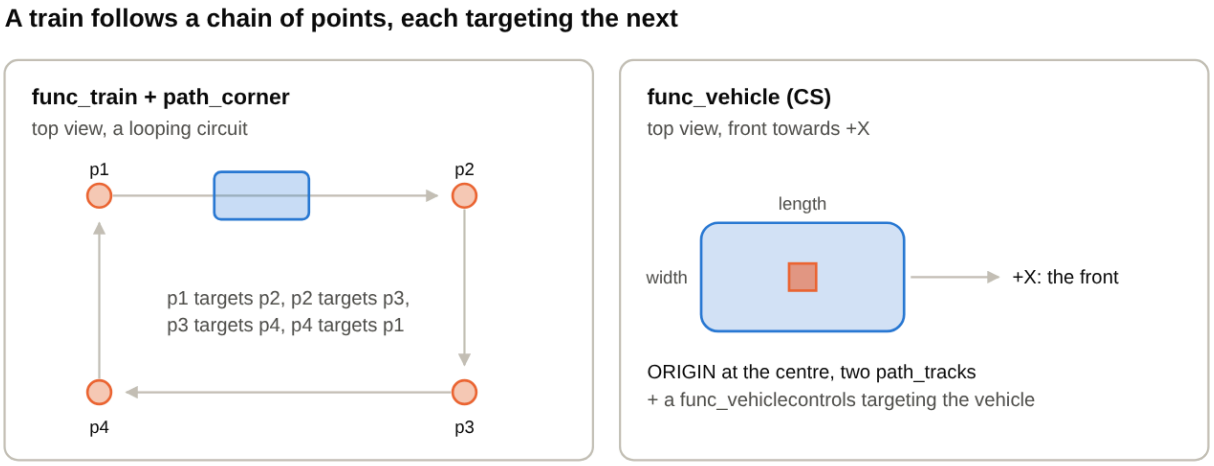
Glows, beams and lasers

Entity	Role	Key attributes
`env_glow`	A fixed glowing halo	Sprite, colour, intensity
`env_beam`	A beam between two entities (`info_target`s)	Start Entity, Ending Entity, width, Noise (zigzag), lifetime, colour
`env_laser`	A laser from the entity towards a target	Target (an `info_target`), sprite, width, damage

Example: an electric arc between two terminals. Two `info_target`s named `terminal_a` and `terminal_b`, an `env_beam` with Start = `terminal_a`, End = `terminal_b`, Noise 20, light blue colour, **Start On** flag.

Sources: TWHL: Teleporters, TWHL: env_sprite, VDC: List of Counter-Strike entities.

16. Means of transport



transport · train circuit and CS vehicle

Simple trains: `func_train`

Entities: `func_train`, `path_corner` · **Type:** brush entity + point entities · **Difficulty:** medium

An object that moves in straight lines from point to point: moving platform, wagon, scenery helicopter (TWHL: func_train).

1. Build the object, add an **ORIGIN** brush at its centre, and make the whole thing a `func_train`, with a Name.
2. Place a first `path_corner`, named `p1`.
3. In the train, **First stop target** = `p1`: on load, the train appears at this point, wherever it is in the editor.
4. Add more `path_corner`s; each has the next one's name as its **Next stop target**. For a loop, the last one targets the first.
5. On a `path_corner`, the **Wait for retrigger** box stops the train until the next call; the **Wait** attribute makes it pause for a few seconds.

> **Warning:** a `func_train` does not turn: it slides from point to point keeping its orientation. And its lighting stays that of its position in the editor; build it somewhere lit like its route (TWHL).

Trains on tracks: `func_tracktrain`

Entities: `func_tracktrain`, `path_track`, `func_traincontrols` · **Difficulty:** hard

Unlike `func_train`, it **turns** in bends and can be driven.

- `path_track`s form the tracks: each has a **Next stop target** and can have a **Branch path** (a switch).
- Called with no Branch path, a `path_track` opens or closes the way; with a Branch path, it flips the switch (TWHL: path_track).
- A `func_traincontrols` brush (Target = the train) defines the area where the player can drive it with the Use key.
- `func_trackchange` and `func_trackautochange` are moving track pieces (a lift or turntable for trains).

CS cars: `func_vehicle`

Entities: `func_vehicle`, `func_vehiclecontrols`, `path_track` · **Difficulty:** hard

Counter-Strike adds a vehicle that players can drive (the303: func_vehicle).

1. Build the vehicle with **its front facing +X** in the editor, and an ORIGIN brush at the centre (in X and Y).
2. Add CLIP brushes for collision, then make the whole thing a `func_vehicle` named `car`.
3. Set **length** (size in X), **width** (size in Y), **height** (from the bottom of the ORIGIN brush to the ground), **speed**, **sounds**.
4. Place two `path_track`s: the vehicle targets the first (its spawn point), the first targets the second, the second targets the first.
5. Create a `func_vehiclecontrols` brush with Target = `car`: the player drives by pressing Use.

> **Warning:** `func_vehicle` is temperamental: it can get stuck or lost. Never make victory depend on a vehicle, and check that it cannot block a required passage (the303).

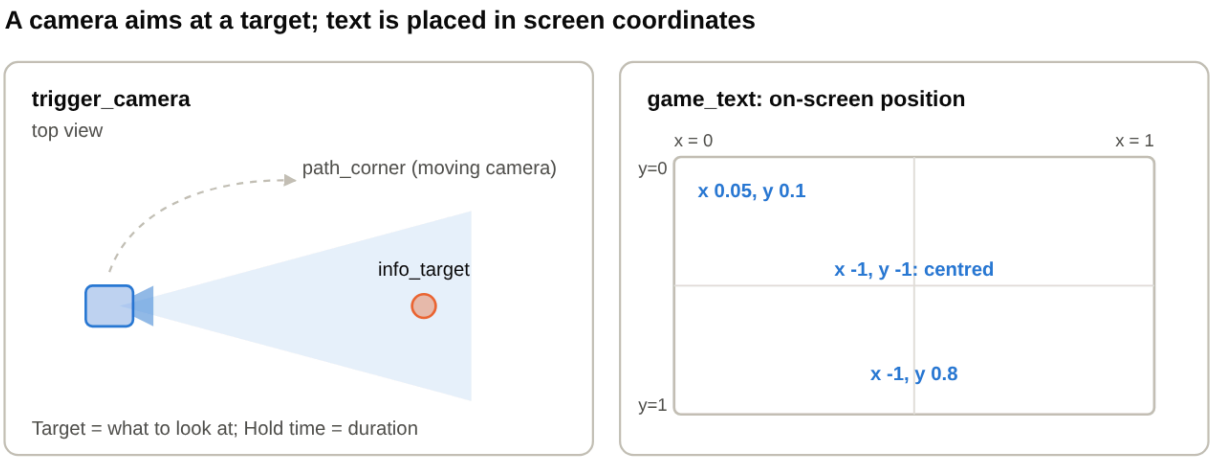
Conveyor belts: `func_conveyor`

A brush whose surface carries whatever is on it. Give its top face a texture starting with `scroll`: it will scroll at the belt's speed (VDC: Texture prefixes). Direction is set with the angle, speed with **Conveyor Speed**.

Sources: TWHL: func_train, TWHL: path_track, the303: func_vehicle, VDC: Texture prefixes.

17. Like at the movies

Entities: `trigger_camera`, `env_fade`, `game_text` · **Type:** point entity · **Difficulty:** medium



camera and text · aimed target and screen coordinates

Cameras: `trigger_camera`

Once called, the camera takes over the player's view for a set time (TWHL: trigger_camera).

Attribute	Role
Target	The entity the camera looks at (an `info_target`, or a train to follow a moving object)
Hold time	How long the view lasts, in seconds
Path Corner	The first `path_corner`, for a moving camera
Initial Speed, Acceleration, Stop Deceleration	The moving camera's motion

Flags: **Start At Player** (1, starts from the player's view), **Follow Player** (2, looks at the player), **Freeze Player** (4, the player cannot move during the sequence).

> **Counter-Strike quirk:** `trigger_camera`s are also used for the intro views you see when joining a server, while choosing a team (TWHL). Place a few, each aimed at an `info_target`, to show off your map.

> **Warning:** the engine only draws the camera's view if the player is close enough to it. A camera at the other end of the map may show empty areas (TWHL).

Fades: `env_fade`

Covers the screen with a colour, then removes it. Set the fade **duration**, the **hold** time, the **colour** and its opacity (FX Amount). The **Fade From** flag goes the other way (from colour to image), **Activator Only** limits the effect to the player who triggered it.

Example: a half-second white flash at the moment of an explosion, by adding a white `env_fade` to the `multi_manager` from chapter 10.

Show text: `game_text`

An on-screen message, at the position and with the effect you choose (VDC: game_text (GoldSrc)).

Attribute	Role
Message	The text
X / Y	Position, from 0 to 1 (left to right, top to bottom). **-1 centres the text**
Text Effect	0: fade; 1: credits; 2: letters appear one by one
Color1 / Color2	The text colour, and that of the current letter with effect 2
Fade in / Hold / Fade out	Appearance, display time, disappearance
Channel	From 0 to 4: a new message on the same channel replaces the old one

The **All Players** flag shows the message to everyone; without it, only the triggering player sees it.

Example: a welcome message. A `game_text` with Message = "Welcome! Defend site A", X = -1, Y = 0.3, Hold 4, All Players flag, called by a `trigger_multiple` in the spawn area.

Sources: TWHL: trigger_camera, VDC: game_text (GoldSrc).

18. Weapons and equipment

Entities: `armoury_entity`, `game_player_equip`, `player_weaponstrip` · **Type:** point entity · **Difficulty:** easy to medium

In Counter-Strike you normally buy your weapons. But a map can also place them on the ground, give them to every player or take them away (TWHL: Stripping and Equipping Weapons in CS).

Weapons on the ground: `armoury_entity`

Places a weapon or item to pick up. Choose the item in the **Item** list and the number of copies in **Count**.

- Pistols are not offered.
- Ammo is not full.
- Do not place it inside the floor or on a start point.

Equip every player: `game_player_equip`



equipment · game_player_equip at spawn

1. Place a `game_player_equip`, **with no Name**, **Use only flag unticked**.
2. Turn SmartEdit off and add one key per item, with the quantity as the value.
3. On every spawn, all players receive exactly this list.

> **Warning:** as soon as a `game_player_equip` is in the map, players spawn **with no weapons at all**, even if the entity has the Use only flag. Remember to add `weapon_knife` = 1, or they will not even have a knife (TWHL).

Key	Item
`weapon_ak47`, `weapon_m4a1`, `weapon_awp`, `weapon_deagle`…	Weapons (same name as in game)
`ammo_762nato`, `ammo_556nato`, `ammo_50ae`, `ammo_buckshot`…	Magazines
`item_kevlar`	Vest
`item_assaultsuit`	Vest + helmet
`item_thighpack`	Defuse kit
`weapon_hegrenade`, `weapon_flashbang`, `weapon_smokegrenade`	Grenades

> **Common mistake:** do not give `weapon_c4` with this entity: it does not create a real bomb (TWHL).

With a Name and the Use only flag ticked, the entity gives its items only to the player who triggers it: useful for a weapon locker opened by a button.

Prevent buying

For a map where everyone has the same weapons, you must also disable buying. Two methods, to combine:

- a small out-of-reach `func_buyzone` with no team: it removes the automatic buy zones around the starts;
- an `info_map_parameters` set so that nobody can buy (chapter 20).

Take weapons away: `player_weaponstrip`

Removes all weapons from the player who triggers it. Name it (`strip`), then create a `trigger_multiple` with Target = `strip`.

> **Warning:** it only acts on the triggering player: a `trigger_auto` or a distant door strips nothing. And do not use it on a de_ map: it also removes the bomb (TWHL).

Source: TWHL: Stripping and Equipping Weapons in Counter-Strike.

19. Sprites and models

Entities: `env_sprite`, `cycler_sprite` · **Type:** point entity · **Difficulty:** medium

Sprites

A sprite (.spr) is a flat image, often animated, that always faces the player: flame, halo, smoke, spark. It is placed with `env_sprite`, covered in chapter 15. The game includes many in `valve/sprites` and `cstrike/sprites`.

Models

A model (.mdl) is a 3D object: a plant, a scenery vehicle, a piece of furniture. It often looks better and costs less than an object made of brushes.

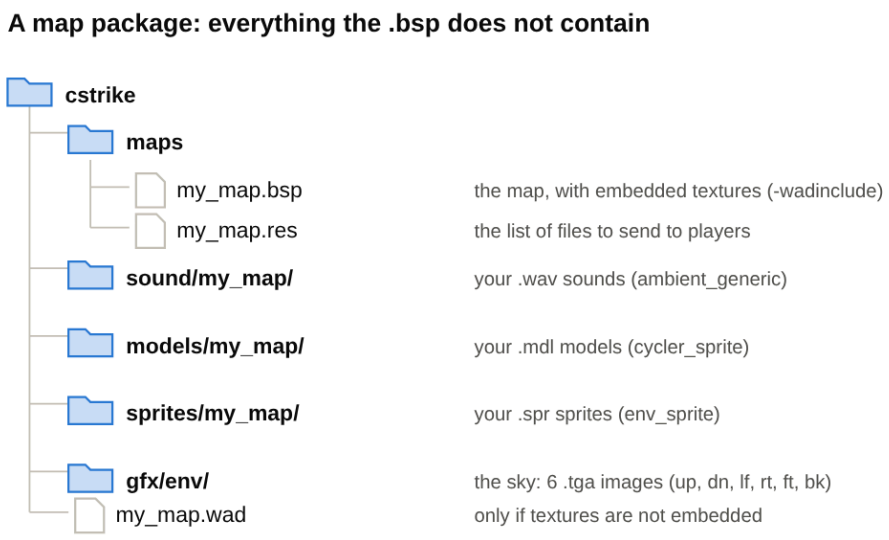
In Counter-Strike, the entity most used to place a decorative model is `cycler_sprite` (TWHL: cycler_sprite):

1. Place a `cycler_sprite`.
2. In **Sprite** (key `model`), give the model's path: `models/my_map/plant.mdl`.
3. Counter-Strike adds the **body**, **skin** and **sequence** keys, to choose a variant or an animation.
4. Orient it with the angles.

> **Warning:** a model does not always block players or bullets. Surround it with a **CLIP** brush if the player must not walk through. And a `cycler_sprite` that gets hit changes animation frame: for static scenery, pick a sequence with no movement (TWHL).

> **Note:** J.A.C.K. shows models and sprites directly in the 3D view, and can open them in its viewer (VDC: J.A.C.K.).

The package



map package · folders and files to ship

A .bsp can only contain WAD textures. Everything else (sounds, models, sprites, custom sky) stays alongside, and the server must know what to send players. That is the job of the **.res** file: a list of paths, one file per line, stored next to the .bsp (VDC: Resource list (GoldSrc)).

Example of `my_map.res`:

```
sound/my_map/alarm.wav
models/my_map/plant.mdl
sprites/my_map/flame.spr
gfx/env/mysky_up.tga
gfx/env/mysky_dn.tga
```

(All six sky images must be listed; only two are shown here.)

To generate it: **ResGuy**, a recent tool made for Sven Co-op but compatible with all GoldSrc games, or **RESGen**, older (2005), whose list you should double-check (VDC).

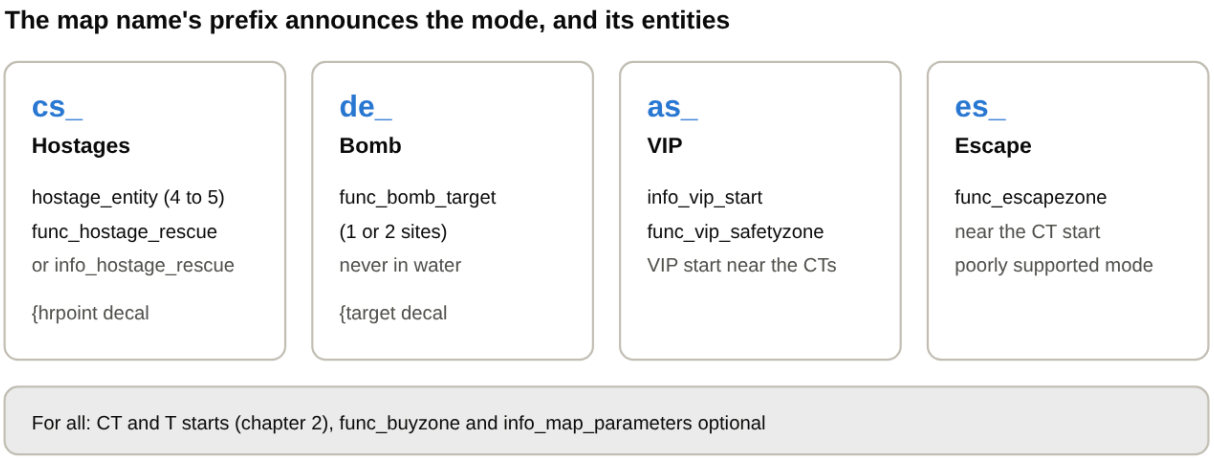
> **Tip:** store all your custom files in subfolders named after the map (`sound/my_map/`, `models/my_map/`). You avoid overwriting another map's files with the same name.

Sources: TWHL: cycler_sprite, VDC: Resource list (GoldSrc), VDC: J.A.C.K..

20. Counter-Strike game modes

Entities: `hostage_entity`, `func_hostage_rescue`, `func_bomb_target`, `info_vip_start`, `func_vip_safetyzone`, `func_escapezone`, `func_buyzone`, `info_map_parameters` · **Difficulty:** easy

The prefix of the map's name announces its game mode. The game itself works out the mode from the entities present (TWHL: Introduction to Counter-Strike mapping).



game modes · 4 official prefixes

cs_ maps: hostage rescue

- Place 4 or 5 `hostage_entity`s, usually near the Terrorist start.
- The rescue zone: a `func_hostage_rescue` (brush, any size) or an `info_hostage_rescue` (point, radius of about 128 units). Use only one of the two: if both exist, the game ignores the second.
- Place the zone away from walls, or a hostage could be rescued through a wall. The `{hrpoint` decal marks it for players.

de_ maps: the bomb

- At least one `func_bomb_target`: its size defines where the bomb can be planted. Most maps have two, sites A and B.
- Alternative: `info_bomb_target`, with a fixed-size zone.
- The `{target` (or `{target2`) decal marks the site.

> **Warning:** never put a bomb site in or right next to water. A bomb planted in water does not explode and the round does not end (TWHL).

as_ maps: the VIP

- An `info_vip_start`, in addition to the normal starts, near the Counter-Terrorists.
- A `func_vip_safetyzone`: the zone the VIP must reach.

es_ maps: escape

A `func_escapezone`, near the CT start, which the Terrorists must reach. This mode is poorly supported by recent versions of CS (TWHL).

Buy zones: `func_buyzone`

- **Without** a `func_buyzone`, each start point creates a small buy zone around itself.
- **With** a `func_buyzone`, those automatic zones disappear: the brush decides. Set its **Team** and provide one per team.

- To remove buying, hide a small `func_buyzone` out of reach.

Map rules: `info_map_parameters`

A point entity that sets who can buy (everyone, CT only, T only, nobody) and the bomb's blast radius (**bombradius**, 500 by default). These values come from the entity's VDC page, shared by the whole Counter-Strike series: check them in your CS 1.6 FGD.

Community prefixes

Outside the official modes, players recognise unofficial prefixes (TWHL): `fy_` (fight yard, a small arena with weapons on the ground), `aim_`, `awp_`, `ka_` (knife arena), `35hp_` and `1hp_` (reduced health), `kz_` (climbing), `surf_` (sliding on ramps).

Sources: TWHL: Introduction to Counter-Strike mapping, VDC: info_map_parameters.

Part 3: Compilation and optimisation

Turn your work into a playable map, then make it run smoothly.

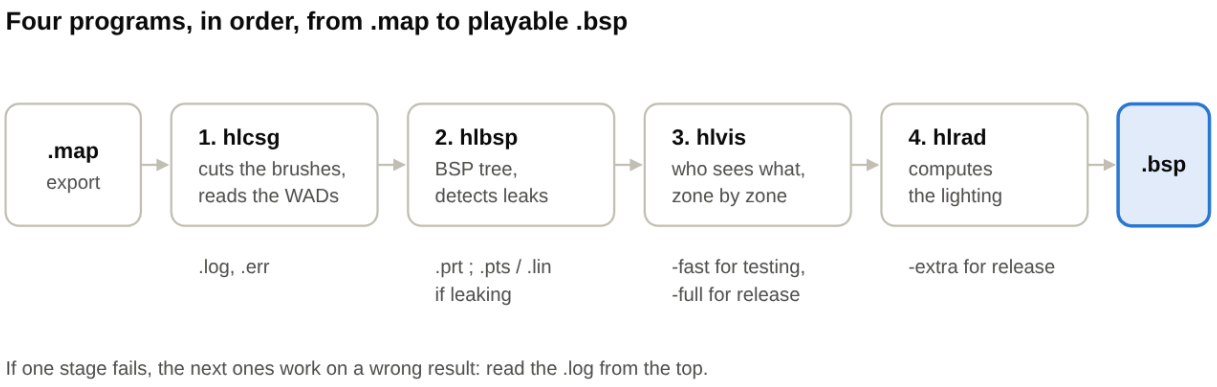
1. Compilation (A: testing your map)

Your map cannot be played as it is: the J.A.C.K. file must be turned into a .bsp, the format the game reads. That is compilation.

File formats

File	Role
.jmf (or .rmf)	Your working file in the editor
.map	The text version, exported for the compilers (Valve220 format in J.A.C.K.)
.bsp	The compiled, playable map
.log / .err	The compile report, and the errors
.pts / .lin	The trace of a leak (pointfile)
.prt	The portals, used by VIS; J.A.C.K. can display them

How it works



compile chain · CSG, BSP, VIS, RAD

Four programs run in sequence (TWHL: Compiling 101):

1. **hlcsg** cuts overlapping brushes, builds the collision hulls and lists the textures used.
2. **hlbsp** builds the BSP tree, the split of the map into volumes. It is the one that detects leaks.
3. **hlvis** works out, for each zone, which other zones can be seen. Without it, the game draws the whole map all the time.
4. **hlrad** computes the light and its bounces. Without it, the map is fully bright, with no shadows.

The recommended compilers are **VHLT** (TWHL: Compile).

Compile in J.A.C.K.

File > Run Map (F9).

- **Normal mode**: for each stage, choose **Normal**, **Fast** or **No**. Handy at first.
- **Expert mode**: you write each command and its parameters yourself, and you can add steps (copy the .bsp, launch the game). Use this mode as soon as you know the parameters.

J.A.C.K. compiles without blocking the editor: you can keep working during compilation (VDC: J.A.C.K.).

> **Note:** to launch the game straight into your map, add `-game cstrike +map my_map` to the game's command line. Add `-dev` and `-console` to see debug messages.

Useful parameters

Program	Parameter	Effect
all	`-chart`	Shows statistics and limits reached
all	`-estimate`	Shows time remaining
all	`-low`	Low priority: the PC stays usable
hlcsg	`-wadinclude name`	Embeds the textures of WAD `name` in the .bsp
hlcsg	`-nowadtextures`	Embeds all the textures used
hlcsg	`-cliptype precise`	More precise collisions in corners
hlbsp	`-leakonly`	Stops after the leak search
hlvis	`-fast`	Fast VIS, for testing
hlvis	`-full`	Precise VIS, for the release version
hlrad	`-extra`	Higher-quality lighting (slower)
hlrad	`-bounce n`	Number of light bounces

Parameter sources: VDC: HLBSP, TWHL: Tool Textures (for `-cliptype precise`), VDC: SirYodaJedi's compile settings.

Example settings

Stage	While building	For the release version
CSG	`-wadinclude my_map`	`-wadinclude my_map -cliptype precise`
BSP	(nothing)	`-chart`
VIS	`-fast`	`-full`
RAD	(nothing)	`-extra -chart`

With these settings, a small map compiles in a few seconds during testing. The final version takes longer, but looks better and runs more smoothly.

Sources: TWHL: Compiling 101, TWHL: Compile, VDC: HLBSP, VDC: J.A.C.K..

2. Compilation (B: debugging)

Your map will not load, it is pitch black or fully bright, or the game lags? The answer is almost always in the compile report.

When it fails: read the log

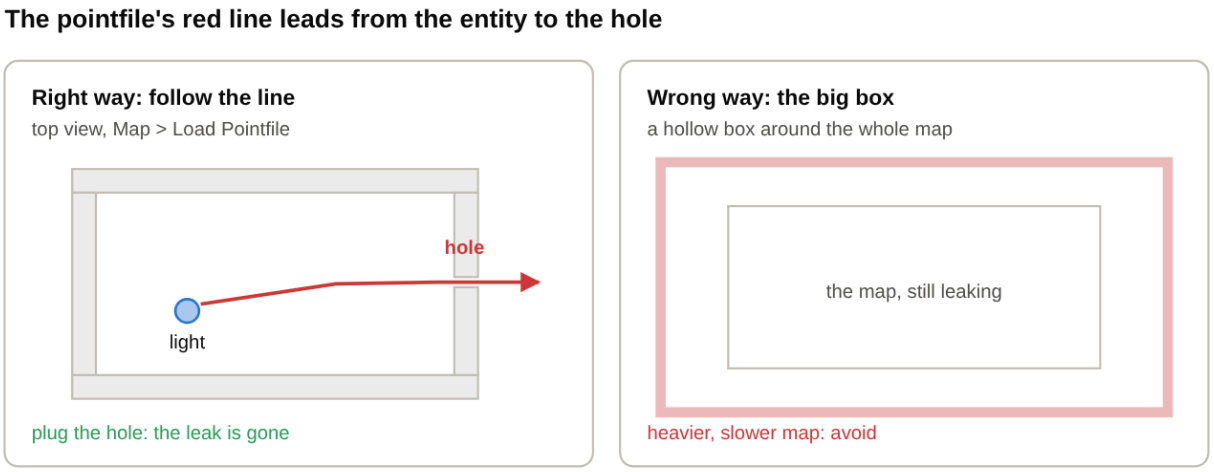
Every compile writes a `my_map.log` file next to the map (and `my_map.err` on error). Read it **from the top**: the first error often causes all the following ones (TWHL: TrenchBroom setup, compilation section).

Invalid brushes

A brush is invalid if it is not convex or if one of its faces is not planar, most often after vertex editing. The compiler "fixes" it its own way, which can open a hole (TWHL: How to fix those leaks).

1. **Map > Check for Problems** (Alt+P) lists invalid brushes and often offers to repair them.
2. If the log mentions a brush number, **Go to Brush Number** (Ctrl+Shift+G) takes you there.
3. For a twisted face, J.A.C.K.'s **triangulation** splits it into flat faces.

Leaks



leak · pointfile versus big box

A leak means an entity "sees" the void through a hole in the map. HLBSP can then no longer tell inside from outside: VIS and RAD do not work, the map is fully bright and slow.

The right method (TWHL: Pointfiles):

1. After a compile with a leak, open **Map > Load Pointfile**. In GoldSrc, choose the **.lin** file rather than .pts.
2. A red line appears in the views: it starts from an entity and goes through the hole. Follow it.
3. Plug the hole with a world brush, and recompile.

Common causes (TWHL):

Cause	Fix
A small gap between two brushes	Work on the grid, realign with Snap to Grid (Ctrl+B)
An outer wall turned into an entity (`func_wall`, door)	Entities do not seal the map: put a world brush back behind it
An entity outside the map	Only its origin point counts: bring it back inside
An ORIGIN brush sticking out into the void	Move the ORIGIN brush inside
An invalid brush	Repair it (see above)

> **Common mistake:** "fixing" a leak by putting the whole map inside a big hollow box. The leak is hidden, not fixed: the map gets heavier, slower to compile and slower in game (TWHL).

> **Note:** the compiler reports only one leak at a time. Recompile after each fix. To go fast, `-leakonly` on hlbsp stops as soon as the leak search is done (VDC: HLBSP).

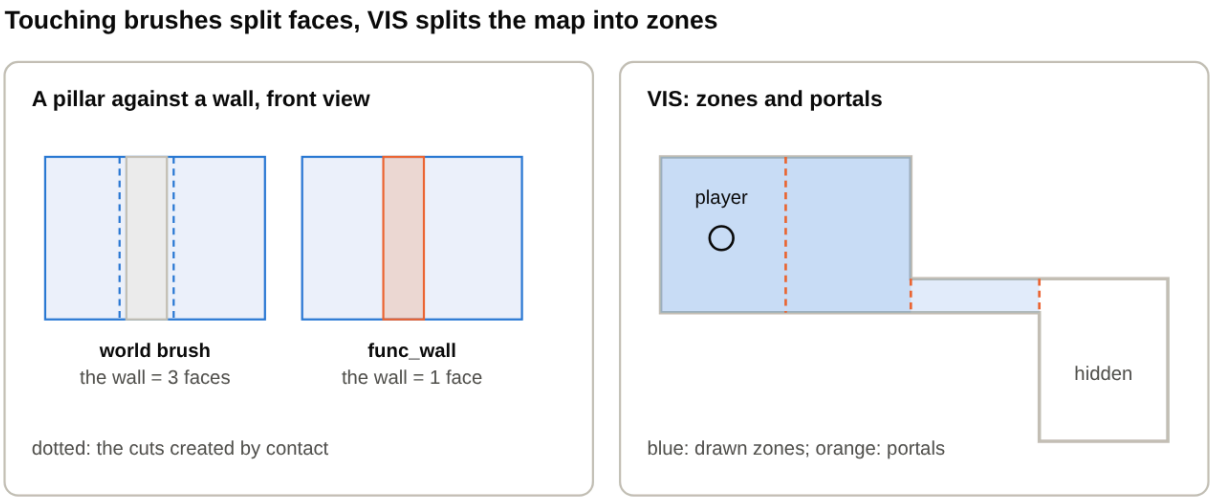
Black or missing sky

If the sky displays badly, check that the sky name in **Map > Map Properties** (worldspawn `skyname` key) matches six images present in `gfx/env/`. A custom sky must also be listed in the .res file (Part 2, chapter 19).

Sources: TWHL: Pointfiles, TWHL: How to fix those leaks, VDC: HLBSP.

3. Optimisation (A: the laws of VIS)

A smooth map is not an empty map: it is a map where the engine only draws what is useful. To help it, you need to understand how it works.



optimisation · face splitting and VIS zones

Polygon splitting

The engine does not draw brushes but **faces** (polygons). The more there are on screen, the harder the graphics card works. Two things multiply faces (TWHL: Total Map Optimisation):

- **World brushes that touch** split each other: a pillar against a wall cuts the wall into several faces. Making the pillar a `func_wall` (or a `func_detail` with VHLT) keeps the wall in one piece. In TWHL's example, this habit alone saved nearly 100 polygons.
- **Large faces** are split again every **240 texture pixels** or so, at scale 1. A texture at scale 2 is only cut every 480 pixels.

What the engine does

1. **hlbsp** splits the map's empty space into convex volumes, the **visleafs** (zones).
2. Between two neighbouring zones lies a **portal**, an opening you can see through.
3. **hlvis** works out, for each zone, the list of zones visible through the portals.
4. In game, the engine only draws the zones visible from the one the player is in, even if part of them is hidden behind a corner.

That is why a straight corridor across the whole map is expensive: from one end you "see" everything to the other.

Measure: r_speeds

In the game console (TWHL):

```
developer 1
sv_cheats 1
r_speeds 1
map my_map
```

At the top left you will see:

- **wpoly**: world polygons (brushes) drawn;
- **epoly**: entity and model polygons (players, weapons, models).

Two commands complete the analysis: `gl_wireframe 1` shows the face splitting, `gl_wireframe 2` shows everything the engine draws, including behind walls.

> **Tip:** stand in the most open spots of the map, note the wpoly, then change and recompile. It is the only way to know whether an optimisation helped.

Source: TWHL: Total Map Optimisation Part 1.

4. Optimisation (B: lowering r_speeds)

You know how to measure. Here is how to bring the numbers down, from the most effective to the finest.

1. Structure your map well

No trick saves a bad layout (TWHL: Hint Brushes). Think about visibility from the start:

- **Indoors**: bent corridors rather than straight ones, rooms linked by narrow passages or corners. Every corner cuts the view.
- **Outdoors**: tall walls, buildings or terrain that separate the areas. A large open square where you can see everything is the worst case.

2. Avoid direct brush contact

Anything decorative that touches a wall (pillars, pipes, ledges, lamps) becomes a `func_wall`, or a `func_detail` with VHLT. The wall is no longer split, and VIS has less work (TWHL: Total Map Optimisation).

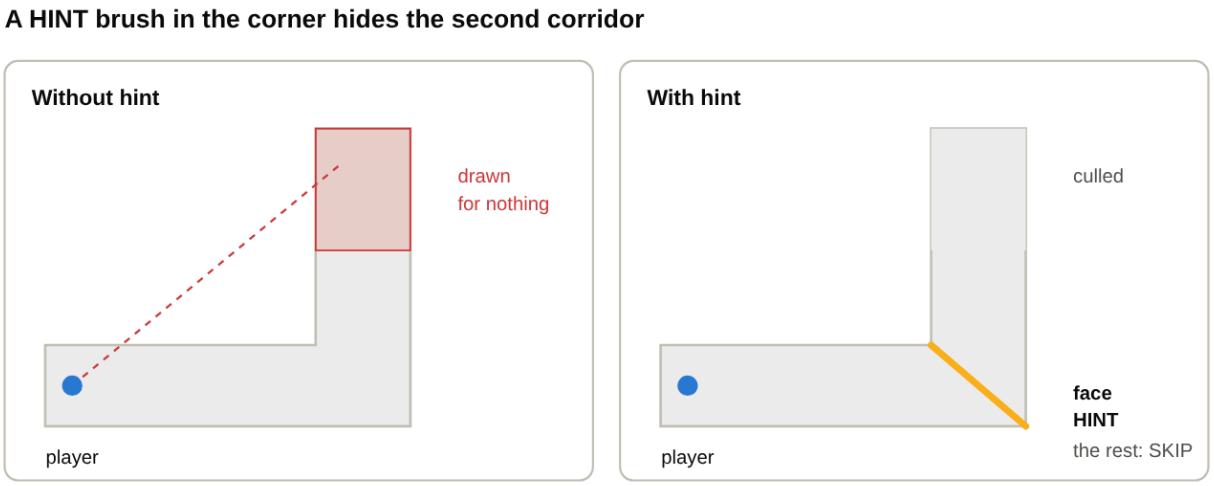
> **Note:** `func_detail` is a world brush (it casts shadows, it is solid) that VIS ignores (VDC: func_detail (GoldSrc)). It is the modern solution for details, provided you use VHLT.

> **Warning:** a `func_wall` or `func_detail` never seals the map. Do not convert an outer wall.

3. Take care of textures

- Put **NULL** on entity faces the player never sees (underside of a crate on the floor, back of a `func_wall` against a wall). The compiler already removes those of world brushes.
- A face is split every 240 texture pixels: a 240-pixel texture, or a slightly larger scale, avoids needless cuts (TWHL).
- Align textures between neighbouring brushes: an offset creates extra faces.

4. Hint brushes



hint brush · L-shaped corridor seen from above

A hint brush forces VIS to split its zones at a precise spot (TWHL: Hint Brushes).

1. Create a thin brush running diagonally across the corner, from the inner corner to the outer corner, and from floor to ceiling.

2. Texture the whole brush **SKIP**.
3. With the Texture Application tool, put **HINT** on just one of the large diagonal faces.
4. Compile with a full VIS (not `-fast`) and compare `r_speeds` at the same spot.

What works best, according to TWHL's tests:

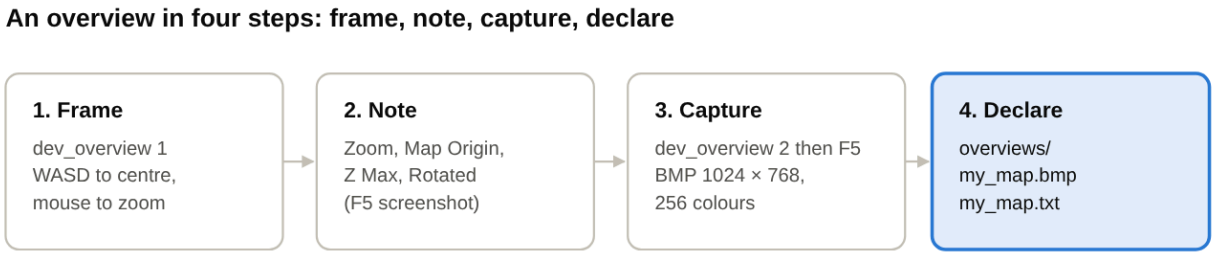
- **cutting corners**: a hint through the corner closest to the player, touching walls, floor and ceiling;
- **closing off** corridor junctions;
- **a horizontal hint** above head height, when walls do not reach the ceiling.

> **Warning:** badly placed, hints can **increase** r_speeds and lengthen compilation. Test each hint with `r_speeds 1` and `gl_wireframe 2`.

Sources: TWHL: Hint Brushes, TWHL: Total Map Optimisation, VDC: func_detail (GoldSrc).

5. Overviews

The overview is the top-down map shown in spectator mode and on the minimap. Without it, your map has no plan. It is made inside the game itself (TWHL: Creating Overviews).



overview · 4 steps

Capture the overview

1. In the game's video options, switch to **OpenGL** and **1024 × 768**. The command does not work in Software mode.
2. Launch your map. Hide the HUD with `hud_draw 0`, or move away from the buy zone.
3. In the console, type `dev_overview 1`. Centre the map with the movement keys, zoom with the mouse buttons.
4. Note the values shown at the top of the screen: **Zoom**, **Map Origin** (three numbers), **Z Min**, **Z Max**, **Rotated**. An F5 screenshot records them.
5. Type `dev_overview 2` to hide the numbers, then **F5**. If it is too dark, `r_fullbright 1` before the screenshot.

> **Note:** a TWHL commenter reports that the overview only displays correctly if the game is launched with `-dev` and you join the spectators before typing the command (TWHL).

Convert the overview

The screenshot appears in the game folder, under a name like `my_map0000.bmp`. Open it in an image editor, remove any green lines, then save it as **8-bit BMP, 256 colours, 1024 × 768**, named `my_map.bmp`, in `cstrike/overviews/`.

Edit the properties

Copy the .txt file of an official map (de_dust's, for example), rename it `my_map.txt` in the same folder, and replace the values with yours:

```
// overview of my_map

global
{
        ZOOM    1.20
        ORIGIN  -50.00  1124.00 -1000.00
        ROTATED 1
}

layer
{
        IMAGE   "overviews/my_map.bmp"
        HEIGHT  -1000.00
}
```

- **ZOOM**: the Zoom value you noted.
- **ORIGIN**: the three Map Origin values.
- **ROTATED**: the Rotated value.
- **IMAGE**: the path of your BMP.
- **HEIGHT**: the Z Max value.

The numbers above are only a format example: use the ones shown for your map.

> **Warning:** the overview only shows the top floor of a multi-level map (TWHL). And remember to add `overviews/my_map.bmp` and `overviews/my_map.txt` to the map package.

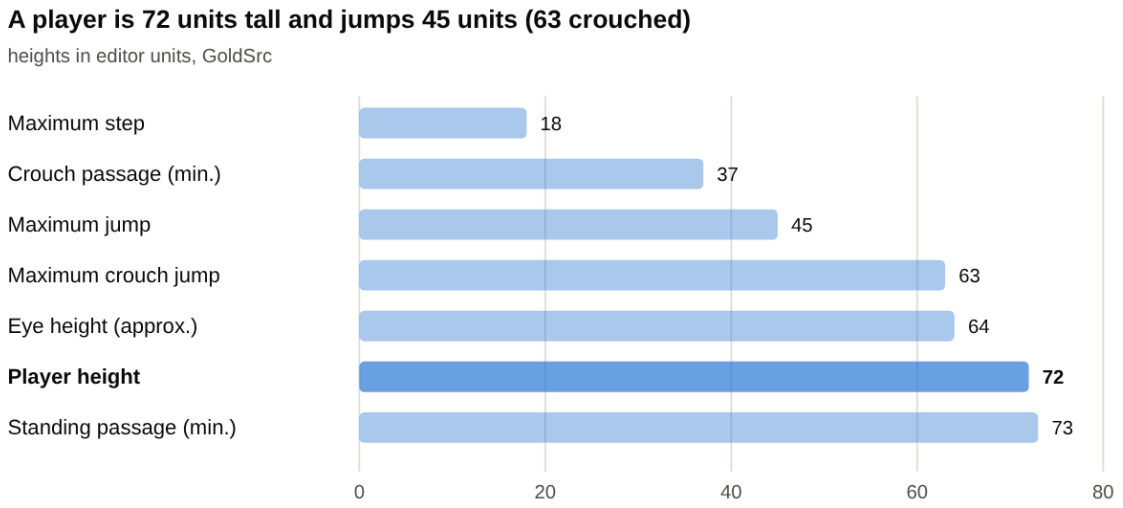
Source: TWHL: Creating Overviews.

Part 4: Appendices

Cheat sheets to keep at hand.

1. Lengths

Everything is measured in editor **units**. Keep these numbers at hand so players pass, climb and jump where you want them to (VDC: GoldSrc Dimensions).



player dimensions · 7 heights (VDC)

Player measurements

Measurement	Units	What it means
Maximum step	18	Above this, you must jump
Crossable gap	32	A gap narrower than 32 can be walked over
Minimum passage width	33	You need at least 33 between two walls to get through
Minimum crouch passage	37	Minimum height of a duct
Maximum jump	45	Height of a ledge reachable by jumping
Maximum crouch jump	63	Jump while crouching in mid-air
Eye height	about 64	Camera height
Player height	72	
Minimum standing passage	73	Minimum height of a door or corridor

> **Tip:** do not build to the exact unit. A door 96 to 112 high, a corridor 64 to 128 wide leave some margin and look natural.

Conversions

One unit is roughly **one inch, i.e. 2.54 cm**. Some estimate it at 0.75 inch (1.9 cm) instead: these are approximations (VDC).

Units	About
16	40 cm (16 in)
64	1.6 m (5.3 ft)
128	3.3 m (10.7 ft)
1024	26 m (85 ft)

Calculated with 1 unit = 2.54 cm.

Maximum map size

8192 × 8192 × 8192 units, i.e. 4096 in each direction from the centre of the grid. Beyond that, the game refuses to load the map, even if J.A.C.K. lets you build larger (VDC).

Source: VDC: GoldSrc Dimensions.

2. Bots and waypoints

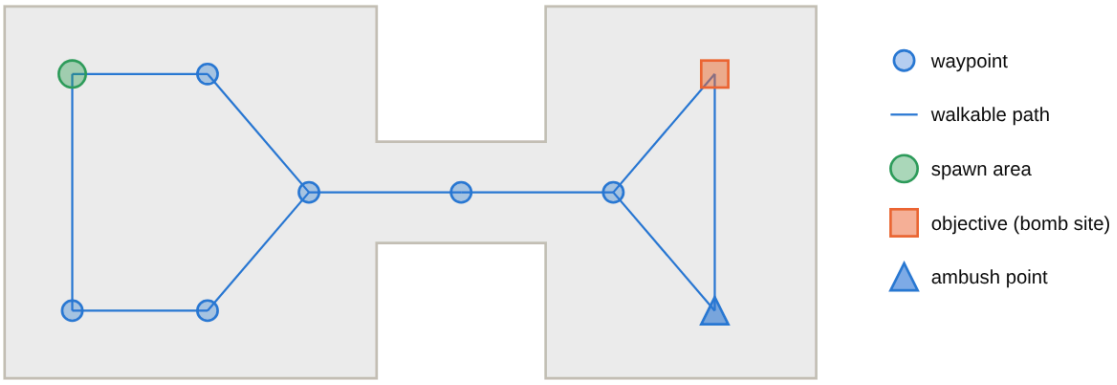
Testing a map alone does not tell you much: you need opponents. But **Counter-Strike 1.6 has no official bots** (VDC: Counter-Strike).

The options

Option	How it works
Condition Zero	CS 1.6 maps run in Counter-Strike: Condition Zero, which has its own bots (CS_Bot) (VDC)
YaPB	Free third-party bots (MIT licence), for CS 1.6 and Condition Zero
Other third-party bots	Listed by the bots-united community, cited by VDC

What is a waypoint?

Bots move along a network of linked points



Top view: without this network, a bot does not know where to go.

waypoints · bot navigation network

A bot does not "see" the map like a player. It moves along a network of **waypoints** (a "navigation graph" in YaPB) linked by paths. Some points have a role: spawn area, objective (bomb site, hostages), ambush point, ladder, jump.

Place and link waypoints

The principle is the same for every third-party bot:

1. Launch the map with the bot installed and turn on its waypoint editor (a console command specific to each bot).
2. Walk through the map and drop a point at each important spot: junctions, doors, top and bottom of stairs and ladders.
3. Check that the links between points go only where a player can go.
4. Mark the special points (objectives, ambushes, ladders).
5. Save: a waypoint file is created for the map.

The exact commands vary from bot to bot: follow the chosen bot's documentation. For YaPB, navigation graphs are managed in a separate repository, where you can also request one for a map (GitHub: YaPB).

Ship your waypoints

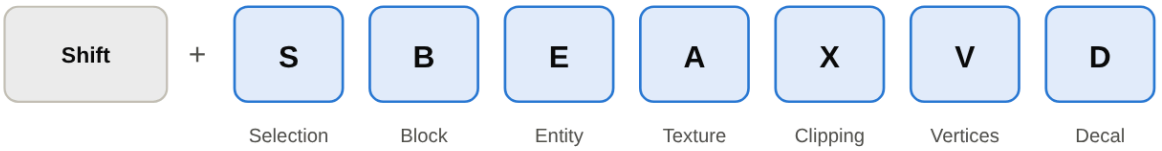
Add the waypoint file to your package (Part 2, chapter 19): players using the same bot will not have to redo it.

Sources: VDC: Counter-Strike, GitHub: YaPB.

3. J.A.C.K. keyboard shortcuts

J.A.C.K. uses Hammer's shortcuts (VDC: Hammer Hotkey Reference). Depending on your version, some may differ: check the menus, where each shortcut is shown next to its command.

Shift + a letter: the tools in the left bar



Shortcuts inherited from Hammer; change them in Tools > Options.

tools · 7 Shift shortcuts

Tools

Shortcut	Tool
Shift+S	Selection
Shift+G	Magnifier (zoom)
Shift+C	Camera
Shift+E	Entity
Shift+B	Block (brush)
Shift+A	Texture application
Shift+T	Apply current texture to the selection
Shift+D	Decal
Shift+X	Clipping
Shift+V	Vertices

Selection

Shortcut	Action
Ctrl+click	Add or remove an object from the selection
Esc or Shift+Q	Clear the selection
Page Up / Page Down	Cycle through objects under the cursor
Shift + drag	Clone
Arrow keys (2D view)	Move by one grid step

Texture application (in the 3D view)

Shortcut	Action
Left-click	Select a face
Ctrl+click	Add a face
Shift+click	Select all faces of the brush

Shortcut	Action
Right-click	Apply the current texture to the face under the cursor
Alt+right-click	Apply texture and alignment

Menus (used in this course)

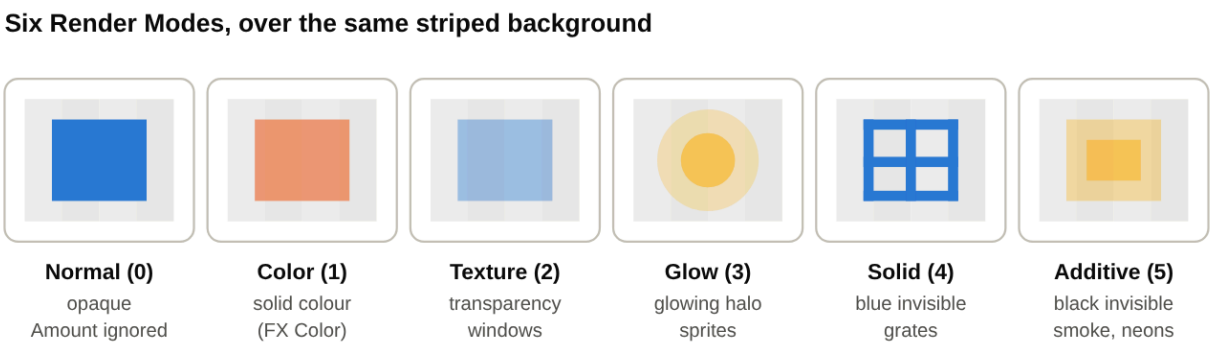
Shortcut	Action
Alt+Enter	Object properties
Ctrl+T / Ctrl+W	Tie to entity / move to world
Ctrl+G / Ctrl+U	Group / ungroup
Ctrl+Shift+H	Hollow
Ctrl+Shift+C	Carve
Ctrl+M	Precise transform
Ctrl+Shift+V	Paste special
Ctrl+B	Snap to grid
Ctrl+Shift+G	Go to brush number…
Alt+P	Check for problems
[and]	Finer / coarser grid
Z	Free-look in the 3D view
F9	Compile (Run Map)

Source: VDC: Hammer Hotkey Reference.

4. Render properties

Almost every visible entity (brushes, models, sprites) has four render properties (TWHL: Render Properties):

- **Render Mode** (`rendermode`): how the entity is drawn;
- **FX Amount** (`renderamt`): from 0 (invisible) to 255 (opaque), depending on the mode;
- **FX Color** (`rendercolor`): an R G B colour;
- **Render FX** (`renderfx`): an animated effect on top.



Render Mode · 6 modes illustrated

Render Mode

Mode	Value	Formula (TWHL)	Use
Normal	0	src	Opaque rendering, FX Amount ignored
Color	1	c × a + dest × (1 − a)	Replaces the texture with FX Color
Texture	2	src × a + dest × (1 − a)	Windows (FX Amount around 100)
Glow	3	src × a + dest, no depth test	Sprite halos, which grow and fade with distance
Solid	4	src × texture alpha + dest × (1 − alpha)	Masked `{` textures (FX Amount 255)
Additive	5	src × a + dest	Smoke, holograms, glowing sprites: black becomes transparent

In the formulas, *src* is the entity, *dest* what is behind it, *a* the FX Amount and *c* the FX Color.

Render FX

Value	Effect
0	Normal
1 to 4	Slow or fast, narrow or wide pulses
5, 6	Fades out, slowly or quickly
7, 8	Fades in, slowly or quickly
9 to 11	Increasingly fast strobe
12, 13	Slow or fast flicker
14	Constant Glow: cancels the Glow mode's growth
15	Distort
16	Hologram (distortion and fade)
19	Glow Shell: aura around a model

Values from TWHL: func_wall and TWHL: Render Properties.

Settings to remember

Desired effect	Render Mode	FX Amount
Window	Texture	about 100
Grate, fence, foliage (`{` texture)	Solid	255
Lamp halo (env_sprite)	Glow	255, with Render FX Constant Glow if the size must not change
Beam, smoke	Additive	100 to 200
Invisible object (to reveal with env_render)	Texture	0

Sources: TWHL: Render Properties, TWHL: func_wall, TWHL: env_render.

5. Sources and useful links

All the pages consulted for this course, grouped by site. The course's outline and spirit come from the French tutorial "Une map avec Hammer !" by Loulimi and Mathieu Nebra (OpenClassrooms, Creative Commons licence, updated 24/08/2011).

Tools

- Valve Developer Community: J.A.C.K.
- Steam: J.A.C.K.
- ModDB: J.A.C.K. 1.1.1058
- VDC: VHLT
- VDC: WadMaker and SpriteMaker
- GitHub: YaPB

Setup and compilation

- TWHL: Setup Guide Hammer/J.A.C.K.
- TWHL: Compile
- TWHL: Compiling 101
- TWHL: Setting up TrenchBroom (compilation section)
- VDC: HLBSP
- VDC: SirYodaJedi's compile settings
- TWHL: Pointfiles
- TWHL: How to fix those leaks
- TWHL: Total Map Optimisation Part 1
- TWHL: Hint Brushes
- VDC: func_detail (GoldSrc)
- TWHL: Creating Overviews
- VDC: Resource list (GoldSrc)

Textures

- VDC: Texture prefixes
- TWHL: Tool Textures
- VDC: Tool textures (GoldSrc)
- the303: GoldSrc Map Texture Tutorial
- TWHL: Sound Effects for Textures (materials.txt)
- TWHL: Render Properties

Counter-Strike

- VDC: Counter-Strike
- VDC: List of Counter-Strike entities
- VDC: Counter-Strike.fgd
- TWHL: Introduction to Counter-Strike mapping
- TWHL: Stripping and Equipping Weapons in Counter-Strike
- VDC: info_map_parameters
- the303: func_vehicle
- VDC: GoldSrc Dimensions

Entities

- TWHL: Doors
- TWHL: func_door_rotating
- TWHL: func_wall
- TWHL: Ladders and func_ladder
- TWHL: light_spot
- TWHL: multi_manager
- TWHL: Multisource
- TWHL: trigger_changetarget
- TWHL: Teleporters
- TWHL: env_sprite
- TWHL: env_render
- VDC: env_explosion (GoldSrc)
- VDC: env_shake
- VDC: func_breakable (GoldSrc)
- TWHL: func_breakable
- TWHL: func_pushable
- TWHL: ambient_generic
- VDC: env_sound (GoldSrc)
- VDC: func_rotating (GoldSrc)
- TWHL: func_pendulum
- VDC: trigger_push (GoldSrc)
- VDC: trigger_hurt (GoldSrc)
- TWHL: func_train
- TWHL: path_track

- TWHL: trigger_camera
- VDC: game_text (GoldSrc)
- TWHL: cycler_sprite
- VDC: Hammer Hotkey Reference