Une map Counter-Strike 1.6 avec J.A.C.K.

Nov 3, 2026

Ce cours vous apprend à créer une map pour Counter-Strike 1.6 avec l'éditeur J.A.C.K., de l'installation à la compilation. Il reprend le plan du tutoriel « Une map avec Hammer ! » (OpenClassrooms), mis à jour pour J.A.C.K. et les compilateurs VHLT. Il se lit dans l'ordre des onglets : Partie 1 (fonctions de base), Partie 2 (entités et effets), Partie 3 (compilation et optimisation), Partie 4 (annexes).

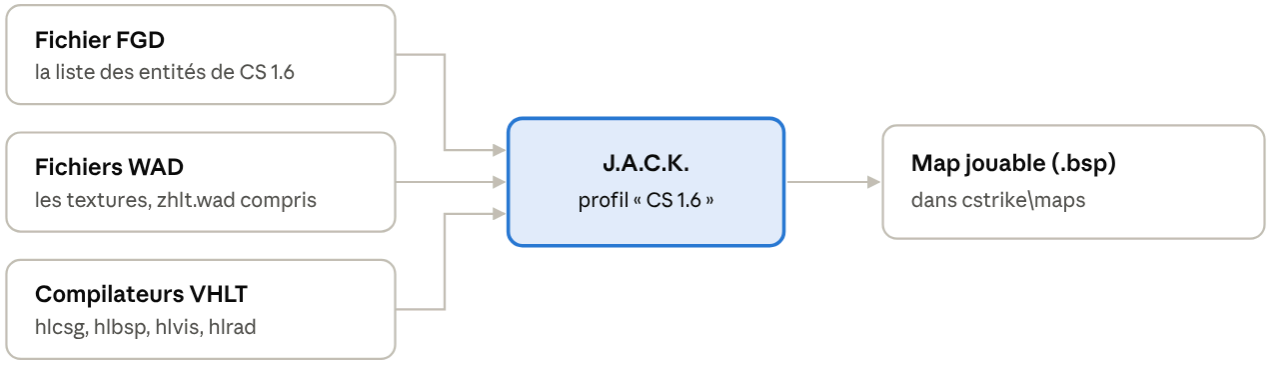
Partie 1 : Fonctions de base de J.A.C.K.

Initiez-vous à J.A.C.K. et apprenez à manier les blocs. C'est par là que doivent commencer les débutants.

1. Installation et configuration

Pour mapper sous Counter-Strike 1.6, il vous faut trois choses : l'éditeur J.A.C.K., des compilateurs (VHLT), et deux types de fichiers fournis avec le jeu ou l'éditeur, le FGD et les WAD. Vous allez les installer, puis les relier entre eux dans un **profil de jeu**.

Le profil de jeu relie J.A.C.K. à vos fichiers



configuration · 3 sources de fichiers, 1 profil, 1 map compilée

Le profil de jeu est le centre de tout : il dit à J.A.C.K. quelles entités proposer (FGD), quelles textures afficher (WAD), et quels programmes lancer pour fabriquer la map jouable.

Télécharger et installer

1. **J.A.C.K.** existe en version gratuite (site officiel jack.hlfx.ru) et en version payante sur Steam (14,99 $). Il fonctionne sous Windows, Linux et macOS.

2. **VHLT** (Vluzacn's Half-Life Tools) : la version v34, sortie en août 2015, est la dernière (VDC : VHLT). Elle se télécharge sur GameBanana. Décompressez-la dans un dossier simple, par exemple `C:\VHLT`. Elle contient le fichier `zhlt.wad`, indispensable pour les textures outils.
3. **Le FGD** : J.A.C.K. est livré avec des FGD mis à jour (VDC : J.A.C.K.). Pour Counter-Strike, prenez `halflife-cs.fgd`, ou sa version améliorée Counter-Strike.fgd.
4. **Les WAD** sont déjà dans le jeu, dans les dossiers `valve` et `cstrike`.

> **Attention :** la version Steam de J.A.C.K. ne contient pas les compilateurs GoldSrc, et la version gratuite en contient une version ancienne. Dans les deux cas, installez VHLT à part (VDC : J.A.C.K.).

> **Info :** avec Steam, le jeu se trouve en général dans `C:\Program Files (x86)\Steam\steamapps\common\Half-Life`. Le dossier `cstrike` est celui de Counter-Strike, le dossier `valve` celui de Half-Life.

Configurer J.A.C.K.

Ouvrez **Tools > Options**, puis l'onglet **Game Profiles** (« Game Configurations » dans Hammer). La démarche suit le guide d'installation de TWHL.

Créer le profil

1. Cliquez sur **Edit**, puis **Add**.
2. Tapez `CS 1.6` et validez.

Sous-onglet « Game »

1. Dans **Game Data files**, cliquez sur **Add** et choisissez le FGD de Counter-Strike.
2. **Default PointEntity class** : `info_player_start`. **Default SolidEntity class** : `func_wall`. Ce sont les entités proposées par défaut, vous gagnerez du temps.
3. Ignorez « Texture format » et « Map type ».

Sous-onglet « Directories »

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

Sous-onglet « Build Programs »

- **Game Executable** : `Half-Life\hl.exe`. Ajoutez `-game cstrike` après le nom pour que le jeu se lance directement sur Counter-Strike.

- **CSG, BSP, VIS, RAD** : les quatre programmes de VHLT (`hlcsg.exe` , `hlbsp.exe` , `hlvis.exe` , `hlrad.exe` , ou leurs versions 64 bits).
- **Source Maps Directory** : `Half-Life\cstrike\maps` . C'est là que la map compilée sera copiée.

> **Attention :** si les compilateurs et le dossier des maps sont sur deux disques différents, la compilation peut échouer. Mettez-les sur le même disque (TWHL).

Sous-onglet « Textures »

Cliquez sur **Add WAD** et ajoutez :

- `halflife.wad` et `liquids.wad` (dossier `valve`), utiles pour les textures d'eau et certaines textures outils ;
- les WAD de `cstrike` dont vous avez besoin (`cstrike.wad` , `cs_dust.wad` …), **sauf** `cached.wad` ;
- `zhlt.wad` , livré avec VHLT.

> **Info :** contrairement à Hammer, J.A.C.K. garde une liste de WAD propre à chaque profil. Vous pouvez donc avoir un profil Half-Life et un profil CS sans tout reconfigurer à chaque fois.

Vérifier que tout marche

1. Validez avec **OK**, puis **File > New** et choisissez le profil `CS 1.6` .
2. Ouvrez le navigateur de textures (**Browse** dans le panneau de droite) : les textures de vos WAD doivent apparaître.
3. Dans la liste des entités, cherchez `armoury_entity` : si elle est là, le FGD de Counter-Strike est bien chargé.

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

2. Les bases de toute map

Avant de construire quoi que ce soit, il faut comprendre trois choses : comment le moteur voit votre map, comment l'éditeur vous la montre, et de quoi elle est faite.

Le moteur GoldSrc

Counter-Strike 1.6 tourne sur le moteur **GoldSrc**, celui de Half-Life. C'est un moteur de type BSP : la map est découpée en volumes convexes, ce qui permet au jeu de savoir très vite ce qui est visible. Cela impose quelques règles que vous retrouverez tout au long du cours.

- **Taille maximale** : 8192 × 8192 × 8192 unités, soit 4096 unités dans chaque direction depuis le centre (VDC : GoldSrc Dimensions). J.A.C.K. accepte des maps bien plus grandes, mais le jeu ne les chargera pas.
- **L'unité** : tout se mesure en unités de l'éditeur. Une unité vaut à peu près un pouce (2,54 cm). Ce n'est qu'une approximation : retenez surtout que le joueur mesure 72 unités de haut.

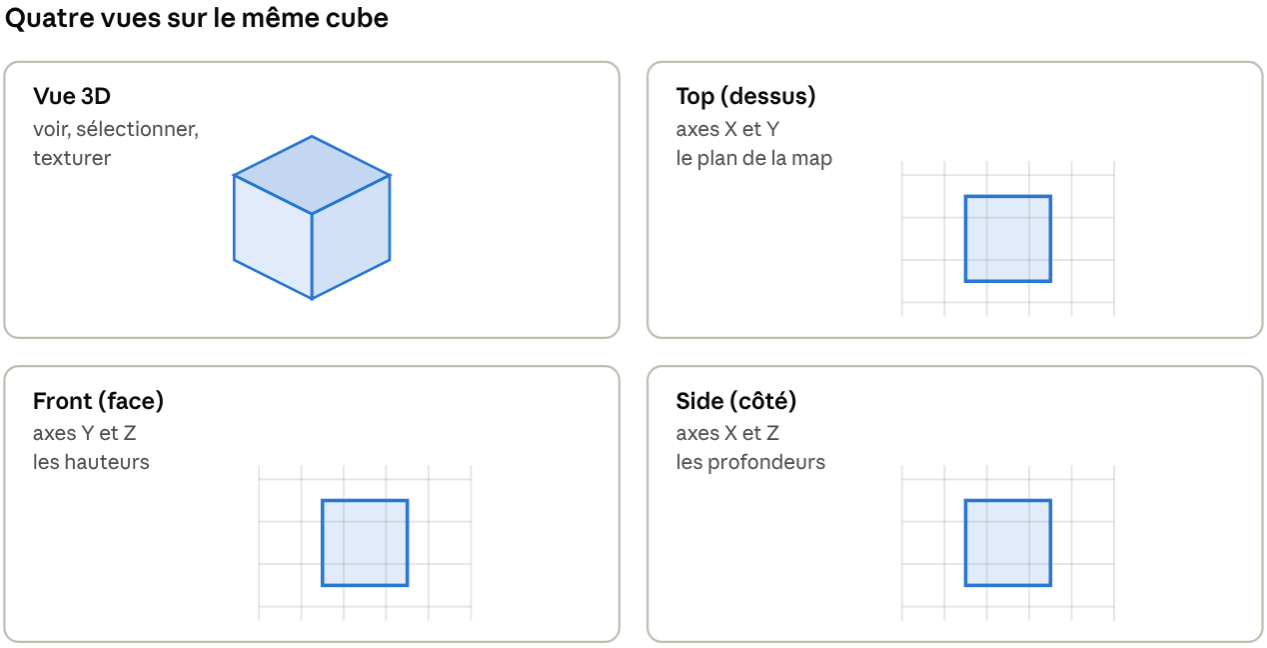
> **Info :** les dimensions utiles (hauteur des marches, des sauts, des passages) sont réunies dans l'annexe « Les longueurs », Partie 4.

L'environnement de travail

L'écran de J.A.C.K. se compose de :

- **quatre vues** au centre (détaillées ci-dessous) ;
- **la barre d'outils de gauche** : sélection, création de blocs, d'entités, texturage, découpe, sommets ;
- **la barre du haut** : fichier, annuler, grille, verrouillage de texture, compilation ;
- **le panneau de droite** : texture courante, forme des blocs, entité à placer, boutons **To Entity** et **To World**, groupes de visibilité ;
- **la barre d'état** en bas : position du curseur, taille de la sélection et, ajout de J.A.C.K., position du centre de la sélection (VDC : J.A.C.K.).

Les vues 2D et la vue 3D



les 4 vues de J.A.C.K. · un même cube

Les trois vues 2D montrent la map à plat, chacune selon deux axes. C'est là que vous dessinez et déplacez les blocs, au pixel près grâce à la grille. La vue 3D sert à vérifier le résultat, à sélectionner des faces et à texturer.

> **Info :** un clic droit sur le nom d'une vue (en haut à gauche de chaque vue) permet de changer son type. Pour la vue 3D, choisissez **3D Textured** pour voir les textures, **3D Flat** pour des couleurs unies, ou **Wireframe** pour les seules arêtes.

J.A.C.K. ajoute aux vues des fonctions absentes de Hammer : une image de fond dans les vues 2D (pratique pour recopier un plan), des caméras que vous pouvez enregistrer et rappeler, et l'affichage en temps réel des textures animées, transparentes et du ciel (VDC : J.A.C.K.).

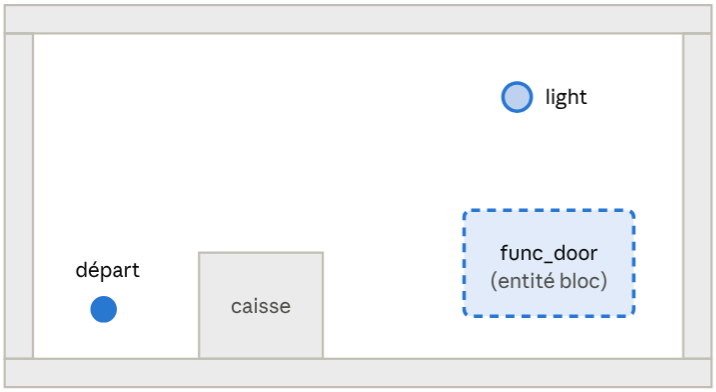
Se déplacer

- **Dans une vue 2D** : la molette zoome, la barre d'espace maintenue + clic gauche fait glisser la vue.
- **Dans la vue 3D** : appuyez sur **Z** pour passer en vol libre, la souris oriente le regard et les touches W, A, S, D (ou les flèches) font avancer. Appuyez de nouveau sur **Z** pour retrouver le curseur.

> **Erreur fréquente :** en vol libre, la souris ne sélectionne plus rien. Si plus aucun clic ne répond dans la vue 3D, appuyez sur Z.

Les constituants d'une map

Une map est une boîte fermée : le vide ne doit jamais être visible



Vue de dessus d'une petite pièce

Blocs du monde
le squelette : murs, sol, plafond, décor fixe (worldspawn)

Entités
les organes : lumières, départs, portes, déclencheurs

Le vide
tout ce qui est hors de la boîte : le joueur ne doit jamais le voir

constituants d'une map · blocs, entités, vide

Une map se compose de deux familles d'objets :

- **Les blocs** (brushes) : des volumes convexes qui forment le squelette de la map. Tant qu'ils ne sont liés à aucune entité, ils font partie du « monde » (worldspawn) : ils sont fixes, solides et projettent des ombres.
- **Les entités** : tout ce qui a un comportement. Certaines sont des points (une lumière, un départ de joueur), d'autres sont des blocs auxquels on a donné un rôle (une porte, une zone d'achat).

Et il y a ce qui n'est rien : **le vide**. Tout ce qui se trouve hors des blocs est du vide, et le joueur ne doit jamais pouvoir le voir. Votre map doit donc être une boîte entièrement fermée. S'il reste un trou, la compilation détecte une « fuite » (leak) et la map sera mal éclairée et lente ; c'est l'objet du chapitre « La compilation (B : débogage) », Partie 3.

> **Attention :** la boîte doit être fermée par des blocs du monde. Une porte, un mur cassable ou une vitre (entités) ne bouchent pas le vide.

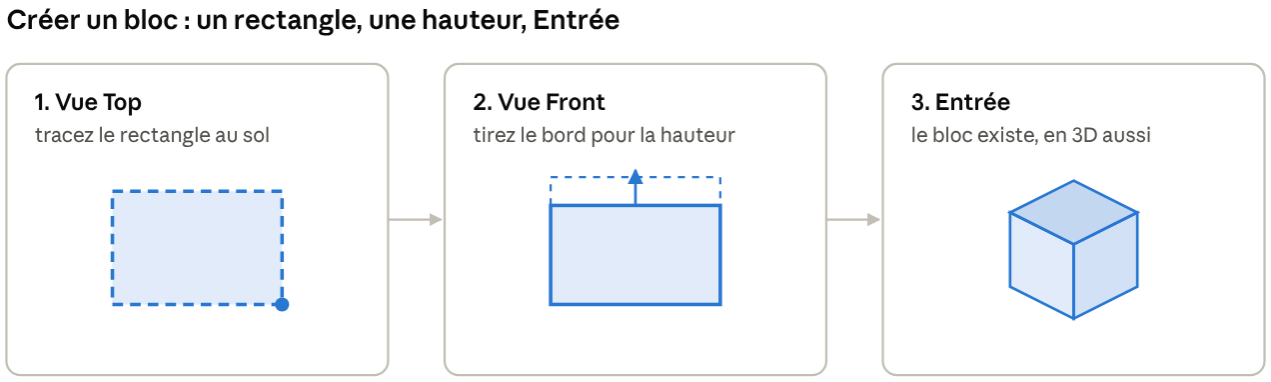
Sources : VDC : GoldSrc Dimensions, VDC : J.A.C.K., VDC : Hammer Hotkey Reference.

3. Les blocs (A : création)

Les blocs sont le squelette de votre map. Ce chapitre vous apprend à en créer, à les sélectionner et à les dupliquer, sans vous soucier pour l'instant de faire une map jouable.

Créer un bloc

Ouvrez une map vierge avec **File > New** et choisissez votre profil `CS 1.6`. Les quatre vues sont vides : c'est normal.



création d'un bloc · 3 étapes

1. Choisissez l'outil **Block** dans la barre de gauche (Shift+B). Le curseur devient une croix.
2. Dans la vue **Top**, cliquez et faites glisser pour tracer un rectangle de 3 × 3 carreaux.
3. Dans la vue **Front**, tirez le bord du haut pour donner au bloc une hauteur de 3 carreaux.
4. Vérifiez dans la vue **Side** que la forme est la bonne.
5. Le curseur toujours dans une vue 2D, appuyez sur **Entrée**.

> **Erreur fréquente :** si vous appuyez sur Entrée alors que la souris est dans la vue 3D, le bloc n'est pas créé. Revenez dans une vue 2D.

Pour annuler avant d'avoir validé, appuyez sur **Échap**. Pour supprimer un bloc déjà créé, sélectionnez-le et appuyez sur **Suppr**.

Sélectionner un bloc

Prenez l'outil **Selection** (Shift+S).

- **Dans la vue 3D**, cliquez sur le bloc.
- **Dans une vue 2D**, cliquez sur un de ses bords ou sur la petite croix au centre. Ailleurs, rien n'est sélectionné.
- **Plusieurs blocs** : maintenez **Ctrl** pendant les clics.

- **Toute une zone** : dans une vue 2D, tracez un cadre autour des blocs puis appuyez sur **Entrée**.
- **Objets superposés** : les touches **Page précédente / Page suivante** passent d'un objet à l'autre sous le curseur (VDC : Hotkeys).

> **Info :** J.A.C.K. sait aussi sélectionner en faisant glisser dans la vue 3D (VDC : J.A.C.K.).

Le clonage

Pour dupliquer un bloc, deux méthodes :

- **Ctrl+C** puis **Ctrl+V**, comme partout ;
- plus rapide : maintenez **Shift** pendant que vous déplacez le bloc avec la souris. Une copie reste en place, l'autre suit le curseur.

J.A.C.K. garde les liens entre entités quand vous clonez ou copiez un groupe : si une porte et son bouton sont copiés ensemble, la copie du bouton ouvre la copie de la porte (VDC : J.A.C.K.).

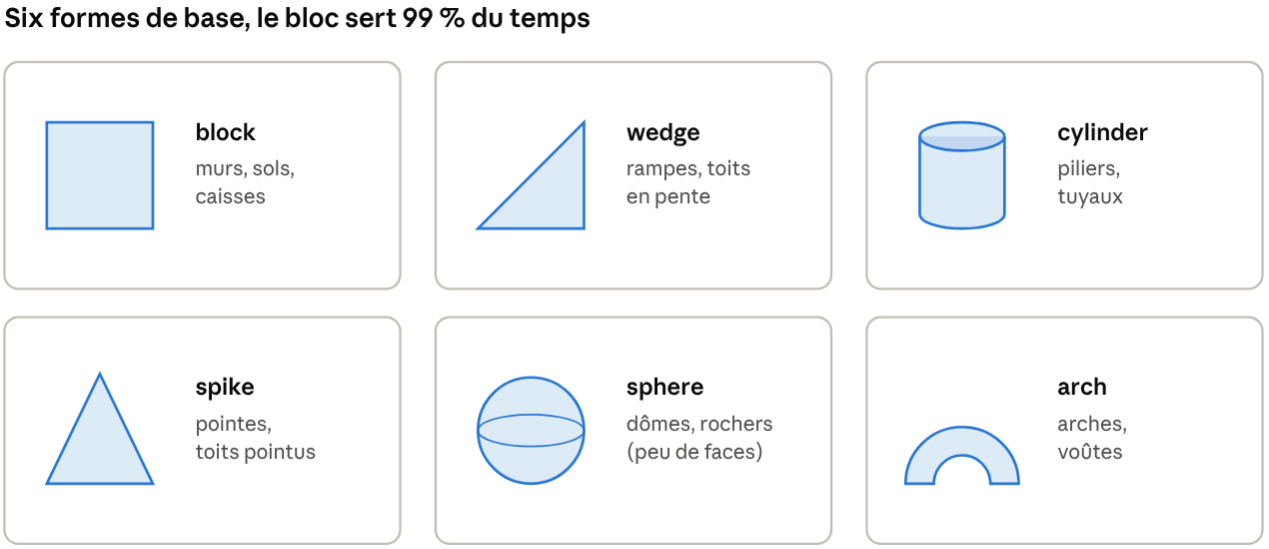
Zoom et précision de la grille

La grille sert d'aimant : tout ce que vous dessinez se cale sur ses lignes.

- **[** rend la grille plus fine (16, 8, 4, 2, 1 unités), **]** la rend plus large.
- La barre d'état affiche le pas actuel, par exemple « Snap: On Grid: 16 ».
- La **molette** zoome. Zoomer ne change pas le pas de la grille : ce sont deux réglages indépendants.

> **Conseil :** travaillez en général avec une grille de 8 ou 16, et descendez à 1 seulement pour les petits détails. Des blocs posés hors grille sont la première cause de micro-trous et de fuites.

Les différentes formes de blocs



formes de blocs · 6 primitives

La forme se choisit dans le panneau de droite, rubrique **Objects** (ou « Primitives » selon la version). Pour les cylindres, sphères et arches, le champ **Faces** règle le nombre de côtés.

> **Attention :** plus un bloc a de faces, plus le moteur doit en dessiner. Un pilier de 8 faces suffit le plus souvent ; montez à 12 ou 16 seulement pour un gros cylindre bien visible.

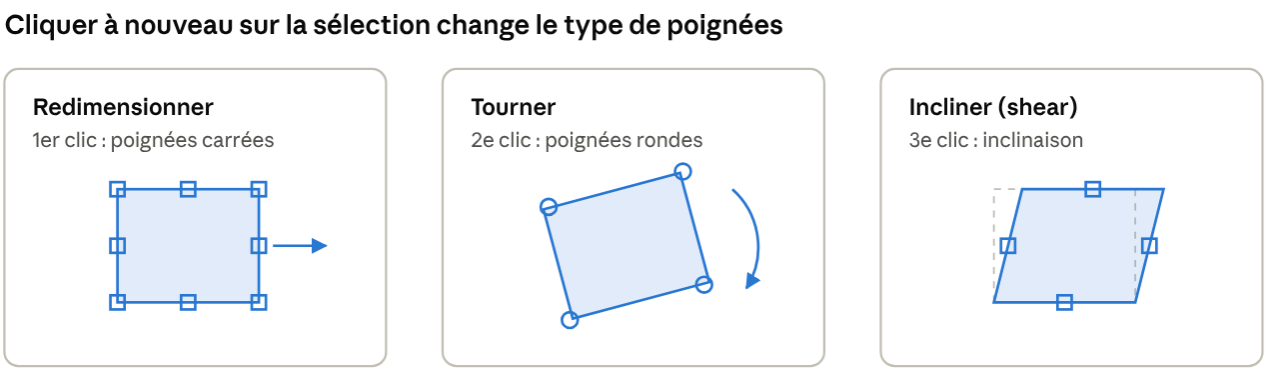
Sources : VDC : Hammer Hotkey Reference, VDC : J.A.C.K..

4. Les blocs (B : modification de la structure)

Vous savez créer des blocs. Il est temps de les transformer : les tourner, les étirer, les couper, les creuser et déplacer leurs sommets.

Les poignées de sélection

Un bloc sélectionné affiche des poignées dans les vues 2D. **Cliquez à nouveau sur le bloc** pour passer d'un type de poignées au suivant.



outil Selection · 3 types de poignées

- **Redimensionner** : tirez une poignée carrée. Les poignées des coins agissent sur deux axes à la fois.
- **Tourner** : tirez une poignée ronde. La rotation se fait par pas de 15° ; maintenez **Shift** pour tourner librement (VDC : Hotkeys).
- **Incliner** : tirez une poignée du milieu d'un bord, le bloc se déforme en parallélogramme.

> **Info :** pour des valeurs exactes (tourner de 45° pile, agrandir de 150 %), utilisez **Tools > Transform** (Ctrl+M).

> **Attention :** après une rotation libre, les sommets d'un bloc tombent rarement sur la grille. Rappelez-vous de **Snap to Grid** (Ctrl+B) pour les recaler, sinon vous créez de minuscules trous entre les blocs.

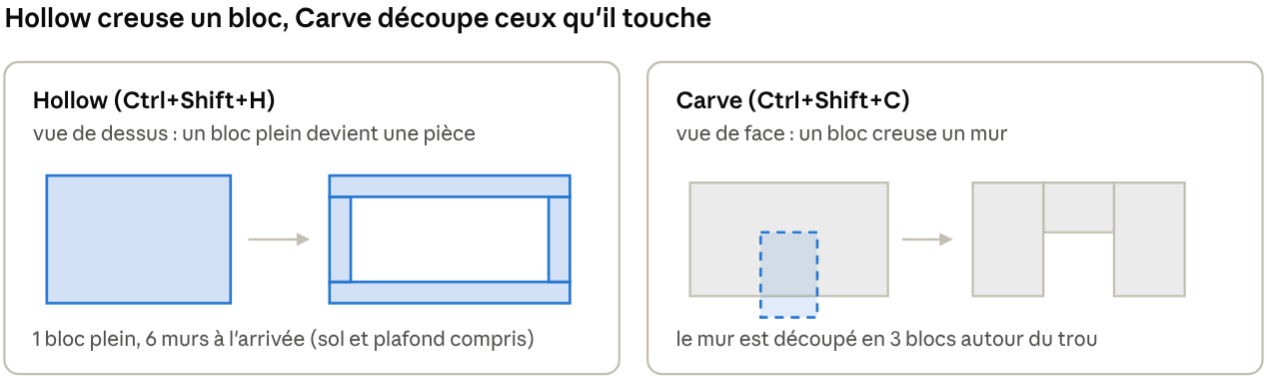
Déplacer avec précision

Avec un bloc sélectionné, les **flèches du clavier** dans une vue 2D le déplacent d'un pas de grille. C'est la meilleure façon de coller deux blocs l'un contre l'autre sans chevauchement.

Grouper et dégrouper

Pour manipuler plusieurs blocs comme un seul objet (une table, un escalier), sélectionnez-les et faites **Ctrl+G**. **Ctrl+U** les sépare. Un groupe n'existe que dans l'éditeur : en jeu, il n'a aucun effet.

Hollow et Carve



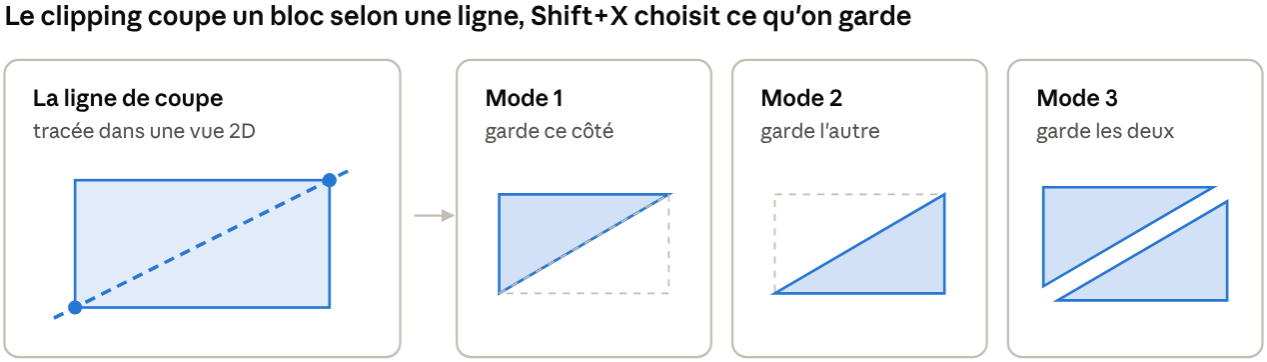
Hollow et Carve · avant et après

Hollow (Ctrl+Shift+H dans J.A.C.K., Ctrl+H dans Hammer) transforme un bloc plein en boîte creuse, avec une épaisseur de murs que vous choisissez. C'est la façon la plus rapide de faire une pièce fermée : créez un bloc de la taille de la pièce, puis Hollow avec une épaisseur de 16.

Carve (Ctrl+Shift+C) se sert du bloc sélectionné comme d'un emporte-pièce : tous les blocs qu'il touche sont découpés autour de lui. Il suffit ensuite de supprimer le bloc emporte-pièce.

> **Attention :** Carve découpe en beaucoup de petits blocs, souvent mal formés, surtout avec un cylindre ou une sphère. Utilisez-le seulement avec des blocs simples, et préférez le Clipping ou la construction à la main pour une porte ou une fenêtre.

Le Clipping



outil Clipping · 3 modes

L'outil **Clipping** (Shift+X) coupe un bloc en deux, net, selon un plan.

1. Sélectionnez le bloc.

2. Prenez l'outil Clipping.
3. Dans une vue 2D, cliquez et faites glisser pour tracer la ligne de coupe.
4. Appuyez sur **Shift+X** pour faire défiler les modes : garder un côté, garder l'autre, garder les deux.
5. Appuyez sur **Entrée** pour valider.

Exemple : pour une rampe, créez un bloc de 256 × 64 × 64, et dans la vue Front, tracez une diagonale du coin bas gauche au coin haut droit. Gardez le morceau du bas : vous obtenez une rampe parfaite, sans passer par l'outil Vertex.

Le Vertex Manipulation (morphing)

L'outil **Vertex** (Shift+V) affiche les sommets du bloc sélectionné. Faites-les glisser pour obtenir des formes qu'aucune primitive ne donne : rochers, rampes tordues, pointes.

- Les points pleins sont les **sommets**.
- Les petits points au milieu des arêtes permettent de déplacer **une arête entière**.
- Si deux sommets se superposent, l'éditeur propose de les fusionner.

> **Erreur fréquente :** un bloc doit toujours rester **convexe** (sans creux) et chaque face doit rester **plane**. Si vous déplacez un seul sommet d'une face carrée, la face se tord et le bloc devient invalide. J.A.C.K. peut alors **trianguler** la face pour la découper en deux triangles plans (VDC : J.A.C.K.).

> **Info :** J.A.C.K. peut verrouiller les axes de texture pendant que vous déplacez des sommets, pour que la texture ne glisse pas (VDC : J.A.C.K.).

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

5. Les textures (A : utiliser des textures)

Un bloc sans texture choisie reçoit la texture courante. Ce chapitre vous montre comment choisir une texture, l'appliquer, l'aligner, et quelles textures ont un rôle spécial.

Choisir une texture

Dans le panneau de droite, cliquez sur **Browse**. Le navigateur affiche toutes les textures des WAD de votre profil.

- Le champ **Filter** affiche seulement les textures dont le nom contient ce que vous tapez (`door`, `crate`, `wall`…).
- La case **Only used textures** ne montre que les textures déjà présentes dans la map.

- J.A.C.K. range les textures dans des **onglets** (VDC : J.A.C.K.).

Double-cliquez sur une texture : elle devient la texture courante.

Appliquer une texture

Avant la création du bloc : choisissez la texture, puis créez le bloc. Toutes ses faces la portent.

Après la création :

- tout le bloc : sélectionnez-le et faites **Shift+T** (appliquer la texture courante) ;
- une seule face : prenez l'outil **Texture Application** (Shift+A), cliquez sur la face dans la vue 3D, puis choisissez la texture.

Avec l'outil Texture Application, dans la vue 3D (VDC : Hotkeys) :

- **clic gauche** sélectionne une face, **Ctrl+clic** en ajoute d'autres, **Shift+clic** prend toutes les faces du bloc ;
- **clic droit** applique la texture courante à la face visée ;
- **Alt+clic droit** applique la texture et l'alignement de la face sélectionnée.

Aligner une texture



fenêtre Texture Application · effet de Scale et Shift

La fenêtre Texture Application règle le placement de la texture sur les faces sélectionnées :

- **Scale X / Y** : la taille. 1 = taille réelle, 0,5 = deux fois plus petite (donc plus répétée), 2 = deux fois plus grande.
- **Shift X / Y** : le décalage en pixels, pour caler un motif sur un bord.
- **Rotation** : l'angle de la texture sur la face.
- **Justify** : cale la texture à gauche, à droite, en haut, en bas, au centre, ou **Fit** pour l'étirer exactement sur la face. J.A.C.K. sait ajuster plusieurs faces en une fois.

- **World / Face** : en mode World, la texture suit les axes du monde, ce qui assure la continuité d'un bloc à l'autre ; en mode Face, elle suit la face, ce qui convient mieux aux pentes.
- **Treat as one** : plusieurs faces sélectionnées sont traitées comme une seule pour Justify.

Exemple : une porte de 64 × 112 unités avec une texture de porte de 64 × 128 pixels. Sélectionnez la face avant et cliquez sur **Fit** : l'échelle passe à 1 × 0,875 et la porte tombe pile sur le bloc.

> **Info :** le bouton **Texture Lock** de la barre du haut fait suivre la texture quand vous déplacez un bloc. Désactivé, la texture reste fixe dans le monde et « glisse » sur le bloc.

> **Info : Replace** (menu Tools) remplace une texture par une autre dans toute la map ; J.A.C.K. peut adapter l'échelle si les deux textures n'ont pas la même taille (VDC : J.A.C.K.).

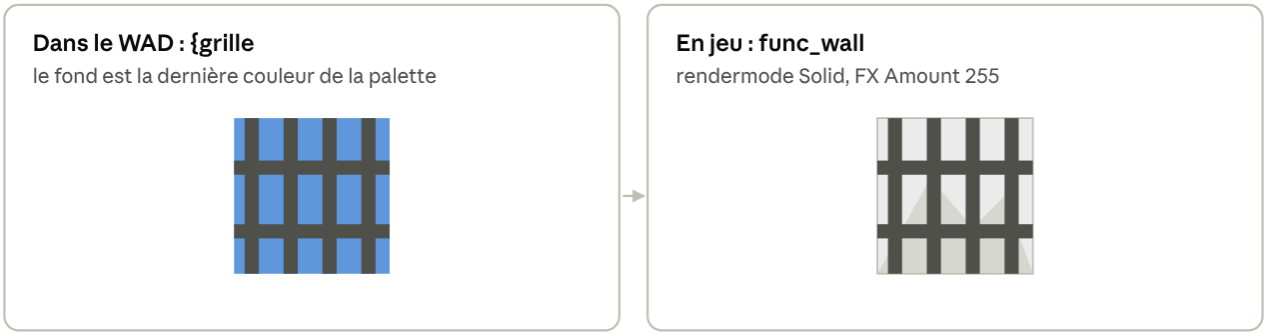
Les noms spéciaux de textures

Le début du nom d'une texture change son comportement en jeu (VDC : Texture prefixes).

Préfixe	Effet	Exemple d'usage
`{`	Texture masquée : la dernière couleur de la palette (en général bleu pur) devient transparente, si l'entité a un rendermode autre que Normal	Grilles, clôtures, feuillages
`!`	Liquide : ondulations, on peut y nager	Eau
`!slime`, `!lava`	Liquide qui blesse (lava : dégâts doublés)	Bassin toxique
`!cur_0`, `!cur_90`…	Courant qui pousse dans une direction	Rivière
`+0` à `+9`	Animation de 10 images maximum, environ 10 images par seconde	Écrans, néons qui clignotent
`+A` à `+J`	Seconde série : un bouton ou un func_wall bascule entre `+0` et `+A` quand on le déclenche	Bouton qui s'allume

Préfixe	Effet	Exemple d'usage
`-0` à `-9`	Variante tirée au hasard sur chaque face	Sols qui ne se répètent pas
`scroll`	Défile, à la vitesse d'un func_conveyor	Tapis roulant
`sky`	Affiche le ciel (skybox)	Plafond extérieur

Une texture masquée laisse voir à travers ses parties bleues



texture masquée · dans le WAD et en jeu

Pour qu'une texture masquée soit transparente, mettez-la sur une entité (un `func_wall` ou un `func_illusionary`) et réglez **Render Mode = Solid** et **FX Amount = 255** (TWHL : Render Properties). Sur un bloc du monde, le bleu reste visible.

Les textures outils

Certaines textures ne s'affichent jamais : elles donnent des consignes au compilateur. Elles sont dans `zhlt.wad` et `halflife.wad` (TWHL : Tool Textures).

Texture	Visible	Solide	Rôle
NULL	Non	Oui	Supprime la face au rendu. Pour les faces que le joueur ne voit jamais
CLIP	Non	Joueurs	Mur invisible : bloque les joueurs, laisse passer les balles
ORIGIN	Non	Non	Fixe l'axe de rotation d'une entité qui tourne
AAATRIGGER	Non	Non	Texture conventionnelle des déclencheurs
SKY	Ciel	Oui	Affiche la skybox de la map
HINT + SKIP	Non	Non	Optimisation de la visibilité (voir Partie 3)

Texture	Visible	Solide	Rôle
BEVEL	Non	Oui	Corrige les accrochages sur les angles extérieurs
NOCLIP	Non	Non	Comme NULL, sans aucune collision

> **Attention :** ne mélangez pas une texture outil avec des textures normales sur un même bloc, ni deux textures outils entre elles. Les exceptions sont NULL (avec des textures normales) et le couple HINT + SKIP (VDC : Tool textures).

Exemple : un escalier à petites marches fait sautiller la vue du joueur. Posez dessus un bloc CLIP en forme de rampe, de la première à la dernière marche : le joueur glisse en douceur, mais les marches restent visibles et les balles les touchent.

Les decals

Les decals sont des images posées par-dessus les murs : traces, affiches, taches, graffitis. Ils viennent de `decals.wad`.

1. Choisissez un decal dans le navigateur de textures.
2. Prenez l'outil **Decal** (Shift+D).
3. Cliquez sur un mur dans la vue 3D : J.A.C.K. crée une entité `infodecal` à cet endroit.

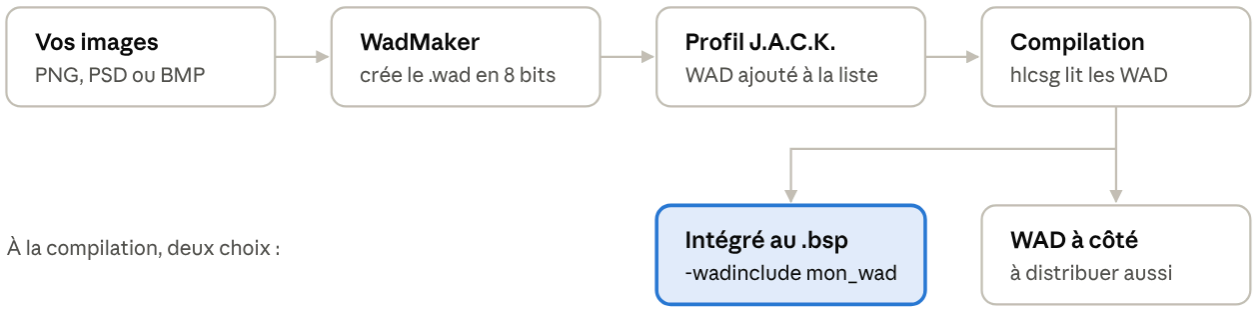
> **Info :** J.A.C.K. affiche les decals de façon plus fidèle que Hammer (VDC : J.A.C.K.). Un decal doit toucher une surface du monde : posé dans le vide, il ne s'affiche pas.

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

6. Les textures (B : créer ses textures)

Les textures du jeu ne suffisent plus ? Vous pouvez créer les vôtres, les ranger dans votre propre fichier WAD et les utiliser dans J.A.C.K. comme les autres.

Le trajet d'une texture, de l'image à la map



texture personnalisée · de l'image au .bsp

Les règles d'une texture GoldSrc

Le moteur est exigeant (the303 : GoldSrc Map Texture Tutorial) :

- **Dimensions** : multiples de 16 (64 × 64, 128 × 128, 256 × 64…). Les puissances de 2 sont les plus pratiques.
- **Taille maximale** : 512 × 512. Utilisez-la avec modération : les joueurs doivent télécharger vos textures.
- **Couleurs** : 8 bits, soit une palette de 256 couleurs par texture.
- **Nom** : 15 caractères au maximum, sans espace.

Les outils

Outil	Ce qu'il fait	Pour qui
WadMaker	Transforme un dossier d'images en WAD. Fabrique la palette 256 couleurs tout seul	Recommandé : rapide, accepte PNG, Photoshop et Krita
Wally	Éditeur de WAD historique : voir, renommer, retoucher	Pour gérer un WAD à la main
HL Texture Tools	Visionneuse et éditeur de WAD et de sprites	Alternative à Wally

> **Info :** WadMaker fonctionne en ligne de commande, sous Windows et Linux. Il sait aussi extraire les textures d'un WAD ou d'un .bsp existant (VDC : WadMaker).

Créer son WAD avec WadMaker

1. Créez un dossier, par exemple `mon_wad`, et mettez-y vos images. **Le nom de chaque fichier deviendra le nom de la texture** : respectez les préfixes du chapitre précédent (`{grille.png`, `+0ecran.png`, `+1ecran.png` …).
2. Glissez le dossier sur `WadMaker.exe`. Un fichier `mon_wad.wad` est créé à côté.
3. Si vous modifiez une image, recommencez l'opération : le WAD est reconstruit.
4. Pour des réglages particuliers (dithering, couleur de transparence), un fichier texte `wadmaker.config` dans le dossier permet de les définir texture par texture.

Les textures masquées

Une texture dont le nom commence par `{` utilise la **dernière couleur de sa palette** (index 255) comme couleur transparente (VDC : Texture prefixes). Par convention, on peint ces zones en bleu pur (0, 0, 255). Avec WadMaker, les zones transparentes d'un PNG sont gérées automatiquement.

> **Rappel :** en jeu, la transparence n'apparaît que sur une entité avec **Render Mode = Solid** et **FX Amount = 255** (chapitre 5).

Utiliser son WAD

1. Dans **Tools > Options > Game Profiles > Textures**, cliquez sur **Add WAD** et choisissez `mon_wad.wad`.
2. Vos textures apparaissent dans le navigateur, comme les autres.
3. À la compilation, décidez comment le jeu les trouvera :
   - **intégrées au .bsp** avec le paramètre `-wadinclude mon_wad` de hlcsg : la map est plus lourde, mais elle fonctionne seule ;
   - **WAD à côté** : il faut fournir le fichier `mon_wad.wad` avec la map, dans le dossier `cstrike`, sinon les joueurs auront des textures manquantes.

> **Conseil :** pour une map destinée à des serveurs, intégrez vos textures personnelles au .bsp. Ne le faites pas pour les WAD du jeu (`cstrike.wad`, `halflife.wad`) : tous les joueurs les ont déjà.

Les sons des textures

Le bruit des pas et l'impact des balles dépendent de la texture. Le jeu les lit dans le fichier `sound/materials.txt`, une ligne par texture : une lettre puis le nom (TWHL : materials.txt).

Lettre	Matériau	Lettre	Matériau
M	Métal	T	Carrelage
V	Conduit de ventilation	G	Grille
D	Terre	W	Bois
S	Liquide	P	Ordinateur
Y	Verre	N	Neige (CS uniquement)
X	Herbe (CS uniquement)		

Exemple : la ligne `W CRATE01` donne un bruit de bois à la texture `CRATE01`.

> **Attention :** Counter-Strike lit le nom complet pour les bruits de pas, mais seulement les 12 premiers caractères pour l'impact des balles (TWHL). Et `materials.txt` est commun à tout le jeu : le modifier sur un serveur change les sons pour toutes les maps. Le plus simple pour une map personnelle est de nommer ses textures comme des textures existantes du bon matériau, ou d'accepter le son par défaut (béton).

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

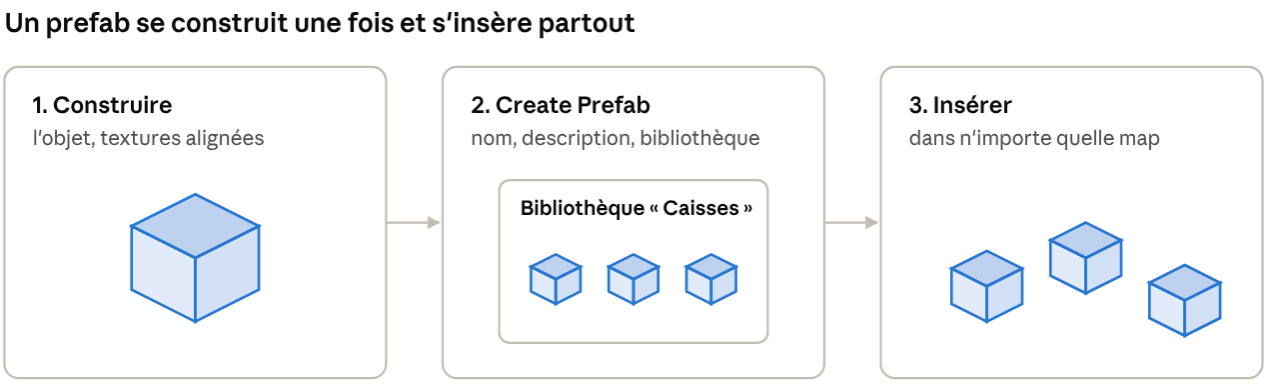
7. Les prefabs

À force de créer des blocs, vous allez vous répéter : une caisse, un lampadaire, une table reviennent dans chaque map. Les **prefabs** (objets préfabriqués) vous évitent de les refaire.

> **Question :** qu'est-ce qu'un prefab ?

Un prefab est un groupe d'objets (blocs et entités) enregistré dans une bibliothèque, que vous insérez d'un clic dans n'importe quelle map.

- **Avantage** : des objets complexes en quelques secondes, toujours identiques.
- **Inconvénient** : une map faite de prefabs téléchargés perd toute originalité. Servez-vous-en pour gagner du temps, pas pour remplacer votre travail.



prefabs · construire, enregistrer, insérer

Insérer un prefab

J.A.C.K. reprend le système de prefabs de Hammer.

1. Dans le panneau de droite, là où vous choisissez la forme des blocs, ouvrez la liste des **catégories** : en plus de « Primitives », elle contient les bibliothèques de prefabs.
2. Choisissez une bibliothèque, puis un objet dans la liste en dessous.
3. Cliquez sur **Insert original prefab** : l'objet apparaît dans la map, sous forme de groupe.
4. Déplacez-le et tournez-le comme n'importe quel groupe.

> **Info :** le nom d'un prefab indique parfois son orientation (« east », « north »…). Vous pouvez de toute façon le tourner après l'insertion.

Créer un prefab

Exemple : une caisse ouverte dans laquelle le joueur peut se cacher.

1. Créez un bloc de 64 × 64 × 64 avec une texture de caisse, et alignez la texture.
2. Faites un **Hollow** (Ctrl+Shift+H) avec une épaisseur de 4.
3. Activez **Ignore Groups** dans la barre du haut, puis sélectionnez et supprimez le mur de devant.
4. Alignez les textures à l'intérieur.
5. Sélectionnez les cinq blocs restants et cliquez sur **Create Prefab** dans le panneau de droite.
6. Donnez un nom, une description, choisissez la bibliothèque, puis **OK**.

La bibliothèque de prefabs

Le menu **Tools > Prefab Factory** ouvre le gestionnaire de prefabs : créer une bibliothèque (Add Library), ajouter, renommer, supprimer ou exporter des objets.

Télécharger des prefabs

Les sites communautaires GoldSrc (TWHL, GameBanana) proposent des prefabs et des maps d'exemple dont vous pouvez copier des objets. Une autre méthode, très simple : ouvrez une map d'exemple dans J.A.C.K., copiez l'objet (Ctrl+C) et collez-le dans votre map (Ctrl+V).

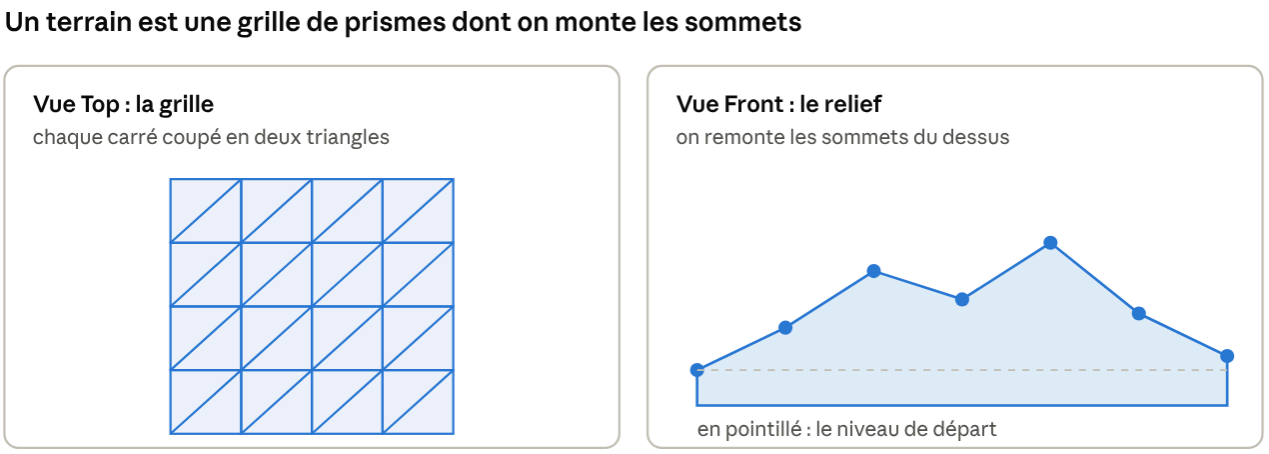
> **Erreur fréquente :** un prefab téléchargé peut contenir des blocs invalides, des textures absentes de vos WAD ou des entités propres à Half-Life. Après l'insertion, lancez **Map > Check for Problems** (Alt+P) avant de compiler.

Source : tutoriel d'origine « Une map avec Hammer ! » (OpenClassrooms, 2011), chapitre « Les prefabs », méthode reprise par J.A.C.K.

8. Décor naturel : modeler un terrain

Collines, dunes, berges : un terrain naturel n'est pas fait de blocs droits. GoldSrc n'a pas de système de terrain dédié ; on le construit avec des blocs dont on déplace les sommets.

> **Info :** les anciens générateurs (GenSurf, Terrain Generator) cités par les vieux tutoriels ne sont plus maintenus. Avec J.A.C.K., l'outil Vertex suffit pour la plupart des terrains. Et l'import de maps au format Source (VMF) ne reprend pas les « displacements », qui n'existent pas en GoldSrc (ModDB : J.A.C.K. 1.1.1058).



terrain · grille vue de dessus et relief vu de face

Préparer la grille

1. Créez un bloc de 128 × 128 unités et 32 de haut : c'est une « case » de terrain.

2. Avec l'outil **Clipping** (Shift+X), coupez-le en diagonale dans la vue Top, en mode « garder les deux ». Vous obtenez deux prismes triangulaires.
3. Sélectionnez les deux prismes et clonez-les (Shift + glisser) pour remplir une grille de 4 × 4 cases, sans espace entre elles.

> **Pourquoi des triangles ?** Une face à trois sommets est toujours plane. Si vous montez un sommet d'une case carrée, sa face du dessus se tord et le bloc devient invalide. Avec des triangles, vous pouvez monter n'importe quel sommet.

Modeler le relief

1. Sélectionnez **toute la grille**, puis prenez l'outil **Vertex** (Shift+V).
2. Dans la vue Front ou Side, sélectionnez un sommet du dessus (cadre de sélection autour) : les sommets superposés des cases voisines sont pris ensemble, la surface reste donc jointive.
3. Montez-le ou descendez-le avec la souris ou les flèches.
4. Recommencez sommet par sommet pour dessiner collines et creux.

> **Attention :** gardez la grille active (8 ou 16) pendant le modelage. Des sommets hors grille créent des fissures entre les cases, et donc des fuites.

Texturer le terrain

Sélectionnez toutes les faces du dessus avec l'outil Texture Application (Ctrl+clic), appliquez une texture d'herbe, de sable ou de terre, puis **Align : World** pour que le motif soit continu d'une case à l'autre. Mettez la texture **NULL** sur les faces de dessous, que le joueur ne verra jamais.

> **Conseil :** un terrain coûte vite cher en polygones. Préférez quelques grandes cases (128 ou 256) aux petites cases de 32 : la différence de relief se voit peu, celle de fluidité beaucoup (voir « Optimisation », Partie 3).

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

Référence J.A.C.K. : menus, barres d'outils et options

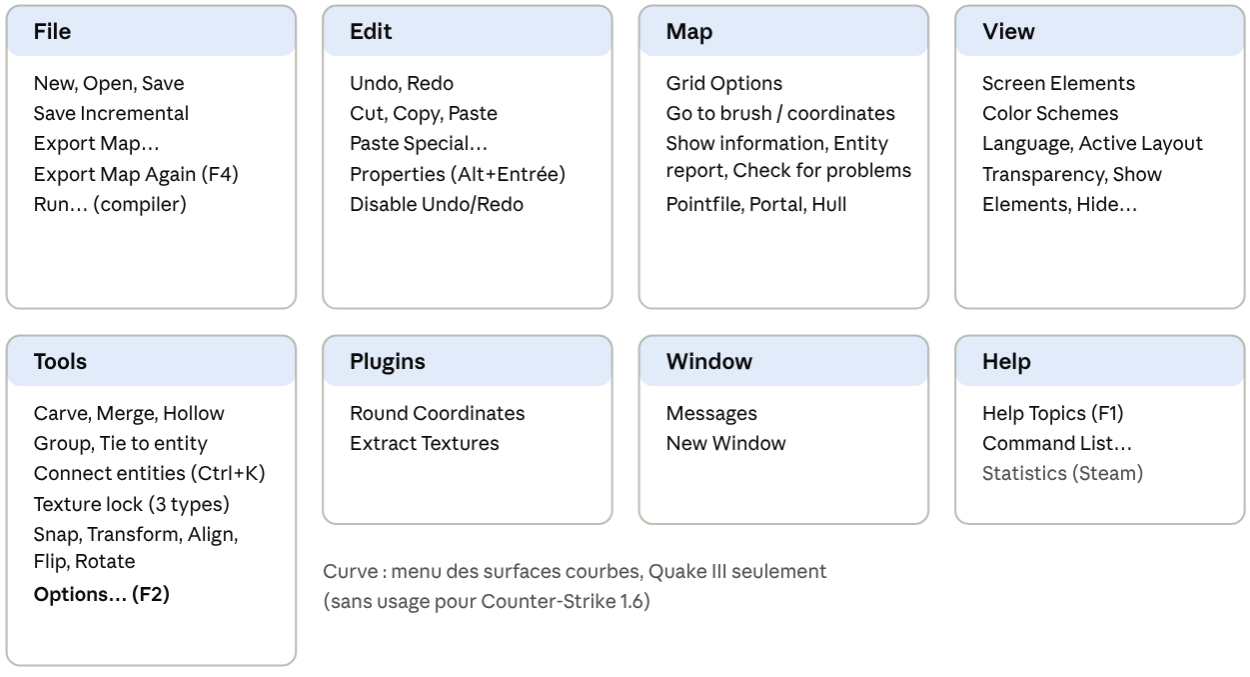
Le catalogue des commandes et réglages de l'éditeur, d'après son manuel officiel.

Cet onglet complète la Partie 1. Il s'appuie sur le **J.A.C.K. Reference Manual** (version 1.1) publié par les auteurs de l'éditeur (site officiel, 2016 ; version Steam, 2018) et sur la page des fonctionnalités du site officiel. Les noms des commandes sont laissés en anglais, comme dans l'éditeur ; J.A.C.K. existe aussi en russe, ukrainien, biélorusse et polonais (View > Language).

> **Info :** certaines parties du manuel (barres d'outils et liste complète des raccourcis) n'ont pas pu être lues en ligne. Ce qui est indiqué ici est confirmé ; la liste complète des raccourcis est dans l'éditeur, menu **Help > Command List…**.

1. Les menus

Huit menus : ce qu'on trouve dans chacun



barre de menus · 8 menus et leurs commandes principales

Les descriptions ci-dessous suivent le chapitre 5.1 du manuel. Les commandes habituelles de tout logiciel (New, Open, Save, Undo, Copy…) ne sont pas détaillées.

File

Commande	Raccourci	Rôle
Save Incremental		Enregistre en ajoutant un numéro de version au nom du fichier
Export Map…		Exporte la map au format .map, le seul que lisent les compilateurs (ils ne lisent pas le .jmf)
Export Map Again	F4	Réexporte au même endroit et sous le même nom que la dernière fois
Run…		Ouvre la fenêtre de compilation, puis lance le jeu sur la map

Edit

Commande	Raccourci	Rôle
Properties	Alt+Entrée	Propriétés du monde ou de l'entité sélectionnée
Paste Special…		Collage multiple avec décalage et rotation (section 4)
Disable / Enable Undo/Redo		Vide l'historique d'annulation pour libérer de la mémoire

Map

Commande	Raccourci	Rôle
Grid Options > Snap to Grid		Aimantation à la grille
Grid Options > Show Grid		Affiche la grille dans les vues 2D
Grid Options > Show 3D Grid		Affiche une grille sur les blocs dans la vue 3D
Grid Options > Grid Settings		Taille de la grille (plus vite avec [et])
Go to brush number…	Ctrl+Shift+G	Saute à un bloc ou une entité par son numéro
Go to coordinates…		Place la caméra à des coordonnées « X Y Z »

Commande	Raccourci	Rôle
Show information…		Statistiques de la map
Entity report…		Liste et filtre des entités (section 4)
Check for problems	Alt+P	Recherche et corrige les erreurs (section 4)
Map properties…		Propriétés du worldspawn (ciel, WAD…)
Load / Unload Pointfile		Affiche le tracé d'une fuite (.pts ou .lin)
Load / Unload Portal file		Affiche les portails de VIS (.prt)
Load / Unload Hull file		Affiche un volume de collision (.p0 à .p3), pour les problèmes de collision

View

Commande	Rôle
Screen Elements	Affiche ou masque les barres d'outils
Color Schemes	Schémas de couleurs de l'interface et des vues, prédéfinis ou personnalisés (voir section 3, « Les couleurs »)
Language	Langue de l'interface, sans redémarrer
Active Layout	Type de la vue active : X/Y, Y/Z, X/Z, ou rendu fil de fer, couleurs unies, texturé, texturé ombré
Autosize 4 Views	Donne la même taille aux quatre vues
Center 2D/3D Views on Selection	Centre toutes les vues sur la sélection
Transparency	Transparence des textures spéciales et/ou des entités
Show Elements	Liens target/targetname, models en fil de fer dans les vues 2D, ciel réaliste, animation des textures, faces des textures spéciales, grille UV
Hide Items / Hide Paths	Masque les entités d'objets et les entités de chemins
Hide Unknown Entities	Masque les entités absentes du FGD
Hide Entity Names	Masque les étiquettes de classe et de nom

Commande	Rôle
Hide Selected / Unselected Objects	Masque la sélection, ou tout le reste
Show Hidden Objects	Réaffiche tout ce qui est masqué

Tools

Commande	Raccourci	Rôle
Carve	Ctrl+Shift+C	Soustrait la forme sélectionnée des blocs autour
Merge	Ctrl+Shift+M	Fusionne en un seul des blocs qui forment ensemble un volume convexe
Make Hollow…	Ctrl+Shift+H	Creuse le bloc (épaisseur à choisir)
Group / Ungroup		Lie des objets entre eux
Make Detail / Structural		Bloc de détail, si le profil de jeu le permet
Tie to entity / Move to world		Transforme en entité, ou rend au monde
Connect entities	Ctrl+K	Crée le lien target → targetname entre deux entités sélectionnées
Texture application…	Shift+A	Fenêtre Surface Properties
Replace textures…		Chercher et remplacer des textures
Texture lock (move+rotate)		La texture suit les déplacements et rotations
Texture lock (scale)		La texture suit les redimensionnements
Texture lock (UV coordinates)		La texture reste attachée aux sommets (UV-Lock)
Align to view		Projette la texture depuis la caméra, comme un projecteur de diapositives
Snap selected to grid		Cale un coin de la sélection sur la grille ; répéter change de coin (8 coins)
Snap selected to grid individually		Cale chaque objet séparément, les entités point par leur origine
Transform		Rotation, échelle ou déplacement en valeurs exactes

Commande	Raccourci	Rôle
Align objects		Aligne les objets sur un côté de la sélection
Flip objects		Retourne horizontalement, verticalement ou les deux, selon la dernière vue 2D active
Rotate objects		Tourne dans le sens horaire ou anti-horaire, selon la dernière vue 2D active
Automatic configuration…	Ctrl+F2	Crée les profils de jeu automatiquement (version Steam)
Options…	F2	La configuration de J.A.C.K. (section 3)

Plugins, Window, Help

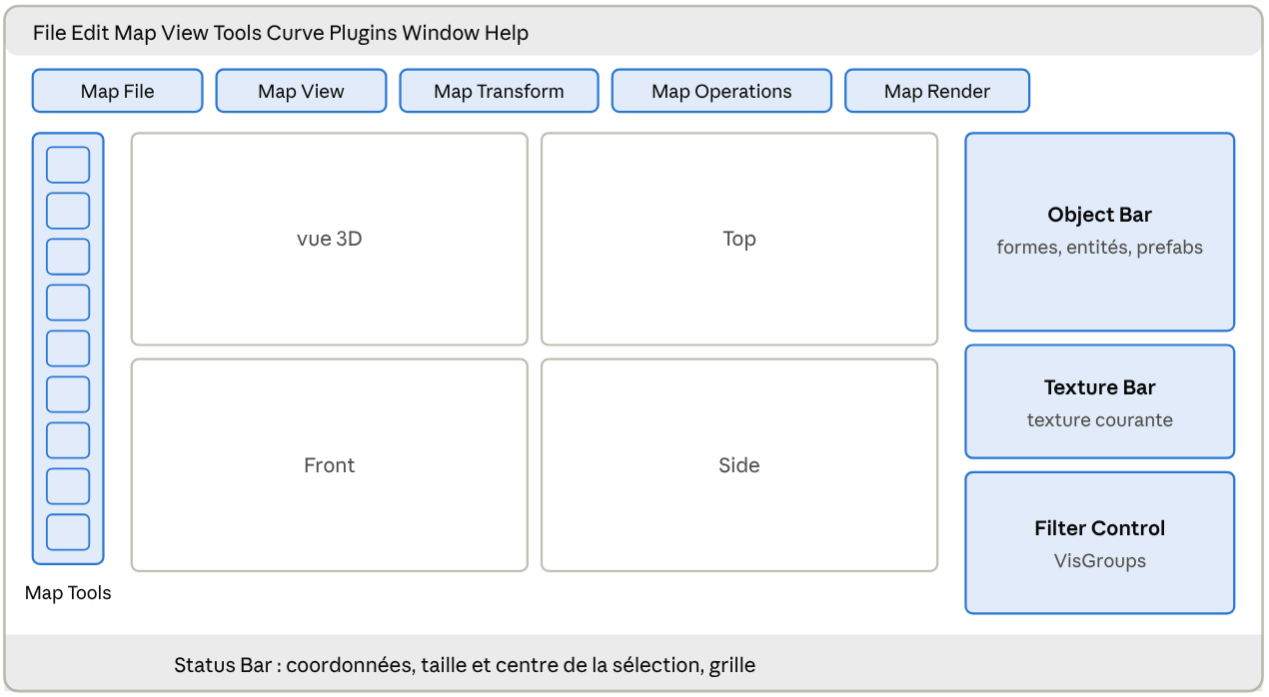
Menu	Commande	Rôle
Plugins	Round Coordinates	Arrondit les coordonnées à l'export, comme Hammer
Plugins	Extract Textures	Extrait les textures d'un .bsp (surtout utile pour Quake)
Window	Messages	Panneau des erreurs, avertissements et informations, exportable en texte
Window	New Window	Fenêtre supplémentaire (mode fenêtres indépendantes)
Help	Help Topics (F1)	Ouvre le manuel
Help	Command List…	La liste de tous les raccourcis
Help	Statistics…	Statistiques Steam (version Steam)

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

2. Les barres d'outils et panneaux

Le manuel décrit dix barres (chapitre 5.3). Elles se déplacent et s'affichent ou se masquent avec **View > Screen Elements** : le schéma montre une disposition indicative, pas forcément celle de votre écran. **Tools > Options > General > Use large map toolbar icons** agrandit leurs icônes.

L'écran de J.A.C.K. : où se trouve chaque barre



écran de J.A.C.K. · disposition indicative des barres

Barre	Ce qu'elle contient (d'après le manuel)
Map Tools	Les outils de la Partie 1, chapitre 3 : sélection, loupe, caméra, entité, bloc, application de texture, decal, clipping (cliquer de nouveau change de mode), sommets, et l'outil Path (chemins)
Map File	Les commandes de fichier
Map View	Les commandes d'affichage, dont **Load Window State**, qui recharge une disposition de fenêtres enregistrée
Map Transform	Les commandes de transformation
Map Operations	**Merge** (Ctrl+Shift+M), **Carve** (Ctrl+Shift+C), et deux bascules : sélection immédiate au cadre dans les vues 2D (sans Entrée), et sélection au cadre par la poignée centrale seulement
Map Render	Les modes de rendu, dont **Show/hide models in 2D** (models en fil de fer dans les vues 2D)

Barre	Ce qu'elle contient (d'après le manuel)
Object Bar	Les formes de blocs (Primitives) et leurs réglages, les bibliothèques de prefabs, la liste des entités à placer, **To Entity** et **To World**
Texture Bar	La texture courante et le bouton **Browse**
Filter Control	Les VisGroups (détail ci-dessous)
Status Bar	Coordonnées du curseur, taille de la sélection, position de son centre, grille

Pour Map File, Map View et Map Transform, le détail bouton par bouton n'a pas pu être lu dans le manuel en ligne : survolez chaque icône dans l'éditeur pour voir son nom.

Filter Control : les VisGroups

Les VisGroups servent à cacher ou montrer des parties de la map pendant l'édition. Un objet peut appartenir à plusieurs VisGroups.

Action	Effet
Double-clic sur un nom	Cache ou montre le groupe (icône de main)
Assign	Ajoute les objets sélectionnés au groupe choisi
Mark	Met en évidence les objets du groupe
Edit	Change le nom et la couleur du groupe
Remove	Supprime le groupe, pas ses objets
Purge	Supprime les groupes vides
Glisser un groupe sur un autre	Fusionne les deux, sous le nom du second

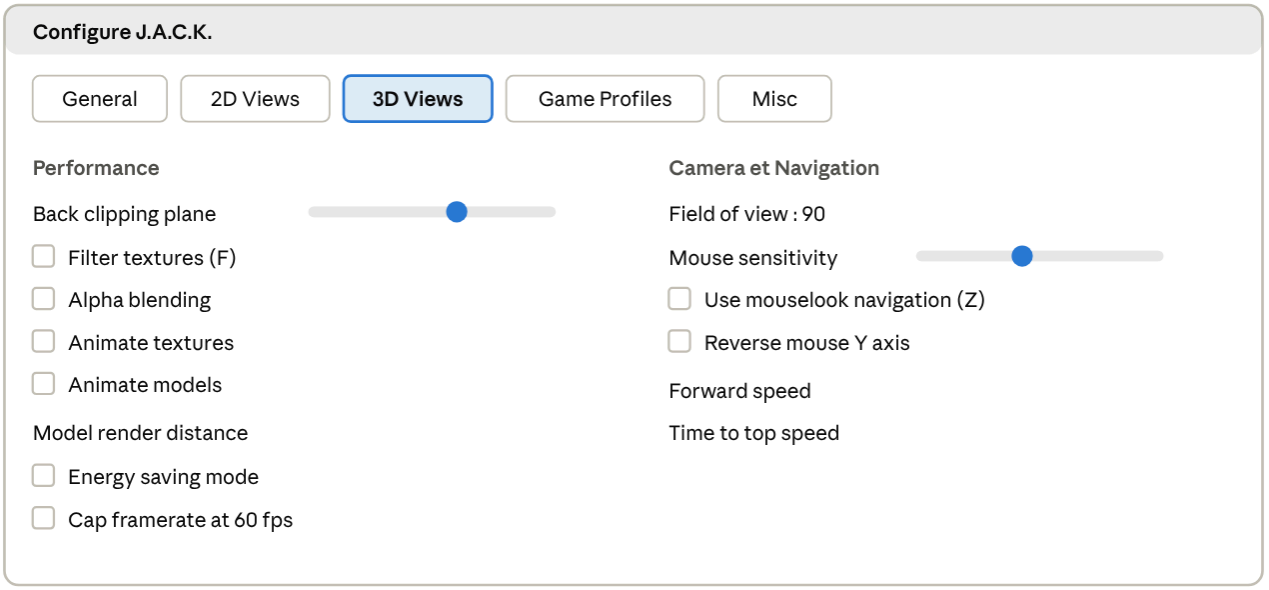
Pour créer un groupe : sélection, Alt+Entrée, onglet VisGroup, **Edit groups…** puis **New Group**.

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

3. Les options : Tools > Options (F2)

La fenêtre **Configure J.A.C.K.** a cinq onglets : **General**, **2D Views**, **3D Views**, **Game Profiles** et **Misc**. Seul Game Profiles est indispensable au début ; les autres peuvent rester aux réglages par défaut (manuel, chapitres 2.1 et 5.2).

F2 ouvre Configure J.A.C.K. : cinq onglets, ici 3D Views



Configure J.A.C.K. · onglet 3D Views

Onglet General

Option	Effet
Independent window configurations	Disposition libre des fenêtres ; on en ajoute par le menu Window
Load default window positions with maps	Recharge la dernière disposition à l'ouverture d'une map (sinon : bouton Load Window State)
Maximum world size (units)	Taille de l'espace de construction ; c'est aussi une limite des compilateurs (8192 pour Half-Life)
Cache game data resources on demand	Charge les ressources d'un profil seulement à sa première utilisation (recommandé)
Load resources from archives	Lit les models et sprites dans les archives .pak/.pk3
Use large map toolbar icons	Grandes icônes dans les barres d'outils

Option	Effet
Transform angles of point entities when rotating	Tourner une sélection tourne aussi l'angle des entités point
Snap selection to grid when dragging	Déplacements calés sur la grille (sinon au pixel)
Don't snap primitive points to integer grid	Les sommets des nouvelles formes ne sont pas arrondis
Stretch arches to fit original bounding rectangle	Les arches remplissent tout le rectangle tracé
Preserve internal connections in cloned entities	Les liens target/targetname restent internes aux copies
Restore cut objects if not pasted	Un objet coupé (Ctrl+X) et jamais collé réapparaît
Precise selection of models	Sélection sur la forme exacte du model, pas sa boîte
ALT+drag to scale models	Agrandir un model avec Alt + glisser (selon le jeu)
ESC to clear selection	Échap vide la sélection
Memory usage	Curseur mémoire / qualité des textures ; 4e cran depuis la gauche recommandé
Undo levels	Nombre d'annulations (50 par défaut)

Onglet 2D Views

Option	Effet
Crosshair cursor	Curseur en croix au-dessus des blocs
Default to fixed-degree rotation	Rotation par pas fixes (15°) au lieu de 1°
Display scrollbars	Barres de défilement (sinon : Espace + glisser)
Draw vertices	Affiche les sommets des blocs
Draw camera	Affiche la caméra 3D (un œil bleu)
Draw models	Dessine les models au lieu de leur boîte

Option	Effet
Draw sizing info	Affiche les dimensions près de la sélection
Center on camera after movement in 3D	Recentre les vues 2D sur la caméra après un déplacement en 3D
Use VisGroup colors for object lines	Lignes des blocs dans la couleur de leur VisGroup
Keep VisGroup when clone-dragging	Les clones gardent le VisGroup de l'original
Arrow keys nudge selected object/vertex	Les flèches déplacent la sélection
Reorient primitives on creation in the active 2D view	Oriente les nouvelles formes selon la vue où on les trace
Automatic infinite selection in 2D windows (no ENTER)	La sélection au cadre se fait sans Entrée
Selection box selects by center handles only	Le cadre ne prend que les objets dont le centre est dedans
Grid : Size	Grille par défaut des nouvelles maps, de 1 à 512
Grid : Highlight every 8 grid lines	Une ligne plus marquée toutes les N lignes (N réglable)
Grid : Hide small grid	Masque la grille quand elle fait moins de 4 pixels
Grid : Dotted grid	Grille en points au lieu de lignes

Onglet 3D Views

Option	Effet
Back clipping plane	Distance au-delà de laquelle rien n'est dessiné ; plus bas = plus fluide
Filter textures	Lissage des textures ; touche **F** dans la vue 3D
Alpha blending	Affiche la transparence
Animate textures	Anime les textures et effets
Animate models	Anime les models
Model render distance	Au-delà, les models sont remplacés par leur boîte

Option	Effet
Energy saving mode	Consomme moins, déplacements parfois moins fluides
Cap framerate at 60 fps	Limite à 60 images par seconde
Multisample antialiasing (MSAA)	Anticrénelage
Field of view	Angle de vue de la caméra : 90 par défaut, 80 à 120 conseillé
Mouse sensitivity	Sensibilité de la souris
Use mouselook navigation	Vol libre (touche Z, puis W/A/S/D)
Reverse mouse Y axis	Inverse le haut et le bas
Forward speed / Time to top speed	Vitesse maximale et temps pour l'atteindre

Onglet Game Profiles

Chaque profil a quatre sous-onglets (Partie 1, chapitre 1) :

Sous-onglet	Réglages
Game	Fichiers FGD, type de map, entités point et bloc par défaut
Directories	Exécutable du jeu, Base Game Directory, Mod Directory, Source Maps Directory, visionneuse de models (Model Viewer)
Textures	Fichiers de textures (Add Files…), luminosité des textures, échelle par défaut (1.0), fichier de palette
Build Programs	Chemins et paramètres de CSG, BSP, VIS et LIGHT (RAD)

Onglet Misc

Option	Effet
Backup existing files on save	Garde une copie à chaque sauvegarde (.jmf devient .jmx)

Option	Effet
Enable autosave	Sauvegarde automatique, toutes les 5 minutes par défaut (réglable)
Keep the last N autosaves	Nombre de sauvegardes gardées (0 = illimité)
Autosave modified files only	Pas de sauvegarde si rien n'a changé
Beep on successful autosave	Bip à chaque sauvegarde
Use directory of the current file	Sauvegardes à côté de la map (sinon : dossier au choix)
Steam Cloud (version Steam)	Stockage en ligne des maps, des sauvegardes et des copies, synchronisation au démarrage

Les couleurs (fond de la vue 3D, grille, sélection…)

Les couleurs ne sont pas dans la fenêtre Options, mais dans **View > Color Schemes** : J.A.C.K. propose des schémas de couleurs tout faits, et une commande pour définir le sien, avec les couleurs des vues et des fenêtres (manuel, 5.1 ; site officiel). C'est là que se change, par exemple, la couleur de fond des vues.

> **Info :** les vues 2D peuvent aussi afficher une **image de fond** (BMP ou PNG), avec décalage, échelle, luminosité, lissage et inversion des couleurs réglables ; ces réglages sont enregistrés dans le .jmf (site officiel). Pratique pour reproduire un plan.

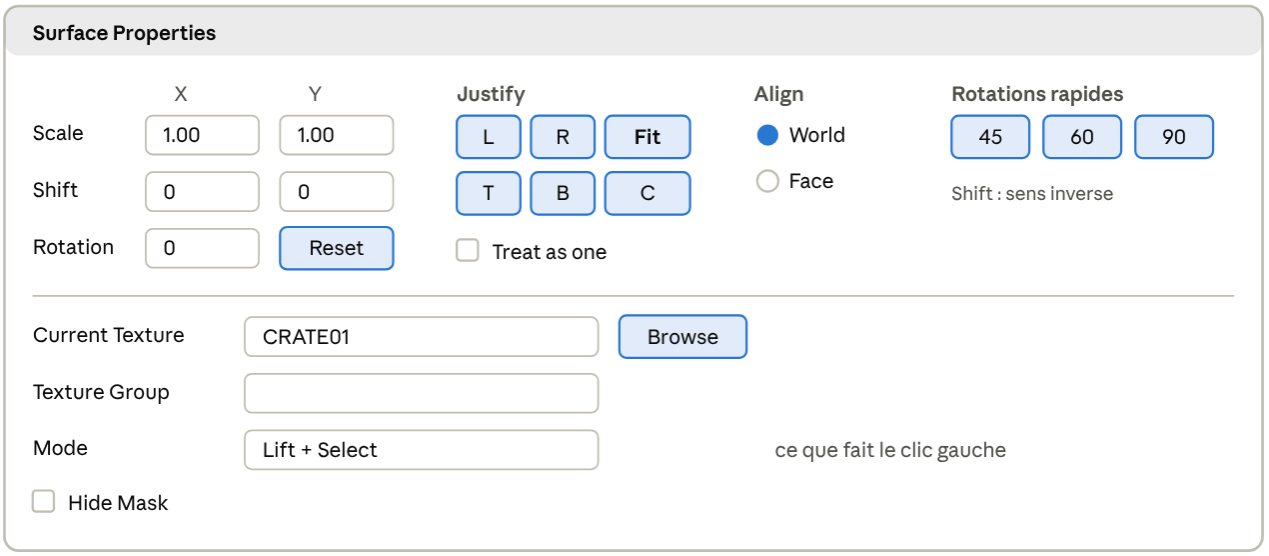
> **Info :** certains réglages avancés, comme l'affichage des valeurs dans le mode SmartEdit, se trouvent dans le fichier `VDKSettings.ini` (site officiel).

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

4. Les autres fenêtres

Surface Properties (Shift+A)

Shift+A ouvre Surface Properties : tout le réglage d'une face



Surface Properties · champs et boutons

Contrôle	Effet
Scale X / Y	Taille de la texture ; une valeur négative la retourne en miroir
Shift X / Y	Décalage
Rotation, Reset	Angle ; Reset remet échelle, décalage et rotation à zéro
Justify : L, R, Fit, T, B, C	Cale à gauche, à droite, ajuste, en haut, en bas, au centre
Treat as one	Traite les faces sélectionnées comme une seule
Align : World / Face	Projection selon le monde ou selon la face
45, 60, 90	Rotations rapides ; avec Shift, dans l'autre sens
Texture Group, Current Texture	Groupe de textures et texture courante
Hide Mask	Masque la teinte rouge des faces sélectionnées
Mode	Ce que fait le clic gauche : Lift + Select, Lift, Select, Apply (texture only), Apply (texture + values)

J.A.C.K. ajoute les modes **NULL to Selected**, **NULL to Unselected** et **Apply (texture + values + axes)** (site officiel).

Object Properties (Alt+Entrée)

- **Class Info** : les attributs, le bouton **SmartEdit**, la boîte Yaw et sa boussole, un bouton **Help**, et des boutons pour parcourir les sprites, sons et models.
- **Flags** : les cases à cocher ; leur valeur numérique s'affiche au survol.
- **VisGroup** : appartenance aux VisGroups, et accès à la fenêtre de gestion des groupes.

Paste Special (Edit)

Champ	Effet
Number of copies	Nombre de copies
Start at center of original	Les copies partent du centre de l'original
Keep internal connections	Les liens target/targetname restent internes à chaque copie
Group copies	Regroupe les copies
Offset (X, Y, Z) et Rotation (X, Y, Z)	Décalage et rotation entre deux copies ; les boutons « < » recopient une dimension de la sélection

Replace Textures (Tools)

- **Find / Replace** : textures à chercher et à mettre, avec aperçu.
- **Replace in** : toute la map ou seulement les objets marqués, objets cachés inclus ou non.
- **Action** : Replace exact matches, Replace partial matches, Substitute partial matches, ou Do not replace textures (seulement marquer les blocs trouvés).
- **Modify texture scale by factor** : adapte l'échelle quand la nouvelle texture n'a pas la même taille.

Hollow (Ctrl+Shift+H)

Un seul champ, l'épaisseur des murs. Une valeur **négative** creuse vers l'extérieur. Une confirmation est demandée si plusieurs blocs sont sélectionnés.

Check for Problems (Alt+P)

Liste les erreurs de la map (par exemple : pas de `info_player_start`). Boutons : **Go to error** (centre les vues dessus), **Fix** (corrige celle-ci), **Fix all (of type)** (toutes celles du même type), **Fix all errors**.

Entity Report (Map)

Liste les entités avec leur classe, leur nom et leur cible. Boutons **Mark**, **Go to**, **Delete**, **Properties**, **Hide**, **Unhide** (les entités cachées sont en italique). Filtres : type (toutes, bloc, point), entités cachées, recherche par clé/valeur (avec **Exact**) et par classe.

Les petites fenêtres

Fenêtre	Usage
Go to Brush (Ctrl+Shift+G)	Numéro d'entité et de bloc ; cochez « Search visible brushes only » si la map a été exportée avec « Save visible objects only »
Go to Coordinates	Coordonnées sans virgule, par exemple `304 -384 0`
Map Information	Statistiques de la map
Entity Help	Aide de l'entité, générée depuis le FGD
Default Room Properties	À la création d'une map, une première pièce : taille, textures des murs, du sol et du plafond, option « Apply NULL to backfaces ». Shift + clic sur New la rouvre
Transformation	Déplacement, rotation ou échelle en valeurs exactes
Automatic Configuration (Ctrl+F2, Steam)	Crée les profils des jeux installés ; « Remove all existing game configurations » repart de zéro (irréversible)

Sources : J.A.C.K. Reference Manual, chapitres 3.4 et 5.2, J.A.C.K. : Features.

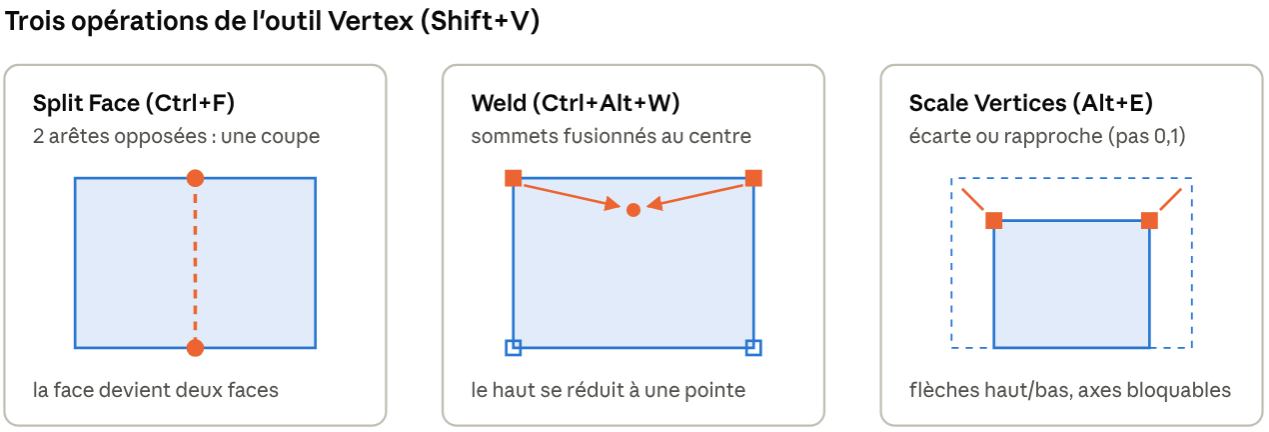
5. Les outils avancés

Caméras et navigation dans la vue 3D

Action	Commande
Outil caméra	Shift+C
Créer une caméra	Shift + glisser dans une vue 2D (le point = l'œil, la ligne = la direction)
Caméra suivante / précédente	Page suivante / Page précédente (outil caméra actif)
Supprimer la caméra courante	Suppr
Déplacer œil et direction ensemble	Ctrl + glisser
Vol libre	Z, puis souris + W, A, S, D
Tourner la vue sur place	Glisser avec le bouton gauche
Monter, descendre, décaler	Glisser avec le bouton droit
Se décaler sur le côté	Glisser avec les deux boutons
Avancer, reculer	Shift + glisser avec le bouton droit

J.A.C.K. enregistre les caméras dans le fichier de la map (.jmf, .rmf, .vmf) (site officiel).

L'outil Vertex



outil Vertex · couper, fusionner, échelle

- **Shift+V** répété fait défiler trois modes : sommets et arêtes (par défaut), sommets seuls, arêtes seules.

- **Alt+clic** évite de désélectionner le bloc.
- Pour annuler (Ctrl+Z), il faut d'abord quitter l'outil Vertex.
- J.A.C.K. **empêche les formes invalides** (non convexes) : le déplacement devient plus difficile quand on s'en approche.
- **Split Face** (Ctrl+F) coupe une face entre deux arêtes opposées sélectionnées ; **Weld Vertices** (Ctrl+Alt+W) fusionne les sommets sélectionnés en leur centre ; **Scale Vertices** (Alt+E) écarte ou rapproche les sommets, par pas de 0,1 avec les flèches, avec des cases pour bloquer certains axes.
- Le menu contextuel (clic droit) propose aussi **Triangulate**, qui découpe une face tordue en triangles plans.

Clipping, Merge et Carve

- **Clipping** (Shift+X) : Shift+X ou un nouveau clic sur le bouton change de mode ; Ctrl + glisser déplace toute la ligne de coupe ; Entrée coupe ; la touche **O** dans une vue 2D affiche la taille des morceaux.
- **Merge** (Ctrl+Shift+M) : fusionne plusieurs blocs en un seul, s'ils forment ensemble un volume convexe.
- **Carve** (Ctrl+Shift+C) : faites d'abord les petits trous carrés, puis les ronds ; évitez de découper avec des formes à nombreuses faces (arches, sphères).

L'outil Arch

Choisissez la forme **arch** dans l'Object Bar et tracez un rectangle : une fenêtre de réglages s'ouvre.

Réglage	Effet
Wall width	Épaisseur de chaque morceau
Number of sides	De 3 à 100 morceaux
Arc	De 8 à 360 degrés ; le bouton **Circle** met 360
Start angle	Angle de départ ; 0 = en haut, sens horaire
Add height	Hauteur ajoutée à chaque morceau : donne un escalier en colimaçon
Group parts	Groupe les morceaux
Fit selection	Ajuste les morceaux à tout le rectangle
Triangulate	Prismes triangulaires au lieu de quadrilatères

Réglage	Effet
Smooth slope	Rampe continue au lieu de marches (avec Add height)

Exemple : un escalier en colimaçon. Arc = 360, Number of sides = 16, Add height = 16, Group parts coché.

L'outil Path (chemins de path_corner)

Action	Commande
Premier point	Shift + clic dans une vue 2D ; une fenêtre demande le nom, le sens (aller simple, circulaire, aller-retour) et l'entité (en général `path_corner`)
Ajouter un point en fin	Shift + clic sur un point existant
Insérer un point avant	Sélectionner un point, Inser
Supprimer des points	Suppr ; Shift+Suppr supprime tout le chemin
Convertir en entités liées	Ctrl+Shift+E (sans retour possible)

C'est la façon la plus rapide de préparer le parcours d'un `func_train` (Partie 2, chapitre 16).

Les models

- Avec une entité qui accepte un model, le bouton « … » dans ses propriétés parcourt les fichiers .mdl ; groupes de corps (body) et skins s'affichent en aperçu.
- Clic droit sur un model : **View model…** l'ouvre dans la visionneuse configurée dans le profil, **Reload model** le recharge depuis le disque.
- Avec l'option **ALT+drag to scale models**, Alt + glisser agrandit un model, si le jeu le permet.

Relier deux entités en un geste

Sélectionnez l'entité qui appelle, puis celle qui est appelée, et faites **Ctrl+K** (Tools > Connect entities) : J.A.C.K. remplit le Target de la première avec le Name de la seconde.

Sources : J.A.C.K. Reference Manual, chapitres 3 et 4, J.A.C.K. : Features.

Partie 2 : Entités et effets spéciaux

Donnez vie à votre map : lumières, portes, déclencheurs, effets et modes de jeu.

1. Les entités

Les blocs forment le squelette de la map ; les entités en sont les organes. Tout ce qui bouge, éclaire, fait du bruit ou réagit au joueur est une entité.

Entités point et entités bloc

	Entité point	Entité bloc
Ce que c'est	Un objet placé à un point précis, sans volume	Un ou plusieurs blocs auxquels on donne un rôle
Comment la créer	Outil **Entity** (Shift+E), clic dans une vue 2D, Entrée	Sélectionner les blocs, puis **To Entity** (Ctrl+T)
Exemples	`light`, `info_player_start`, `ambient_generic`	`func_door`, `func_buyzone`, `trigger_once`
Pour la retirer	Suppr	**To World** (Ctrl+W) rend les blocs au monde

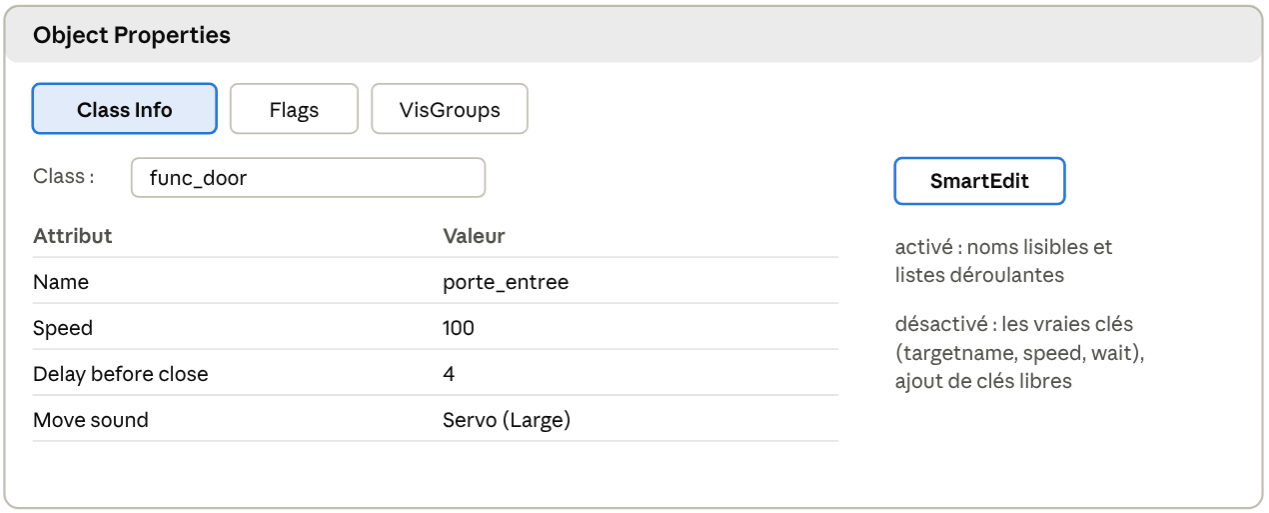
> **Info :** un clic droit dans une vue 2D ouvre dans J.A.C.K. un menu de création d'entités, plus rapide que la liste du panneau de droite (VDC : J.A.C.K.).

> **Attention :** une entité bloc n'est pas un bloc du monde. Elle ne ferme pas la map (le vide reste du vide derrière une porte) et, par défaut, elle ne projette pas d'ombre.

Les propriétés d'une entité

Sélectionnez l'entité et appuyez sur **Alt+Entrée**.

Alt+Entrée ouvre les propriétés : classe, paramètres et flags



propriétés d'une entité · fenêtre Object Properties

- **Class** : le type d'entité. Vous pouvez le changer ici : un `func_wall` devient un `func_door` sans refaire les blocs.
- **Les attributs** (keyvalues) : chaque entité a les siens, définis par le FGD.
- **Flags** : des cases à cocher qui modifient le comportement (« Use only », « Starts open »…).
- **SmartEdit** : activé, l'éditeur affiche des noms lisibles et des listes déroulantes. Désactivé, il montre les vraies clés et permet d'en ajouter à la main, ce qui est nécessaire pour le `multi_manager` ou les clés des compilateurs (`zhlt_…`).

J.A.C.K. améliore le mode SmartEdit avec un aperçu des valeurs choisies (VDC : J.A.C.K.).

Les attributs que vous retrouverez partout

Attribut	Clé	Rôle
Name	`targetname`	Le nom de l'entité, pour qu'une autre puisse l'appeler
Target	`target`	Le nom de l'entité à appeler
Delay before trigger	`delay`	Attente en secondes avant d'appeler la cible
Kill target	`killtarget`	Entité à supprimer au moment de l'appel
Master	`master`	Entité qui autorise ou bloque celle-ci (chapitre 5)

Attribut	Clé	Rôle
Pitch Yaw Roll	`angles`	L'orientation
Render FX, Render Mode, FX Amount, FX Color	`renderfx`, `rendermode`, `renderamt`, `rendercolor`	L'apparence (annexe « Propriétés de rendu », Partie 4)

Le couple **Name / Target** est le cœur de la logique d'une map : il fait l'objet des chapitres 4 et 5.

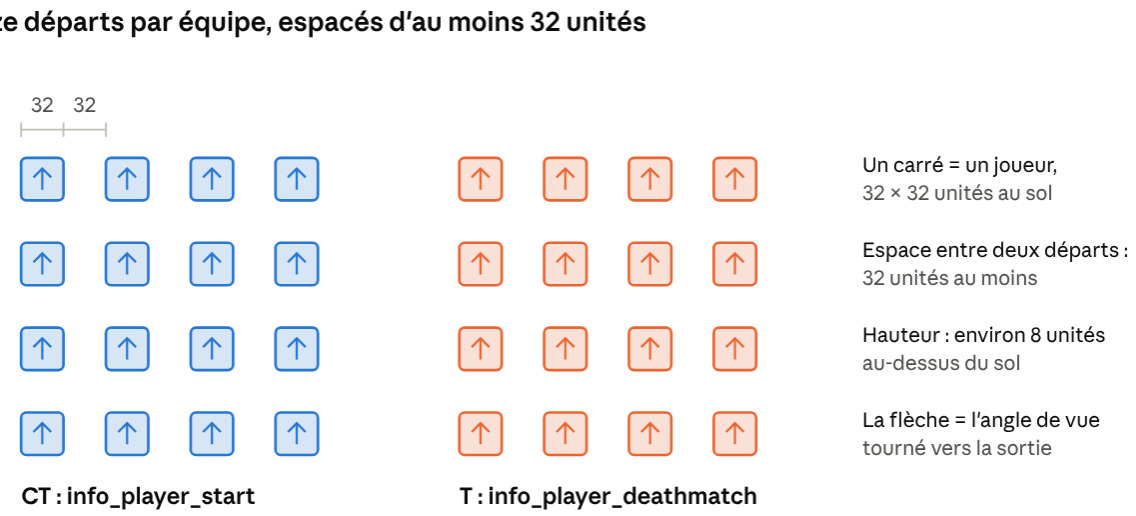
Sources : VDC : List of Counter-Strike entities, VDC : J.A.C.K..

2. Position de départ des joueurs

Entités concernées : `info_player_start`, `info_player_deathmatch` · **Type :** entité point · **Difficulté :** facile

Sans point de départ, personne ne peut apparaître dans votre map. Dans Counter-Strike, chaque équipe a le sien (TWHL : Introduction to Counter-Strike mapping) :

- `info_player_start` : départ des **Anti-Terroristes** (CT) ;
- `info_player_deathmatch` : départ des **Terroristes** (T).



zone de départ · 16 CT et 16 T vus de dessus

Placer les départs

1. Prenez l'outil **Entity**, choisissez `info_player_start` et cliquez dans la vue Top, à l'endroit voulu.
2. Dans la vue Front, placez-le environ 8 unités au-dessus du sol.

3. Dans ses propriétés, réglez l'angle (le petit cadran **Angles**) pour que les joueurs regardent vers la sortie de la zone.
4. Clonez-le (Shift + glisser) pour en faire une grille, en laissant au moins 32 unités entre deux départs.
5. Faites de même avec `info_player_deathmatch` de l'autre côté de la map.

Combien en mettre ?

Environ **16 par équipe**, soit 32 au total : c'est le maximum que le jeu sait gérer (TWHL). Si un serveur reçoit plus de joueurs qu'il n'y a de départs dans une équipe, les joueurs en trop ne peuvent pas apparaître.

> **Erreur fréquente :** des départs trop proches, ou qui touchent un mur ou le plafond, font apparaître des joueurs coincés, ou qui s'entretuent en apparaissant. Laissez de l'espace, et vérifiez qu'aucun départ ne traverse un bloc.

> **Info :** sans entité `func_buyzone`, une petite zone d'achat est créée automatiquement autour de chaque départ. Pour en décider vous-même, voir le chapitre 20, « Les modes de jeu Counter-Strike ».

Tester

Lancez la map avec des bots (chapitre « Les bots et waypoints », Partie 4) et rechargez-la plusieurs fois pleine : c'est la seule façon de voir si tous les départs sont sûrs.

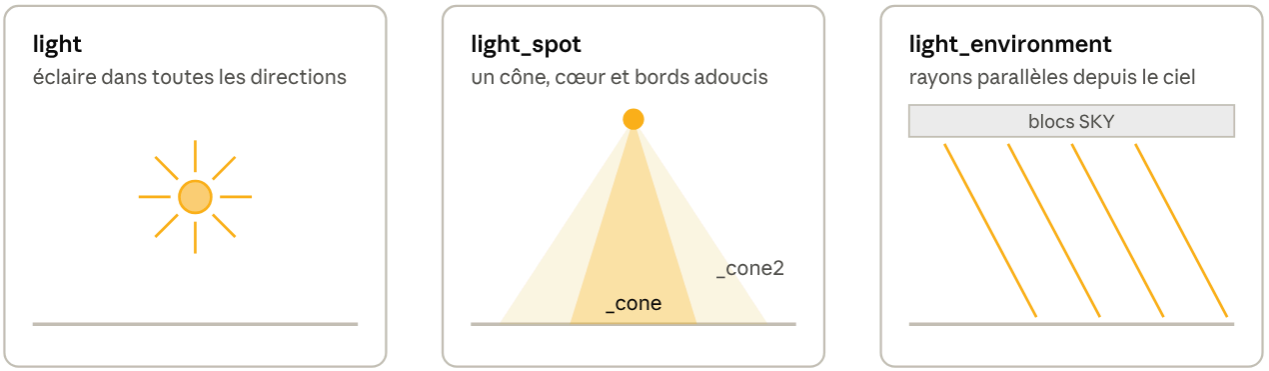
Source : TWHL : Introduction to Counter-Strike mapping.

3. Lumière !

Entités concernées : `light`, `light_spot`, `light_environment` · **Type :** entité point · **Difficulté :** facile

Une map sans lumière est entièrement noire (ou entièrement claire, si elle fuit). L'éclairage est calculé une fois pour toutes à la compilation, par le programme RAD : il ne change pas en jeu, sauf pour les lumières commutables.

Trois façons d'éclairer : une ampoule, un projecteur, le soleil



éclairage · light, light_spot, light_environment

Une lampe : `light`

Placez l'entité sous un plafond, près d'un objet qui justifie la lumière (néon, lampe).

- **Brightness** : quatre nombres, `R V B Intensité`. Par exemple `255 255 200 250` donne une lumière blanc chaud d'intensité 250. L'intensité peut dépasser 255.
- **Appearance** (style) : clignotement, pulsation, bougie, néon défectueux…
- **Name** : une lumière nommée devient **commutable** : un bouton ou un déclencheur peut l'allumer et l'éteindre.
- Flag **Initially dark** : la lumière commence éteinte.

Le soleil : `light_environment`

Il éclaire toutes les surfaces qui voient le ciel, par des rayons parallèles, depuis les blocs texturés **SKY**. Il n'en faut **qu'un seul par map**.

- **Pitch** : l'inclinaison des rayons, de -90 (à la verticale, vers le bas) à 0 (à l'horizontale).
- **Angles** (yaw) : la direction d'où vient le soleil.
- **Brightness** : couleur et intensité du soleil, au même format que `light`.

> **Attention :** sans bloc SKY visible, le `light_environment` n'éclaire rien. C'est l'erreur classique d'une map extérieure toute noire.

Les spots : `light_spot`

Un projecteur qui éclaire dans un cône (TWHL : light_spot).

- **Pitch** : de -90 (vers le bas) à 90 (vers le haut).
- **Inner Radius** (`_cone`) : l'angle du cœur, le plus lumineux.

- **Outer Radius** (`_cone2`) : l'angle où la lumière s'estompe. Au-delà de 90, la valeur n'a plus d'effet.
- **Target** : vise une entité (de préférence un `info_null`) au lieu de régler l'angle à la main.
- **Is Sky** : réglé à Yes, le spot se comporte comme un `light_environment`.

Les textures lumineuses

Certaines textures émettent elles-mêmes de la lumière : néons, écrans, dalles de plafond. La liste est dans le fichier `lights.rad`, placé à côté des compilateurs et lu par RAD : chaque ligne associe un nom de texture à une couleur et une intensité (TWHL : Compiling 101). Si ce fichier est vide ou absent, ces textures n'éclairent rien.

Allumer et éteindre une lumière

Exemple d'interrupteur :

1. Placez une `light`, nommez-la `lampe_salle`.
2. Créez un petit bloc sur le mur et faites-en un `func_button` (Ctrl+T).
3. Dans le bouton : **Target** = `lampe_salle`, **Delay before reset** = 1, flag **Don't move** coché.
4. Compilez : chaque appui sur le bouton allume ou éteint la lampe.

> **Attention :** une map peut contenir au maximum **31 lumières nommées différentes**, et une même face ne peut recevoir que 4 styles de lumière (TWHL : light_spot). Donnez le même nom à toutes les lampes d'une même pièce si elles doivent s'allumer ensemble.

Sources : TWHL : light_spot, TWHL : Compiling 101.

4. Les déclencheurs (A : simples)

Entités concernées : `trigger_once`, `trigger_multiple`, `trigger_auto`, `func_button`, `func_rot_button` · **Type :** entités bloc et point · **Difficulté :** facile

Une porte qui s'ouvre, une lumière qui s'allume, une explosion : tout commence par un déclencheur qui « appelle » une autre entité.

Les appels d'entités



Le joueur appuie sur le bouton, la porte s'ouvre, et en s'ouvrant elle allume la lampe.

appel d'entités · Target vers Name, en chaîne

Une entité en appelle une autre grâce à deux champs :

- **Name** (`targetname`) : le nom de l'entité appelée ;
- **Target** (`target`) : le nom de l'entité à appeler.

Quand le déclencheur se déclenche, il cherche toutes les entités dont le **Name** est égal à son **Target**, et les active. Une porte s'ouvre (ou se ferme), une lumière nommée s'allume (ou s'éteint), un son se joue. Plusieurs entités peuvent porter le même nom : elles réagiront toutes ensemble.

Trois autres champs complètent l'appel :

- **Delay before trigger** (`delay`) : attente, en secondes, avant d'appeler la cible ;
- **Kill target** (`killtarget`) : nom d'une entité à **supprimer** au moment de l'appel ;
- **Master** : nom d'un `multisource` qui autorise ou bloque l'entité (chapitre suivant).

> **Conseil :** donnez des noms parlants et sans espace : `porte_bombe_a`, `lampe_couloir`. Avec 50 entités, `door1` et `door2` deviennent vite un casse-tête. J.A.C.K. dessine les liens entre entités dans les vues, ce qui aide à s'y retrouver.

Au premier passage : `trigger_once`

Un volume invisible qui déclenche sa cible la première fois qu'un joueur y entre, puis disparaît.

1. Créez un bloc, texturez-le avec **AAATRIGGER**.
2. Ctrl+T, classe `trigger_once` .
3. **Target** : le nom de l'entité à déclencher.

La texture AAATRIGGER est conventionnelle : les déclencheurs sont invisibles en jeu quelle que soit leur texture, mais elle permet de les repérer dans l'éditeur (VDC : Tool textures).

À chaque passage : `trigger_multiple`

Même chose que `trigger_once`, mais il se réarme. L'attribut **Delay before reset** (`wait`) fixe le temps, en secondes, avant qu'il puisse se déclencher de nouveau.

Au chargement de la map : `trigger_auto`

Une entité point qui déclenche sa cible toute seule quand la map démarre, après le délai indiqué. Utile pour lancer une musique d'ambiance, allumer un décor ou démarrer un train.

Les boutons : `func_button`

Un bloc que le joueur active en appuyant sur la touche **Utiliser** (E), ou en le touchant si le flag le permet.

Attribut ou flag	Rôle
Target	L'entité appelée
Delay before reset	Secondes avant que le bouton revienne. -1 = reste enfoncé pour toujours
Sounds	Le bruit du bouton
Speed, Lip	Vitesse et distance d'enfoncement
Flag **Don't move**	Le bouton ne s'enfonce pas (interrupteur mural)
Flag **Toggle**	Un appui pour activer, un second pour désactiver
Flag **Sparks**	Projette des étincelles
Flag **Touch Activates**	Il suffit de toucher le bouton

Pour qu'un bouton change d'aspect quand on l'active, donnez à une de ses faces une texture `+0…` qui a une version `+A…` : le bouton passe de l'une à l'autre (VDC : Texture prefixes, TWHL : Doors).

Les boutons rotatifs : `func_rot_button`

Un levier ou une manivelle qui tourne au lieu de s'enfoncer. Comme toute entité qui tourne, il lui faut un bloc **ORIGIN**, lié avec lui, à l'endroit de l'axe. L'attribut **Distance** donne l'angle de rotation en degrés.

> **Exemple complet :** un bouton qui ouvre une porte. Faites la porte (`func_door` , Name = `porte_1`), puis le bouton (`func_button` , Target = `porte_1` , Delay before reset = 3, flag Don't move). Le détail des portes est au chapitre 8.

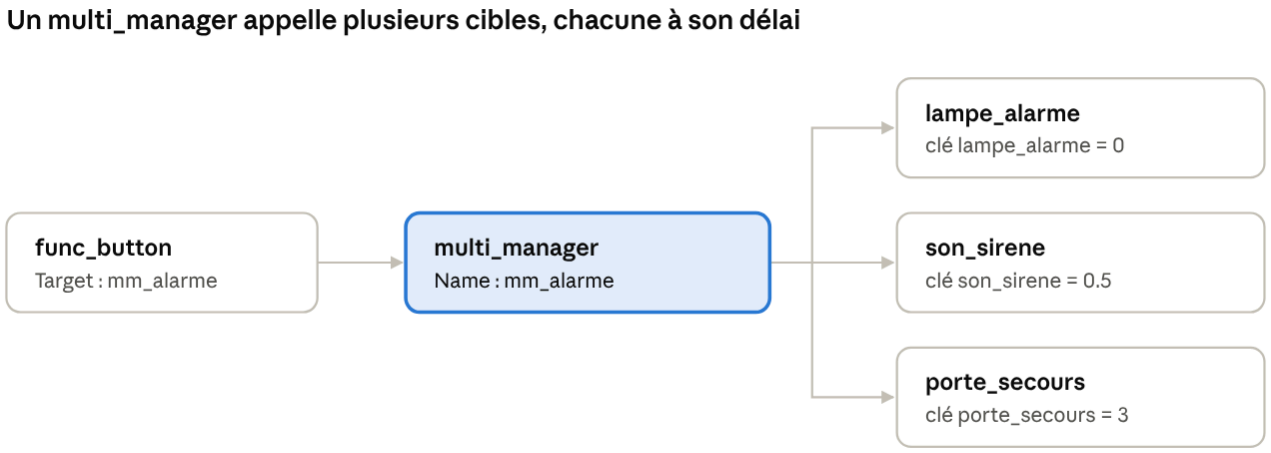
Sources : TWHL : Doors, VDC : Texture prefixes, VDC : Tool textures (GoldSrc).

5. Les déclencheurs (B : complexes)

Entités concernées : `multi_manager` , `multisource` , `trigger_relay` , `trigger_changetarget` , `trigger_counter` · **Type :** entités point (sauf `trigger_counter` , entité bloc) · **Difficulté :** moyen

Un déclencheur simple appelle une cible à la fois. Pour une vraie séquence (une alarme qui allume des lampes, fait hurler une sirène puis ouvre une porte), il faut des entités logiques.

Appeler plusieurs entités dans l'ordre : `multi_manager`



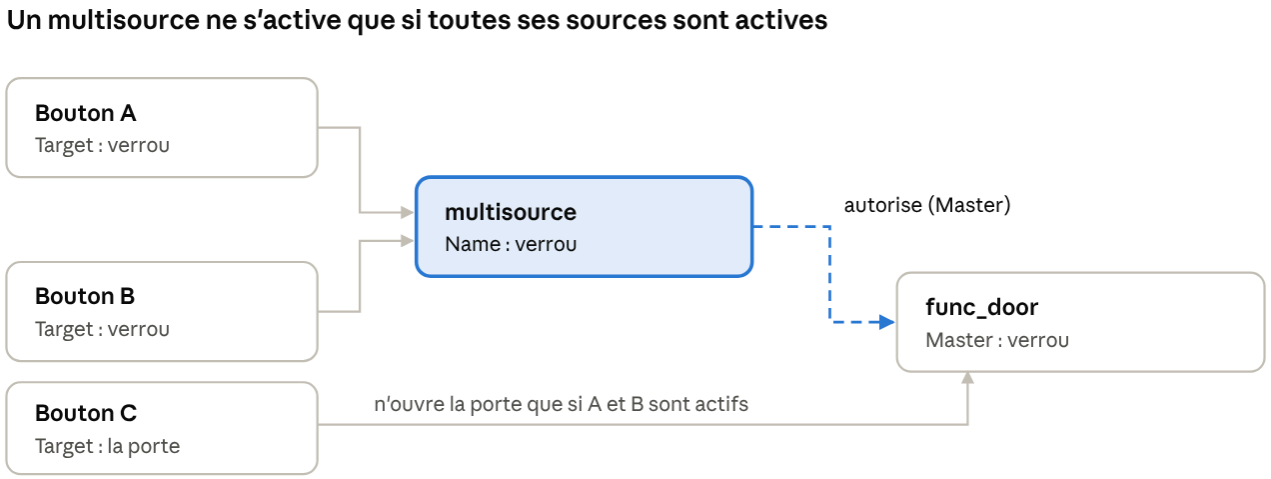
multi_manager · 1 appel, 3 cibles décalées

Le `multi_manager` appelle jusqu'à **16 cibles**, chacune après son propre délai (TWHL : multi_manager).

1. Placez un `multi_manager` et donnez-lui un **Name** (`mm_alarme`).
2. Faites pointer votre déclencheur dessus (**Target** = `mm_alarme`).
3. Dans ses propriétés, **désactivez SmartEdit**.
4. Cliquez sur **Add** : la **clé** est le nom de la cible, la **valeur** est le délai en secondes (`son_sirene` = `0.5`).
5. Répétez pour chaque cible.

> **Info :** pour appeler deux fois la même cible, ajoutez `#1`, `#2`… à la fin de la clé : `lampe`, `lampe#1`, `lampe#2`. Le jeu ignore ce suffixe, qui sert seulement à garder des clés différentes (TWHL). Exemple : `lampe` = 0, `lampe#1` = 1, `lampe#2` = 2 fait clignoter une lampe.

Verrouiller une entité : `multisource`



multisource · deux conditions pour ouvrir

Le `multisource` est un « ET » logique : il n'est actif que lorsque **tous** les déclencheurs qui le visent sont actifs (TWHL : Multisource).

On s'en sert surtout comme **Master** : une entité dont le champ Master contient le nom du multisource ne fonctionne que tant que celui-ci est actif.

> **Attention :** le multisource ne déclenche pas la porte lui-même, il l'autorise. Et si un bouton se réinitialise, le multisource redevient inactif : la porte se verrouille à nouveau. Réglez des délais de réinitialisation assez longs pour que le joueur ait le temps d'agir.

Les autres entités logiques

Entité	Rôle	Attributs clés
`trigger_relay`	Relais : reçoit un appel et le transmet, avec un délai et un état forcé (On, Off ou Toggle)	Trigger state ; flag **Remove on fire** pour qu'il ne serve qu'une fois

Entité	Rôle	Attributs clés
`trigger_changetarget`	Change le **Target** d'une autre entité en cours de partie	Target = l'entité à modifier, New target = sa nouvelle cible (TWHL)
`trigger_counter`	Ne déclenche sa cible qu'après avoir été activé un certain nombre de fois	Count before activation

Exemple avec `trigger_counter` : trois cibles à détruire (`func_breakable`), chacune avec Target = `compteur`. Le `trigger_counter` nommé `compteur`, avec un compte de 3, ouvre la porte quand la troisième est cassée.

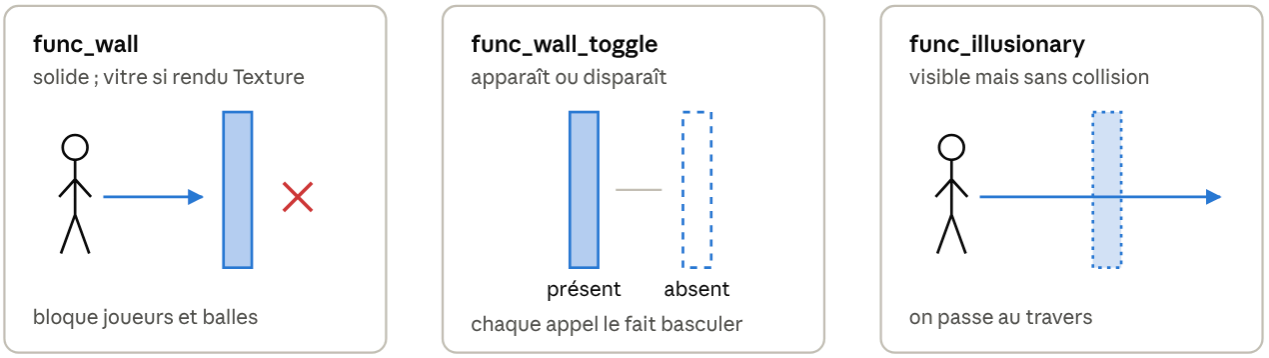
Sources : TWHL : multi_manager, TWHL : Multisource, TWHL : trigger_changetarget.

6. Les murs

Entités concernées : `func_wall`, `func_wall_toggle`, `func_illusionary` · **Type :** entité bloc · **Difficulté :** facile

Un bloc du monde est toujours opaque, solide et immobile. Transformé en entité, un mur peut devenir transparent, apparaître et disparaître, ou se laisser traverser.

Trois murs-entités : solide, à éclipse, traversable



murs-entités · func_wall, func_wall_toggle, func_illusionary

La transparence

Toutes ces entités ont les propriétés de rendu (**Render Mode**, **FX Amount**…). Les deux réglages à connaître (TWHL : func_wall, TWHL : Render Properties) :

- **Vitre** : Render Mode = **Texture**, FX Amount autour de **100**. Plus la valeur est basse, plus c'est transparent (0 = invisible, 255 = opaque).

- **Grille ou clôture** (texture `{`) : Render Mode = **Solid**, FX Amount = **255**. Les parties bleues de la texture disparaissent.

Les murs simples : `func_wall`

Un bloc fixe et solide, mais qui est une entité. On s'en sert pour :

- les vitres et les grilles (réglages ci-dessus) ;
- les **petits détails** (tuyaux, luminaires, rebords) : en entité, ils ne découpent pas les blocs du monde autour d'eux, ce qui évite des faces et des transitions de lumière inutiles (TWHL : func_wall) ;
- les murs dont la texture `+0` / `+A` doit basculer quand on les déclenche.

> **Astuce :** une face texturée CLIP sur un `func_wall` laisse passer les projectiles à travers ce côté (TWHL).

Les murs toggle : `func_wall_toggle`

Un mur qui disparaît ou réapparaît à chaque fois qu'il est appelé. Donnez-lui un **Name**, et faites-le viser par un bouton ou un déclencheur. Le flag **Starts Invisible** le fait commencer absent.

Exemple : un passage secret. Un `func_wall_toggle` nommé `passage` bouche un couloir ; un bouton caché avec Target = `passage` le fait disparaître.

Les murs illusoires : `func_illusionary`

Visible, mais **sans aucune collision** : ni les joueurs ni les balles ne le rencontrent. On l'utilise pour les feuillages, les rideaux, les toiles d'araignée, ou un faux mur qui cache un passage.

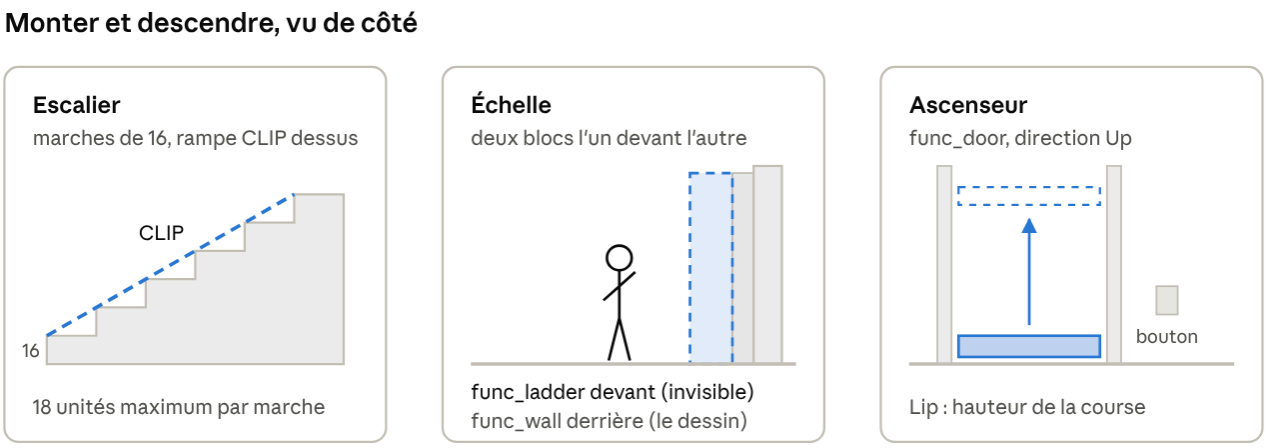
> **Attention :** comme toute entité, un `func_illusionary` ne ferme pas la map. Derrière un faux mur, il faut toujours un vrai bloc du monde pour arrêter le vide.

Sources : TWHL : func_wall, TWHL : Render Properties.

7. Monter et descendre

Entités concernées : `func_ladder`, `func_door`, `func_plat` · **Type :** entité bloc · **Difficulté :** facile à moyen

Trois façons de changer d'étage : l'escalier, l'échelle, l'ascenseur.



déplacements verticaux · escalier, échelle, ascenseur

Les escaliers

Un escalier n'est fait que de blocs du monde, mais il doit respecter le moteur : un joueur monte seul une marche de **18 unités au plus** (VDC : GoldSrc Dimensions). Au-delà, il faut sauter.

- Marches de **16 unités de haut** sur 16 à 32 de profondeur : la valeur la plus courante.
- Marches de 8 : plus réaliste, mais la vue du joueur saute à chaque marche.

> **Astuce :** posez par-dessus l'escalier un bloc **CLIP** en forme de rampe, du nez de la première marche à celui de la dernière. Le joueur glisse en douceur, les balles touchent toujours les vraies marches.

Les échelles : `func_ladder`

Le `func_ladder` est un volume invisible dans lequel le joueur grimpe. Le dessin de l'échelle est un autre bloc (TWHL : Ladders).

1. Créez le bloc visible de l'échelle, avec une texture d'échelle (32 unités de large pour une texture de 32).
2. Si la texture est masquée (`{ladder…`), faites-en un `func_wall` avec Render Mode = **Solid** et FX Amount = **255**.
3. Copiez ce bloc et placez la copie juste devant.
4. Faites de la copie un `func_ladder` , texturé en **AAATRIGGER** pour la repérer.

> **Attention :** n'utilisez pas de textures `CLIPHULL` autour d'une échelle : elle se comporte bizarrement et le joueur grimpe trop vite (TWHL : func_ladder).

> **Info :** un `func_ladder` peut être incliné (grillage en pente) ou horizontal (le joueur se déplace alors de côté) (TWHL).

Les ascenseurs

Avec un `func_door` (le plus souple) :

1. Créez la plateforme et faites-en un `func_door`.
2. **Angle** (ou Move direction) = **Up** : la « porte » monte au lieu de s'ouvrir sur le côté.
3. **Lip** : la distance à retrancher de la course. Par défaut, une porte se déplace de sa propre hauteur ; pour une plateforme de 8 qui doit monter de 128, mettez Lip = -120.
4. Donnez-lui un **Name**, cochez le flag **Toggle**, et placez un bouton en haut et un en bas, avec Target = ce nom.

Avec un `func_plat` : une plateforme automatique qui monte quand on marche dessus et redescend après quelques secondes. Le réglage **Travel altitude** fixe la hauteur. Le `func_platrot` fait la même chose en tournant.

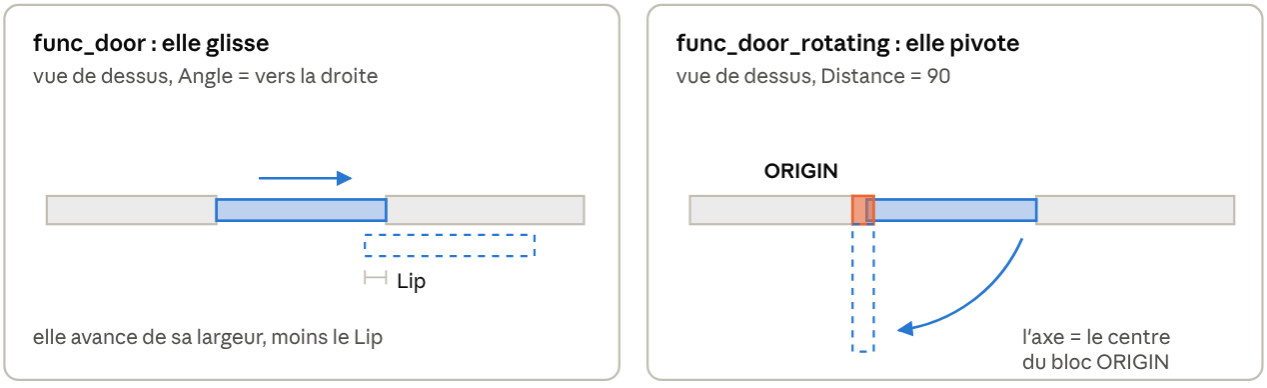
> **Attention :** si un joueur bloque un ascenseur, celui-ci repart en sens inverse, sauf si vous réglez des dégâts (**Damage inflicted when blocked**). Pour éviter les accidents, laissez des dégâts à 0 et prévoyez assez d'espace au-dessus de la plateforme.

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

8. Les portes

Éléments indispensables à presque toutes les maps, les portes existent en deux familles : celles qui glissent et celles qui pivotent.

Une porte glisse, ou pivote autour de son bloc ORIGIN



portes · translative et rotative, vues de dessus

Les portes translatives

Entité concernée : `func_door` · **Type :** entité bloc · **Difficulté :** assez facile

Une porte qui s'ouvre en glissant sur le côté, vers le haut ou vers le bas.

1. Créez un bloc avec une texture de porte, de la même taille que la texture.
2. Sélectionnez-le et cliquez sur **To Entity** (Ctrl+T).
3. Choisissez la classe `func_door`.

Les attributs utiles (TWHL : Doors) :

Attribut	Rôle
Angle	La direction d'ouverture : le petit cadran, ou **Up** / **Down** dans la liste
Name	Pour l'ouvrir avec un bouton ou un déclencheur
Speed	La vitesse d'ouverture
Master	Un `multisource` qui doit être actif pour que la porte s'ouvre
Move sound / Stop sound	Le bruit pendant et à la fin du mouvement
Delay before close	Secondes avant la fermeture automatique. **-1** = reste ouverte
Lip	Ce qui reste visible une fois ouverte. 0 = la porte disparaît entièrement dans le mur
Damage inflicted when blocked	Dégâts infligés à qui la bloque
Health (shoot open)	S'ouvre quand on lui tire dessus

Flag	Rôle
Starts Open	Commence ouverte
Passable	On la traverse sans être bloqué
Toggle	Un appel ouvre, l'appel suivant ferme
Use Only	S'ouvre seulement avec la touche Utiliser, pas en s'en approchant

> **Info :** une porte sans Name s'ouvre quand un joueur la touche. Avec un Name, elle attend d'être appelée (ou utilisée, si le flag Use Only est coché).

Les portes rotatives

Entité concernée : `func_door_rotating` · **Type :** entité bloc · **Difficulté :** moyen

1. Créez le bloc de la porte et texturez-le.
2. Créez un petit bloc texturé **ORIGIN** au niveau des gonds : son centre sera l'axe de rotation.
3. Sélectionnez **les deux** (Ctrl+clic), puis Ctrl+T et `func_door_rotating`.
4. **Distance** = 90 (l'angle d'ouverture en degrés).

Flags propres à cette entité (TWHL : func_door_rotating) : **Reverse Direction** (tourne dans l'autre sens), **One-way** (ne s'ouvre que dans un sens), **X axis / Y axis** (par défaut, la porte tourne autour de l'axe vertical Z).

> **Erreur fréquente :** oublier de lier le bloc ORIGIN à la porte. Elle tourne alors autour du centre de la map, et disparaît dans le décor (TWHL : Doors).

> **Particularité Counter-Strike :** une porte rotative en mouvement à la fin d'un round peut continuer à tourner au round suivant, jusqu'à la fin de son cycle (TWHL).

Les portes momentanées

Entités concernées : `momentary_door`, `momentary_rot_button` · **Difficulté :** moyen

Une porte qui s'ouvre tant que le joueur tourne une vanne, et qui s'arrête quand il la lâche. La vanne (`momentary_rot_button`, avec un bloc ORIGIN) a pour Target le nom du `momentary_door`.

Le bouton et la porte

Pour ouvrir une porte avec un bouton : la porte a un **Name** (`porte_1`), le bouton a **Target** = `porte_1` et le flag **Don't move**. Pour une double porte, créez deux `func_door` qui s'ouvrent dans des directions opposées et donnez-leur le **même Name** : un seul bouton les ouvre ensemble.

Le truc des portes vitrées

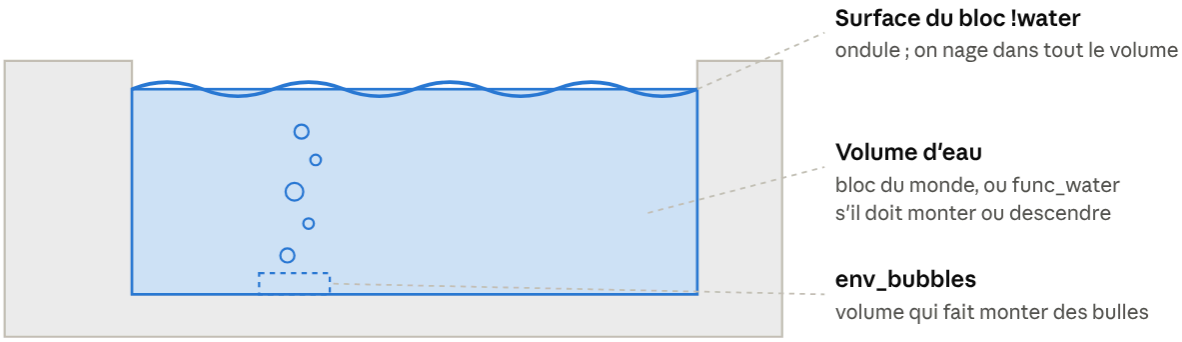
Une porte vitrée n'a rien de spécial : c'est une porte dont on règle le rendu. Mettez une texture de verre, Render Mode = **Texture**, FX Amount autour de **100**. Pour une porte avec un cadre opaque et une vitre, faites deux `func_door` superposées avec le même Name : l'une pour le cadre, l'autre pour la vitre transparente.

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

9. L'eau et les bulles

Entités concernées : `func_water`, `env_bubbles` · **Type :** entité bloc · **Difficulté :** facile

Un bassin vu en coupe : l'eau est un bloc, posé dans un creux fermé



En gris : les blocs du monde qui ferment le bassin. L'eau ne bouche jamais le vide.

bassin · coupe avec eau et bulles

L'eau

Dans GoldSrc, l'eau n'est pas une entité obligatoire : c'est un **bloc texturé avec une texture qui commence par `!`** (`!water`, `!waterblue`…). Ce préfixe suffit à rendre le volume liquide (VDC : Texture prefixes).

1. Creusez le bassin dans les blocs du monde.
2. Créez un bloc qui remplit le bassin jusqu'à la hauteur de l'eau.
3. Appliquez une texture `!` sur toutes ses faces.

Façon de faire	Ce qu'on obtient
Bloc du monde avec `!water`	Eau où l'on nage, surface qui ondule

Façon de faire	Ce qu'on obtient
`func_water`	La même eau, mais c'est une entité : seule la face du dessus est dessinée, et l'eau peut être déplacée comme une porte (niveau qui monte ou baisse quand on l'appelle)
`!slime`, `!lava`	Liquide qui blesse (la lave inflige le double de dégâts), seulement en bloc du monde
`!cur_0`, `!cur_90`…	Courant qui pousse le joueur dans une direction, seulement en bloc du monde

> **Attention :** l'eau ne ferme pas la map. Le bassin doit être entièrement entouré de blocs du monde, y compris sur les côtés qui dépassent de la surface.

> **Particularité Counter-Strike :** une bombe posée dans l'eau n'explose pas et ne termine pas le round. Ne placez jamais un site de bombe dans ou près de l'eau (TWHL : Introduction to CS mapping).

Les bulles : `env_bubbles`

Une entité bloc qui fait monter des bulles dans tout son volume. Réglez la **densité** (nombre de bulles), la **fréquence** et la **vitesse** de montée. Avec un Name, on peut l'allumer et l'éteindre.

Exemple : un petit bloc `env_bubbles` au fond du bassin, sous une grille d'aération, pour simuler une arrivée d'air.

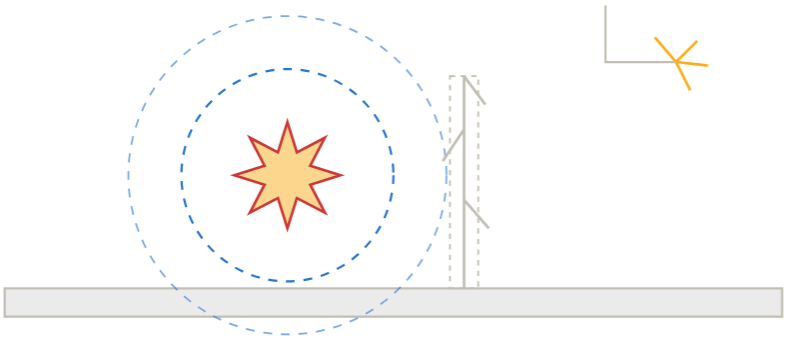
Sources : VDC : Texture prefixes, TWHL : Introduction to Counter-Strike mapping.

10. Effets spectaculaires

Entités concernées : `env_spark`, `env_shake`, `env_explosion`, `func_mortar_field` · **Type :** entités point (sauf `func_mortar_field`) · **Difficulté :** facile

Une explosion réussie n'est jamais une seule entité : c'est une boule de feu, un tremblement d'écran, des débris et des étincelles appelés en même temps.

Une « grosse » explosion = plusieurs entités appelées ensemble



env_spark
un câble arraché qui crépite

env_explosion
boule de feu, bruit, dégâts (Magnitude)

env_shake
l'écran tremble dans son rayon

func_breakable
le mur fissuré vole en éclats

Un multi_manager appelle les quatre au même moment (chapitre 5).

effets · explosion, tremblement, éclats, étincelles

Les étincelles : `env_spark`

Des étincelles et leur crépitement, à intervalle irrégulier. **Max Delay** fixe l'attente maximale entre deux gerbes. Avec un Name et les flags **Toggle** et **Start ON**, on peut l'allumer et l'éteindre. Placez-la à un endroit qui justifie l'effet : câble arraché, boîtier électrique.

Les tremblements : `env_shake`

Fait trembler l'écran des joueurs proches.

Attribut	Rôle
Amplitude	Force du tremblement, de 0 à 16
Frequency	Nombre de secousses par seconde ; autour de 40 donne un bon résultat
Duration	Durée en secondes
Effect radius	Rayon d'action
Flag GlobalShake	Toute la map tremble, quel que soit le rayon

Valeurs d'après la page VDC de l'entité, écrite pour Source : elles servent de point de départ, à vérifier en jeu.

Les explosions : `env_explosion`

Une explosion à l'endroit de l'entité, quand on l'appelle (VDC : env_explosion (GoldSrc)).

- **Magnitude** : la puissance, donc les dégâts.

- Flags : **No Damage** (1), **Repeatable** (2), **No Fireball** (4), **No Smoke** (8), **No Decal** (16), **No Sparks** (32).

> **Attention :** sans le flag **Repeatable**, l'explosion ne se produit qu'une fois et l'entité disparaît. Ses dégâts touchent aussi les joueurs : testez la Magnitude en jeu avant de publier la map. Cochez No Damage pour un effet purement visuel.

Les bombardements : `func_mortar_field`

Une entité bloc qui fait tomber des obus au hasard dans sa surface quand on l'appelle. Le nombre d'obus et leur dispersion se règlent dans ses propriétés.

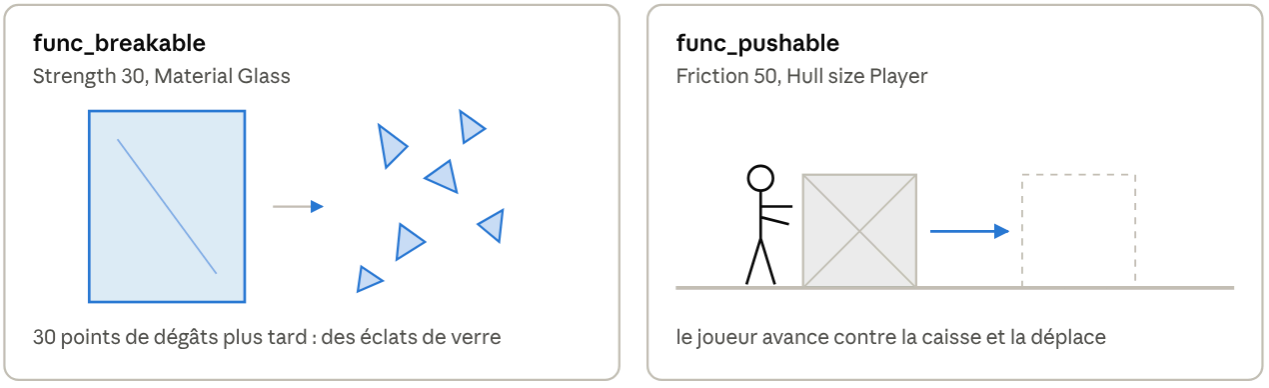
Exemple : un mur qui explose

1. Construisez une partie de mur fissurée et faites-en un `func_breakable` nommé `mur_casse` (chapitre suivant), flag **Only Trigger**.
2. Placez devant un `env_explosion` nommé `boum`, Magnitude 100, flag No Damage.
3. Ajoutez un `env_shake` nommé `secousse`, Amplitude 8, Duration 1, rayon 512.
4. Un `multi_manager` avec les clés `boum` = 0, `secousse` = 0, `mur_casse` = 0.1 les déclenche ensemble.
5. Un bouton ou un `trigger_once` a pour Target le `multi_manager`.

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

11. Casser et pousser

Une vitre se brise après assez de dégâts, une caisse se pousse



objets cassables et poussables

Les objets cassables

Entité concernée : `func_breakable` · **Type :** entité bloc · **Difficulté :** facile

Vitres, caisses, grilles d'aération, murs fragiles : tout ce qui se casse est un `func_breakable` (VDC : func_breakable (GoldSrc)).

Attribut	Rôle
Strength	Les dégâts à encaisser avant de casser. **0** = incassable
Material Type	Le bruit et l'aspect des débris
Gibs Direction	Débris au hasard (0) ou projetés dans le sens du tir (1)
Spawn on Break	Un objet qui apparaît à la place
Explode Magnitude	Une explosion au moment de casser. 0 = aucune
Gib Model	Un modèle personnalisé pour les débris

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

Flag	Rôle
Only Trigger (1)	Ne casse que si on l'appelle (Name), pas sous les balles
Touch (2)	Casse quand on le touche
Pressure (4)	Casse quand on monte dessus (plancher qui cède)
Instant Crowbar (256)	Casse au premier coup de corps à corps

> **Particularité Counter-Strike :** dans CS 1.6, une grenade qui touche un `func_breakable` peut le casser, quel que soit son matériau. D'après des mappeurs, l'objet doit avoir un Render Mode autre que Normal pour que certains comportements fonctionnent ; Solid avec FX Amount 255 garde un aspect normal (commentaires TWHL : func_breakable).

> **Info :** le `func_breakable` envoie un appel à son **Target** au moment où il casse. C'est ainsi qu'on fait le `trigger_counter` du chapitre 5.

Les objets poussables

Entité concernée : `func_pushable` · **Type :** entité bloc · **Difficulté :** moyen

Une caisse que le joueur déplace en avançant contre elle (TWHL : func_pushable).

Attribut	Rôle
Hull size	Le gabarit de collision : Point (16 × 16 × 16), Player (32 × 32 × 72), Big (64 × 64 × 72), Player duck (32 × 32 × 36)
Friction	La résistance, de 0 à 400 (400 = la plus lourde)
Buoyancy	La flottaison dans l'eau
Strength	Les dégâts avant de casser, si le flag **Breakable** est coché

> **Attention :** la collision d'un `func_pushable` est une boîte alignée sur les axes, pas la forme exacte de ses blocs. Une caisse plus petite que 32 × 32 se comporte comme une caisse de 32, une plus grande comme une caisse de 64. Faites des caisses carrées de 32 ou 64 pour éviter les surprises (TWHL).

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

12. Effets sonores

Entités concernées : `ambient_generic`, `env_sound` · **Type :** entité point · **Difficulté :** facile

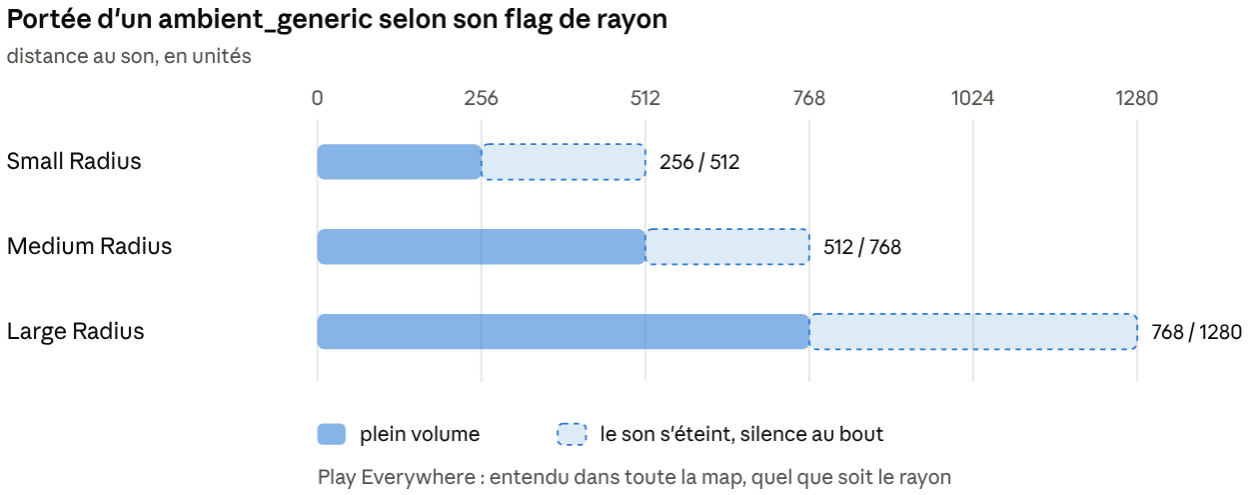
Un bourdonnement de machine, le vent, une alarme : le son fait beaucoup pour l'ambiance d'une map.

Jouer un son : `ambient_generic`

Attribut	Rôle
WAV Name	Le chemin du son depuis le dossier `sound` (`ambience/drips.wav`)
Volume	De 0 (muet) à 10 (maximum)
Pitch	La hauteur du son. 100 = normal

Attribut	Rôle
Fade in / Fade out	Montée et baisse progressives du volume
Dynamic Presets	Effets tout faits (machine, oscillation…) qui remplacent les autres réglages

Valeurs et flags d'après [TWHL : ambient_generic].



ambient_generic · portée des 3 rayons (TWHL)

Flag	Effet
Play Everywhere (1)	Entendu dans toute la map
Small / Medium / Large Radius (2, 4, 8)	La portée du son (schéma ci-dessus). Si plusieurs sont cochés, le plus petit l'emporte
Start Silent (16)	Ne joue pas au démarrage : il faut l'appeler
Not Toggled (32)	Rejoue à chaque appel au lieu de s'allumer et s'éteindre ; pour les sons courts

> **Info :** un son ne boucle pas parce qu'on le demande à l'entité : il boucle seulement si le fichier .wav contient des **cue points** (repères de boucle). Sans eux, il est joué une fois ([TWHL]).

Exemple : une alarme déclenchée par un bouton. Un `ambient_generic` nommé `alarme`, son `ambience/alarm1.wav`, flags **Start Silent** et **Medium Radius**. Le bouton a pour Target `alarme` : un appui lance l'alarme, un second l'arrête.

Modifier l'environnement sonore : `env_sound`

Quand le joueur entre dans son rayon (et le voit), l'acoustique change : écho, réverbération (VDC : env_sound (GoldSrc)).

Room Type	Effet
0	Normal, aucun effet
1	Générique
2 à 4	Métal (petit, moyen, grand)
5 à 7	Tunnel
8 à 10	Chambre
11 à 13	Pièce claire (« Bright »)
14 à 16	Sous l'eau
17 à 19	Béton
20 à 22	Très grand espace
23 à 25	Caverne
26 à 28	Effets étranges

> **Conseil :** placez un `env_sound` à l'entrée de chaque type de lieu (un tunnel, un hangar) et un autre réglé sur 0 à la sortie, pour revenir à un son normal.

Utiliser ses propres sons

Rangez vos fichiers dans un sous-dossier de `cstrike/sound` (par exemple `sound/ma_map/`) et indiquez ce chemin dans WAV Name. Comme pour les textures, ces fichiers doivent être fournis avec la map : voir le chapitre 19, « Sprites et models », pour le package.

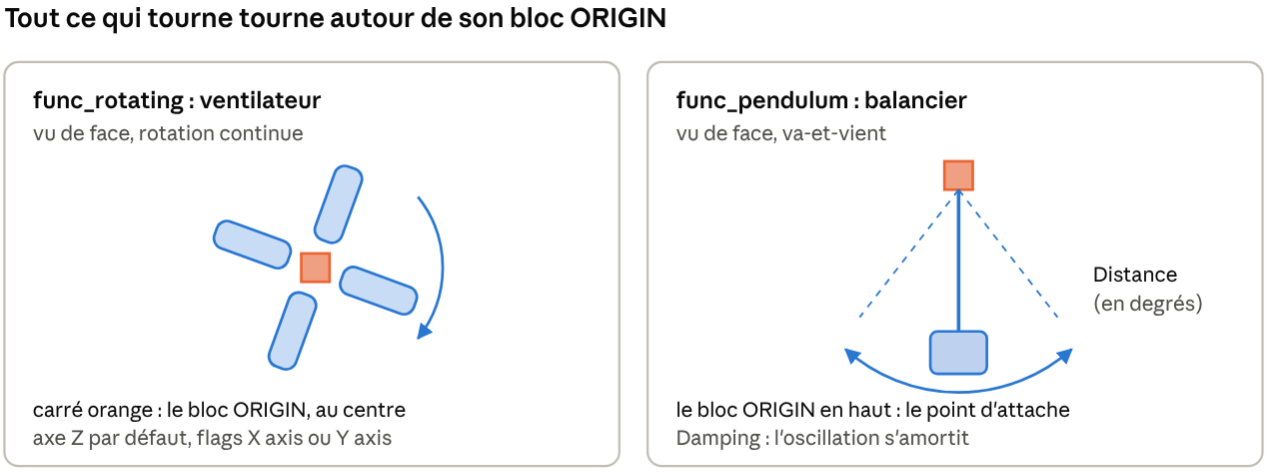
Les bruits de pas et d'impact liés aux textures sont traités au chapitre 6 de la Partie 1.

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

13. Rotations

Entités concernées : `func_rotating`, `func_pendulum` · **Type :** entité bloc · **Difficulté :** moyen

Ventilateurs, hélices, roues, pendules : comme les portes rotatives, ces entités ont besoin d'un bloc **ORIGIN** lié avec elles, placé sur l'axe de rotation.



rotations · ventilateur et balancier

Les objets rotatifs : `func_rotating`

Un objet qui tourne en continu (VDC : func_rotating (GoldSrc)).

1. Construisez l'objet (les pales d'un ventilateur).
2. Ajoutez un petit bloc **ORIGIN** à son centre.
3. Sélectionnez le tout, Ctrl+T, `func_rotating`.

Attribut	Rôle
Rotation Speed	La vitesse maximale
Friction	De 0 à 100 % : plus la valeur est basse, plus il met de temps à démarrer et à s'arrêter (avec le flag Acc/Dcc)
Damage when blocked	Dégâts infligés à qui le bloque
Fan Sounds / WAV Name	Le bruit du mouvement
Volume	De 0 à 10

Flag	Rôle
Start ON (1)	Tourne dès le début
Reverse Direction (2)	Tourne dans l'autre sens

Flag	Rôle
X Axis (4) / Y Axis (8)	Change l'axe ; par défaut il tourne autour de l'axe vertical
Acc/Dcc (16)	Accélère et ralentit progressivement
Fan Pain (32)	Blesse au contact
Not Solid (64)	On passe au travers

> **Exemple :** un ventilateur au plafond d'une salle. Pales en `func_rotating`, flags **Start ON** et **Not Solid**, Rotation Speed 200. Pour l'allumer avec un interrupteur, retirez Start ON, donnez-lui un Name, et faites-le viser par un `func_button`.

Les objets pendants : `func_pendulum`

Un objet qui se balance d'un côté à l'autre : lustre, enseigne, pendule (TWHL : func_pendulum).

Attribut	Rôle
Speed	La vitesse du balancement
Distance	L'amplitude, en degrés. Une valeur négative inverse le sens
Damping	De 0 à 1000 : le balancement s'amortit jusqu'à s'arrêter au milieu de sa course
Damage when blocked	Dégâts infligés à qui le bloque

Flags : **Start on** (1), **Passable** (8), **Auto-return** (16, revient à sa position de départ quand on l'appelle), **X-axis** (64) et **Y-axis** (128).

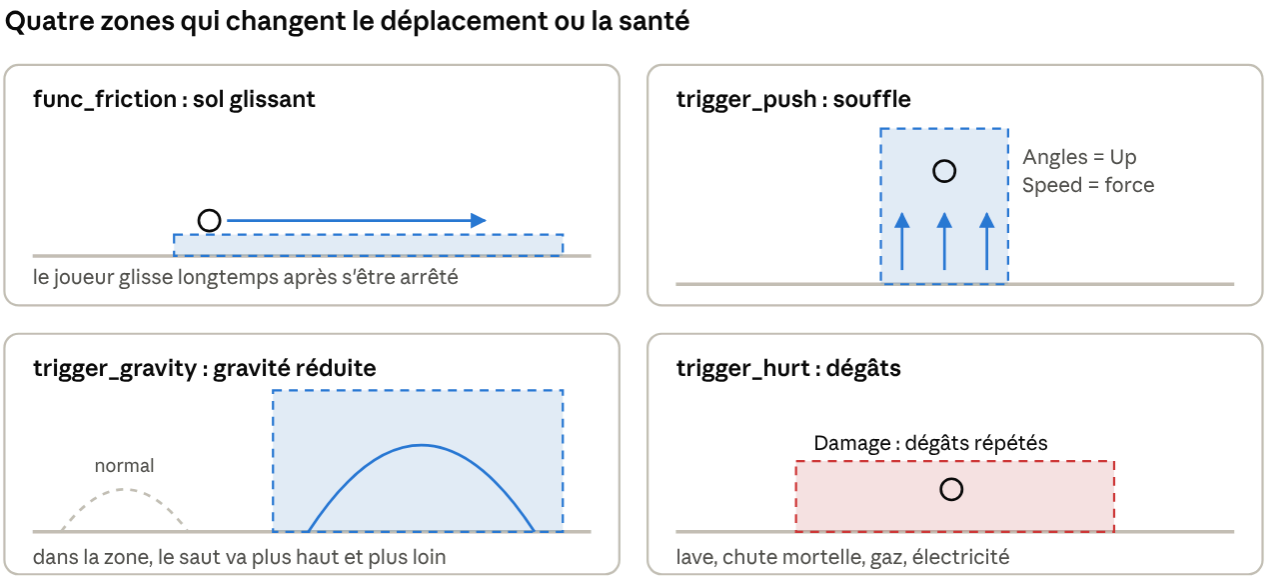
> **Astuce :** placé très loin de l'objet, le bloc ORIGIN transforme le balancement en un va-et-vient presque en ligne droite : une petite rotation autour d'un point lointain déplace beaucoup l'objet. TWHL donne la formule pour calculer l'angle (TWHL).

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

14. Environnement dangereux

Entités concernées : `func_friction`, `trigger_push`, `trigger_gravity`, `trigger_hurt` · **Type :** entité bloc · **Difficulté :** facile

Ces quatre entités sont des volumes qui changent ce qui arrive au joueur quand il y entre.



zones spéciales · friction, poussée, gravité, dégâts

Zone glissante : `func_friction`

Un bloc posé sur (ou formant) le sol, qui réduit l'adhérence : glace, huile. L'attribut **Percentage of standard** (`modifier`) règle la friction : 100 = normal, plus bas = plus glissant. Il est listé parmi les entités de Counter-Strike (VDC : List of CS entities).

Zone ventilée : `trigger_push`

Pousse tout ce qui le touche, dans la direction de ses **Angles** (Up pour un courant d'air vertical) et à la vitesse **Speed** (VDC : trigger_push (GoldSrc)).

Flag	Effet
Once Only (1)	Ne pousse qu'une fois
Start Off (2)	Commence éteint : il faut l'appeler pour l'activer
Pushables (4)	Pousse aussi les `func_pushable`

Exemple : un conduit d'aération vertical. Un `trigger_push` de la hauteur du conduit, Angles = Up, Speed 400, fait monter le joueur jusqu'en haut.

Modifier la gravité : `trigger_gravity`

Change la gravité du joueur qui le traverse. L'attribut **Gravity** est un multiplicateur : 1 = gravité normale, 0.5 = moitié moins forte.

> **Attention :** le changement de gravité reste appliqué au joueur après qu'il est sorti de la zone. Placez un second `trigger_gravity` réglé à 1 à la sortie pour lui rendre une gravité normale.

Blesser : `trigger_hurt`

Inflige des dégâts à qui se trouve dedans (VDC : trigger_hurt (GoldSrc)).

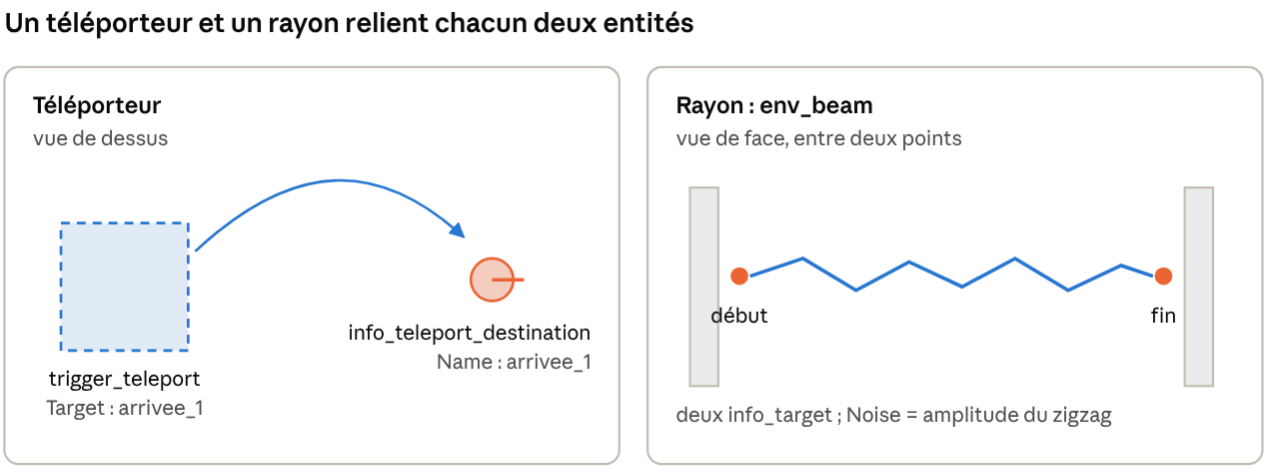
Attribut	Rôle
Damage	Les dégâts infligés, de façon répétée tant qu'on reste dedans
Damage Type	Le type : Burn (8), Shock (256), Poison (131072), Radiation (262144)… Il change l'icône et l'effet à l'écran
Name	Pour l'activer et le désactiver

Flags : **Target Once** (1), **Start Off** (2), **No Clients** (8, ne touche pas les joueurs), **FireClientOnly** (16), **TouchClientOnly** (32).

Exemple : un puits sans fond. Un `trigger_hurt` au fond, Damage 1000 : le joueur qui tombe meurt en arrivant, sans attendre les dégâts de chute.

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

15. Téléportation et effets visuels



téléporteur et rayon · deux entités reliées

La téléportation

Entités concernées : `trigger_teleport`, `info_teleport_destination` · **Type :** entité bloc + entité point · **Difficulté :** facile

Il faut deux entités : le volume de départ et le point d'arrivée (TWHL : Teleporters).

1. Créez un bloc là où le joueur doit disparaître, texturé AAATRIGGER, et faites-en un `trigger_teleport`.
2. Donnez-lui un **Target** : `arrivee_1`.
3. Placez un `info_teleport_destination` à l'arrivée et nommez-le `arrivee_1`. Son **angle** donne la direction dans laquelle le joueur regardera.

> **Attention :** placez la destination un peu au-dessus du sol et loin des murs et du plafond, sinon le joueur arrive coincé dans le décor.

> **Info :** si votre FGD ne propose pas `info_teleport_destination`, un `info_target` fonctionne de la même façon (TWHL).

Téléporteur activé par un bouton : donnez au `trigger_teleport` un **Master** = `tele_master`, créez un `multisource` nommé `tele_master`, et un `func_button` avec Target = `tele_master` et Delay before reset = 5. Le téléporteur fonctionne pendant 5 secondes après l'appui (TWHL).

Les sprites : `env_sprite`

Une image animée (fichier .spr) placée dans la map : flamme, lueur de lampe, fumée (TWHL : env_sprite).

Attribut	Rôle
Sprite Name	Le chemin du fichier (`sprites/glow01.spr`)
Framerate	Images par seconde de l'animation
Scale	La taille
Render Mode	**Glow** (grossit et s'estompe avec la distance), **Additive** (le noir devient transparent), **Texture**…
FX Amount / FX Color	Transparence et teinte

Flags : **Start on** (visible dès le début) et **Play Once** (l'animation se joue une fois à chaque appel).

Lueurs, rayons et lasers

Entité	Rôle	Attributs clés
`env_glow`	Un halo lumineux fixe	Sprite, couleur, intensité
`env_beam`	Un rayon entre deux entités (des `info_target`)	Start Entity, Ending Entity, largeur, Noise (zigzag), durée de vie, couleur
`env_laser`	Un laser qui part de l'entité vers une cible	Target (un `info_target`), sprite, largeur, dégâts

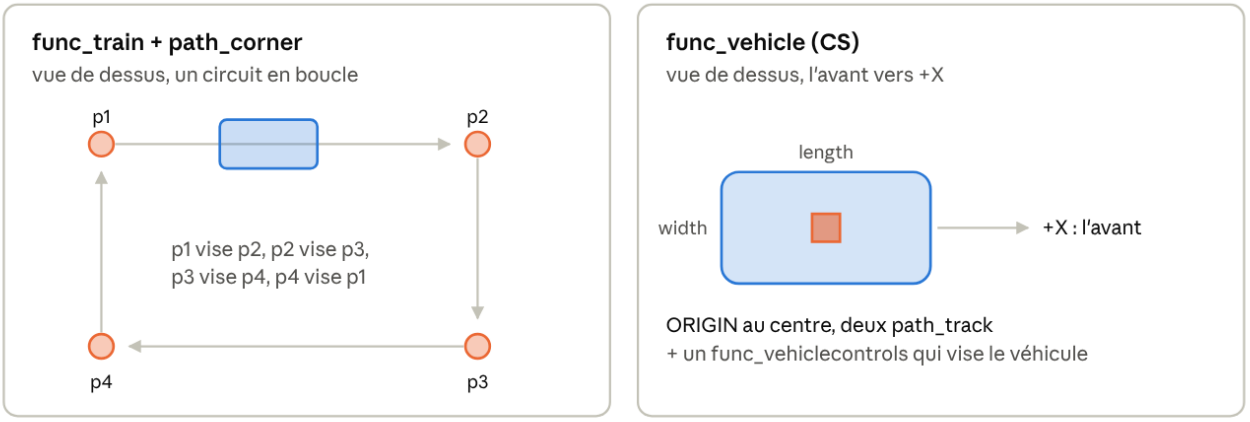
Exemple : un arc électrique entre deux bornes. Deux `info_target` nommés `borne_a` et `borne_b` , un `env_beam` avec Start = `borne_a` , End = `borne_b` , Noise 20, couleur bleu clair, flag **Start On**.

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

16. Moyens de transport

transports · circuit de train et véhicule CS

Un train suit une chaîne de points, chacun visant le suivant



transports · circuit de train et véhicule CS

Les trains simples : `func_train`

Entités concernées : `func_train`, `path_corner` · **Type :** entité bloc + entités point · **Difficulté :** moyen

Un objet qui se déplace en ligne droite de point en point : plateforme mobile, wagon, hélicoptère de décor (TWHL : func_train).

1. Construisez l'objet, ajoutez un bloc **ORIGIN** à son centre, et faites du tout un `func_train`, avec un Name.
2. Placez un premier `path_corner`, nommé `p1`.
3. Dans le train, **First stop target** = `p1` : au chargement, le train apparaît sur ce point, quelle que soit sa position dans l'éditeur.
4. Ajoutez d'autres `path_corner` ; chacun a pour **Next stop target** le nom du suivant. Pour une boucle, le dernier vise le premier.
5. Sur un `path_corner`, la case **Wait for retrigger** arrête le train jusqu'au prochain appel ; l'attribut **Wait** le fait patienter quelques secondes.

> **Attention :** un `func_train` ne tourne pas : il glisse d'un point à l'autre en gardant son orientation. Et son éclairage reste celui de sa position dans l'éditeur ; construisez-le dans un endroit éclairé comme son parcours (TWHL).

Les trains sur rails : `func_tracktrain`

Entités concernées : `func_tracktrain`, `path_track`, `func_traincontrols` · **Difficulté :** difficile

Contrairement au `func_train`, il **tourne** dans les virages et peut être piloté.

- Les `path_track` forment les rails : chacun a un **Next stop target** et peut avoir un **Branch path** (un aiguillage).
- Appelé sans Branch path, un `path_track` ouvre ou ferme le passage ; avec un Branch path, il bascule l'aiguillage (TWHL : path_track).
- Un bloc `func_traincontrols` (Target = le train) définit la zone où le joueur peut le piloter avec la touche Utiliser.
- `func_trackchange` et `func_trackautochange` sont des morceaux de rails mobiles (ascenseur ou plaque tournante pour trains).

Les voitures de CS : `func_vehicle`

Entités concernées : `func_vehicle`, `func_vehiclecontrols`, `path_track` · **Difficulté :** difficile

Counter-Strike ajoute un véhicule que les joueurs conduisent (the303 : func_vehicle).

1. Construisez le véhicule avec **l'avant tourné vers +X** dans l'éditeur, et un bloc ORIGIN au centre (en X et Y).
2. Ajoutez des blocs CLIP pour la collision, puis faites du tout un `func_vehicle` nommé `voiture`.
3. Réglez **length** (taille en X), **width** (taille en Y), **height** (du bas du bloc ORIGIN au sol), **speed**, **sounds**.
4. Placez deux `path_track` : le véhicule vise le premier (son point d'apparition), le premier vise le second, le second vise le premier.
5. Créez un bloc `func_vehiclecontrols` avec Target = `voiture` : le joueur conduit en appuyant sur Utiliser.

> **Attention :** le `func_vehicle` est capricieux : il peut se coincer ou se perdre. Ne rendez jamais la victoire dépendante d'un véhicule, et vérifiez qu'il ne peut pas bloquer un passage obligatoire (the303).

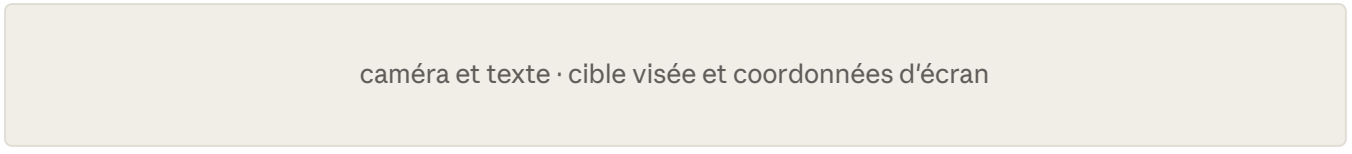
Les tapis roulants : `func_conveyor`

Un bloc dont la surface entraîne ce qui est posé dessus. Donnez à sa face du dessus une texture qui commence par `scroll` : elle défilera à la vitesse du tapis (VDC : Texture prefixes). La direction se règle avec l'angle, la vitesse avec **Conveyor Speed**.

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

17. Comme au cinéma

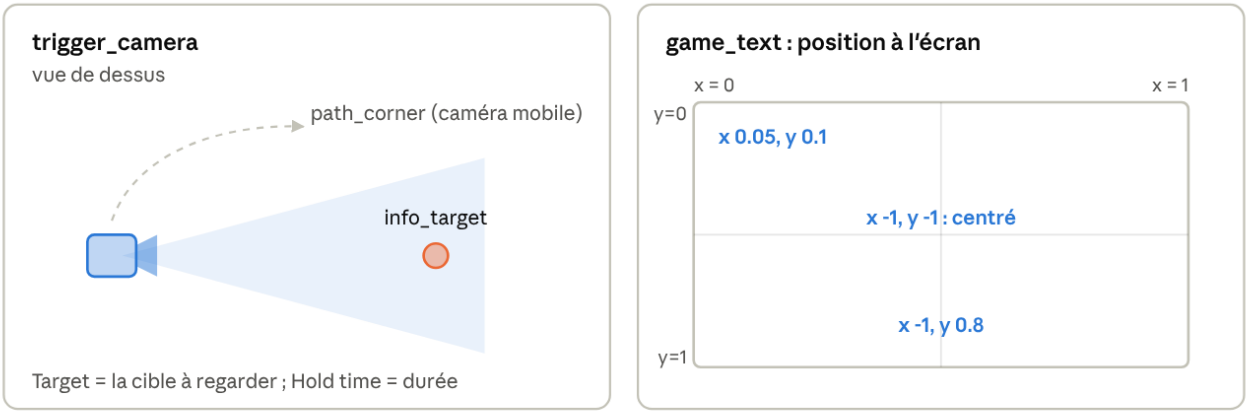
Entités concernées : `trigger_camera`, `env_fade`, `game_text` · **Type :** entité point · **Difficulté :** moyen



Les caméras : `trigger_camera`

Une fois appelée, la caméra prend la vue du joueur pendant un temps donné (TWHL : trigger_camera).

Une caméra vise une cible ; un texte se place en coordonnées d'écran



caméra et texte · cible visée et coordonnées d'écran

Attribut	Rôle
Target	L'entité que la caméra regarde (un `info_target`, ou un train pour suivre un objet en mouvement)
Hold time	La durée de la vue, en secondes
Path Corner	Le premier `path_corner`, pour une caméra qui se déplace
Initial Speed, Acceleration, Stop Deceleration	Le mouvement de la caméra mobile

Flags : **Start At Player** (1, part de la vue du joueur), **Follow Player** (2, regarde le joueur), **Freeze Player** (4, le joueur ne bouge plus pendant la séquence).

> **Particularité Counter-Strike :** les `trigger_camera` servent aussi aux vues d'introduction qu'on voit en arrivant sur un serveur, pendant le choix de l'équipe (TWHL). Placez-en quelques-unes, chacune visant un `info_target`, pour présenter votre map.

> **Attention :** le moteur ne dessine la vue de la caméra que si le joueur est assez proche d'elle. Une caméra à l'autre bout de la map peut afficher des zones vides (TWHL).

Les fondus : `env_fade`

Recouvre l'écran d'une couleur, puis l'enlève. Réglez la **durée** du fondu, le temps de **maintien** (Hold), la **couleur** et son opacité (FX Amount). Le flag **Fade From** fait le chemin inverse (de la couleur vers l'image), **Activator Only** limite l'effet au joueur qui a déclenché.

Exemple : un flash blanc d'une demi-seconde au moment d'une explosion, en ajoutant un `env_fade` blanc au `multi_manager` du chapitre 10.

Afficher du texte : `game_text`

Un message à l'écran, à la position et avec l'effet choisis (VDC : game_text (GoldSrc)).

Attribut	Rôle
Message	Le texte

Attribut	Rôle
X / Y	La position, de 0 à 1 (gauche à droite, haut en bas). **-1 centre le texte**
Text Effect	0 : fondu ; 1 : générique ; 2 : lettres qui apparaissent une à une
Color1 / Color2	La couleur du texte, et celle de la lettre en cours avec l'effet 2
Fade in / Hold / Fade out	Apparition, durée d'affichage, disparition
Channel	De 0 à 4 : un nouveau message sur le même canal remplace l'ancien

Le flag **All Players** affiche le message à tous ; sans lui, seul le joueur qui a déclenché le voit.

Exemple : un message de bienvenue. Un `game_text` avec Message = « Bienvenue ! Protégez le site A », X = -1, Y = 0.3, Hold 4, flag All Players, appelé par un `trigger_multiple` dans la zone de départ.

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

18. Armes et équipement

Entités concernées : `armoury_entity`, `game_player_equip`, `player_weaponstrip` · **Type :** entité point · **Difficulté :** facile à moyen

Dans Counter-Strike, on achète normalement ses armes. Mais une map peut aussi en poser au sol, en donner à tous les joueurs ou les retirer (TWHL : Stripping and Equipping Weapons in CS).

Les armes au sol : `armoury_entity`

Pose une arme ou un équipement à ramasser. Choisissez l'objet dans la liste **Item** et le nombre d'exemplaires dans **Count**.

- Les pistolets ne sont pas proposés.
- Les munitions ne sont pas pleines.
- Ne la placez pas dans le sol ni sur un point de départ.

Équiper tous les joueurs : `game_player_equip`

équipement · game_player_equip au spawn

1. Placez un `game_player_equip`, **sans Name**, flag **Use only décoché**.
2. Désactivez SmartEdit et ajoutez une clé par objet, avec la quantité en valeur.
3. À chaque apparition, tous les joueurs reçoivent exactement cette liste.

> **Attention :** dès qu'un `game_player_equip` est dans la map, les joueurs apparaissent **sans aucune arme**, même si l'entité a le flag Use only. Pensez à ajouter `weapon_knife` = 1, sinon ils n'auront même pas de couteau (TWHL).

Clé	Objet
`weapon_ak47`, `weapon_m4a1`, `weapon_awp`, `weapon_deagle`…	Les armes (même nom que dans le jeu)
`ammo_762nato`, `ammo_556nato`, `ammo_50ae`, `ammo_buckshot`…	Les chargeurs
`item_kevlar`	Gilet
`item_assaultsuit`	Gilet + casque
`item_thighpack`	Kit de désamorçage
`weapon_hegrenade`, `weapon_flashbang`, `weapon_smokegrenade`	Les grenades

> **Erreur fréquente :** ne donnez pas `weapon_c4` avec cette entité : elle ne crée pas une vraie bombe (TWHL).

Avec un Name et le flag Use only coché, l'entité ne donne ses objets qu'au joueur qui la déclenche : utile pour un casier d'armes à ouvrir avec un bouton.

Empêcher les achats

Pour une map où tout le monde a les mêmes armes, il faut aussi couper les achats. Deux méthodes, à combiner :

- un petit `func_buyzone` hors d'atteinte, sans équipe : il supprime les zones d'achat automatiques autour des départs ;

Un game_player_equip sans nom équipe chaque joueur à son apparition



équipement · game_player_equip au spawn

- un `info_map_parameters` réglé pour que personne ne puisse acheter (chapitre 20).

Confisquer les armes : `player_weaponstrip`

Retire toutes les armes du joueur qui le déclenche. Nommez-le (`confisque`), puis créez un `trigger_multiple` avec Target = `confisque` .

> **Attention :** il n'agit que sur le joueur qui déclenche : un `trigger_auto` ou une porte lointaine ne confisque rien. Et ne l'utilisez pas sur une map de_ : il retire aussi la bombe (TWHL).

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

19. Sprites et models

Entités concernées : `env_sprite` , `cycler_sprite` · **Type :** entité point · **Difficulté :** moyen

Les sprites

Un sprite (.spr) est une image plate, souvent animée, qui fait toujours face au joueur : flamme, halo, fumée, étincelle. On le place avec `env_sprite` , détaillé au chapitre 15. Le jeu en contient beaucoup dans `valve/sprites` et `cstrike/sprites` .

Les models

Un model (.mdl) est un objet 3D : une plante, un véhicule de décor, un meuble. Il est souvent plus beau et moins coûteux qu'un objet fait de blocs.

Dans Counter-Strike, l'entité la plus utilisée pour poser un model décoratif est `cycler_sprite` (TWHL : cycler_sprite) :

1. Placez un `cycler_sprite` .
2. Dans **Sprite** (clé `model`), indiquez le chemin du model : `models/ma_map/plante.mdl` .
3. Counter-Strike ajoute les clés **body**, **skin** et **sequence**, pour choisir une variante ou une animation.
4. Orientez-le avec les angles.

> **Attention :** un model ne bloque pas toujours les joueurs ni les balles. Entourez-le d'un bloc **CLIP** si le joueur ne doit pas passer au travers. Et un `cycler_sprite` qu'on frappe change d'image d'animation : pour un décor fixe, choisissez une séquence sans mouvement (TWHL).

> **Info :** J.A.C.K. affiche les models et les sprites directement dans la vue 3D, et sait les ouvrir dans sa visionneuse (VDC : J.A.C.K.).

Le package

package de map · dossiers et fichiers à fournir

Un .bsp ne peut contenir que des textures WAD. Tout le reste (sons, models, sprites, ciel personnalisé) reste à côté, et le serveur doit savoir quoi envoyer aux joueurs. C'est le rôle du fichier **.res** : une liste de chemins, un fichier par ligne, rangée à côté du .bsp (VDC : Resource list (GoldSrc)).

Exemple de `ma_map.res` :

```
sound/ma_map/alarme.wav
models/ma_map/plante.mdl
sprites/ma_map/flamme.spr
gfx/env/monciel_up.tga
gfx/env/monciel_dn.tga
```

(Les six images du ciel sont toutes à lister ; seules deux sont montrées ici.)

Pour le générer : **ResGuy**, outil récent fait pour Sven Co-op mais compatible avec tous les jeux GoldSrc, ou **RESGen**, plus ancien (2005), dont il faut relire la liste (VDC).

> **Conseil :** rangez tous vos fichiers personnels dans des sous-dossiers au nom de la map (`sound/ma_map/` , `models/ma_map/`). Vous évitez d'écraser les fichiers d'une autre map qui porteraient le même nom.

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

20. Les modes de jeu Counter-Strike

Entités concernées : `hostage_entity` , `func_hostage_rescue` , `func_bomb_target` , `info_vip_start` , `func_vip_safetyzone` , `func_escapezone` , `func_buyzone` , `info_map_parameters` · **Difficulté :** facile

C'est le préfixe du nom de la map qui annonce son mode de jeu. Le jeu, lui, déduit le mode des entités présentes (TWHL : Introduction to Counter-Strike mapping).

Le package d'une map : tout ce que le .bsp ne contient pas



package de map · dossiers et fichiers à fournir

modes de jeu · 4 préfixes officiels

Maps cs_ : le sauvetage d'otages

- Placez 4 ou 5 `hostage_entity`, en général près du départ des Terroristes.
- La zone de sauvetage : un `func_hostage_rescue` (bloc, taille libre) ou un `info_hostage_rescue` (point, rayon d'environ 128 unités). N'utilisez qu'un seul des deux : si les deux existent, le jeu ignore le second.
- Placez la zone loin des murs, sinon un otage peut être sauvé à travers un mur. Le decal `{hrpoint` la signale aux joueurs.

Maps de_ : la bombe

- Au moins un `func_bomb_target` : sa taille définit où l'on peut poser la bombe. La plupart des maps en ont deux, les sites A et B.
- Alternative : `info_bomb_target`, avec une zone de taille fixe.
- Le decal `{target` (ou `{target2`) marque le site.

> **Attention :** ne mettez jamais un site de bombe dans l'eau ou juste à côté. Une bombe posée dans l'eau n'explose pas et le round ne se termine pas (TWHL).

Maps as_ : le VIP

- Un `info_vip_start`, en plus des départs normaux, près des Anti-Terroristes.
- Un `func_vip_safetyzone` : la zone que le VIP doit atteindre.

Maps es_ : l'évasion

Un `func_escapezone`, près du départ des CT, que les Terroristes doivent atteindre. Ce mode est peu pris en charge par les versions récentes de CS (TWHL).

Les zones d'achat : `func_buyzone`

- **Sans** `func_buyzone`, chaque point de départ crée une petite zone d'achat autour de lui.
- **Avec** un `func_buyzone`, ces zones automatiques disparaissent : c'est le bloc qui décide. Réglez son équipe (**Team**) et prévoyez-en un par équipe.
- Pour supprimer les achats, cachez un petit `func_buyzone` hors d'atteinte.

Le préfixe du nom de la map annonce le mode, et ses entités

cs_
Otages

hostage_entity (4 à 5)
func_hostage_rescue
ou info_hostage_rescue

décal {hrpoint

de_
Bombe

func_bomb_target
(1 ou 2 sites)
jamais dans l'eau

décal {target

as_
VIP

info_vip_start
func_vip_safetyzone
départ VIP près des CT

es_
Évasion

func_escapezone
près du départ CT
mode peu suivi

Pour toutes : départs CT et T (chapitre 2), func_buyzone et info_map_parameters en option

modes de jeu · 4 préfixes officiels

Les règles de la map : `info_map_parameters`

Une entité point qui fixe qui peut acheter (tout le monde, CT seulement, T seulement, personne) et le rayon de l'explosion de la bombe (**bombradius**, 500 par défaut). Ces valeurs viennent de la page VDC de l'entité, commune à toute la série Counter-Strike : vérifiez-les dans votre FGD de CS 1.6.

Les préfixes de la communauté

En dehors des modes officiels, les joueurs reconnaissent des préfixes non officiels (TWHL) : `fy_` (fight yard, petite arène avec armes au sol), `aim_`, `awp_`, `ka_` (arène au couteau), `35hp_` et `1hp_` (santé réduite), `kz_` (escalade), `surf_` (glisse sur des rampes).

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

Partie 3 : Compilation et optimisation

Transformez votre travail en map jouable, puis rendez-la fluide.

1. La compilation (A : tester sa map)

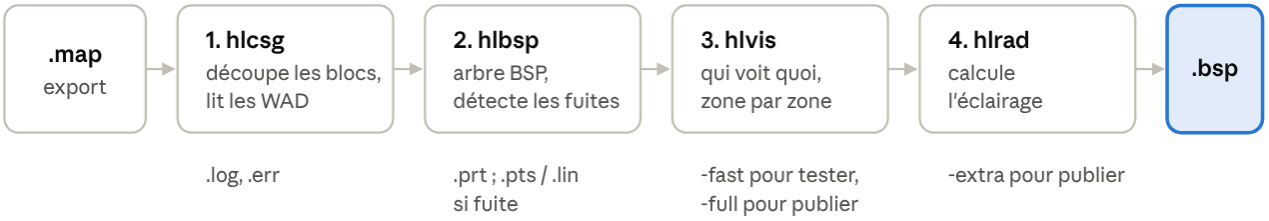
Votre map ne se joue pas telle quelle : le fichier de J.A.C.K. doit être transformé en .bsp, le format que lit le jeu. C'est la compilation.

Les formats de fichiers

Fichier	Rôle
.jmf (ou .rmf)	Votre fichier de travail dans l'éditeur
.map	La version texte, exportée pour les compilateurs (format Valve220 chez J.A.C.K.)
.bsp	La map compilée, jouable
.log / .err	Le compte rendu de la compilation, et les erreurs
.pts / .lin	Le tracé d'une fuite (pointfile)
.prt	Les portails, utilisés par VIS ; J.A.C.K. sait les afficher

Comment ça marche

Quatre programmes, dans l'ordre, du .map au .bsp jouable



Si une étape échoue, les suivantes travaillent sur un résultat faux : lisez le .log depuis le début.

chaîne de compilation · CSG, BSP, VIS, RAD

Quatre programmes s'enchaînent (TWHL : Compiling 101) :

1. **hlcsg** découpe les blocs qui se chevauchent, construit les volumes de collision et liste les textures utilisées.
2. **hlbsp** construit l'arbre BSP, le découpage de la map en volumes. C'est lui qui détecte les fuites.
3. **hlvis** calcule, pour chaque zone, quelles autres zones on peut voir. Sans lui, le jeu dessine toute la map en permanence.
4. **hlrad** calcule la lumière et ses rebonds. Sans lui, la map est entièrement éclairée, sans ombres.

Les compilateurs recommandés sont ceux de **VHLT** (TWHL : Compile).

Compiler dans J.A.C.K.

File > Run Map (F9).

- **Mode normal** : pour chaque étape, choisissez **Normal**, **Fast** ou **No**. Pratique au début.
- **Mode expert** : vous écrivez vous-même chaque commande et ses paramètres, et vous pouvez ajouter des étapes (copier le .bsp, lancer le jeu). C'est le mode à utiliser dès que vous connaissez les paramètres.

J.A.C.K. compile sans bloquer l'éditeur : vous pouvez continuer à travailler pendant la compilation (VDC : J.A.C.K.).

> **Info :** pour lancer le jeu directement sur votre map, ajoutez à la ligne de commande du jeu `-game cstrike +map ma_map`. Ajoutez `-dev` et `-console` pour voir les messages de débogage.

Les paramètres utiles

Programme	Paramètre	Effet
tous	`-chart`	Affiche les statistiques et les limites atteintes
tous	`-estimate`	Affiche le temps restant
tous	`-low`	Priorité basse : le PC reste utilisable
hlcsg	`-wadinclude nom`	Intègre au .bsp les textures du WAD `nom`
hlcsg	`-nowadtextures`	Intègre toutes les textures utilisées
hlcsg	`-cliptype precise`	Collisions plus précises dans les angles
hlbsp	`-leakonly`	S'arrête après la recherche de fuites

Programme	Paramètre	Effet
hlvis	`-fast`	VIS rapide, pour les tests
hlvis	`-full`	VIS précis, pour la version publiée
hlrad	`-extra`	Éclairage de meilleure qualité (plus long)
hlrad	`-bounce n`	Nombre de rebonds de la lumière

Sources des paramètres : VDC : HLBSP, TWHL : Tool Textures (pour `-cliptype precise`), VDC : compile settings de SirYodaJedi.

Exemple de réglage

Étape	Pendant la construction	Pour la version publiée
CSG	`-wadinclude ma_map`	`-wadinclude ma_map -cliptype precise`
BSP	(rien)	`-chart`
VIS	`-fast`	`-full`
RAD	(rien)	`-extra -chart`

Avec ces réglages, une petite map se compile en quelques secondes pendant les tests. La version finale est plus longue, mais plus belle et plus fluide.

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

2. La compilation (B : débogage)

Votre map ne se lance pas, elle est toute noire ou toute claire, ou le jeu rame ? La réponse est presque toujours dans le compte rendu de compilation.

Quand ça plante : lire le log

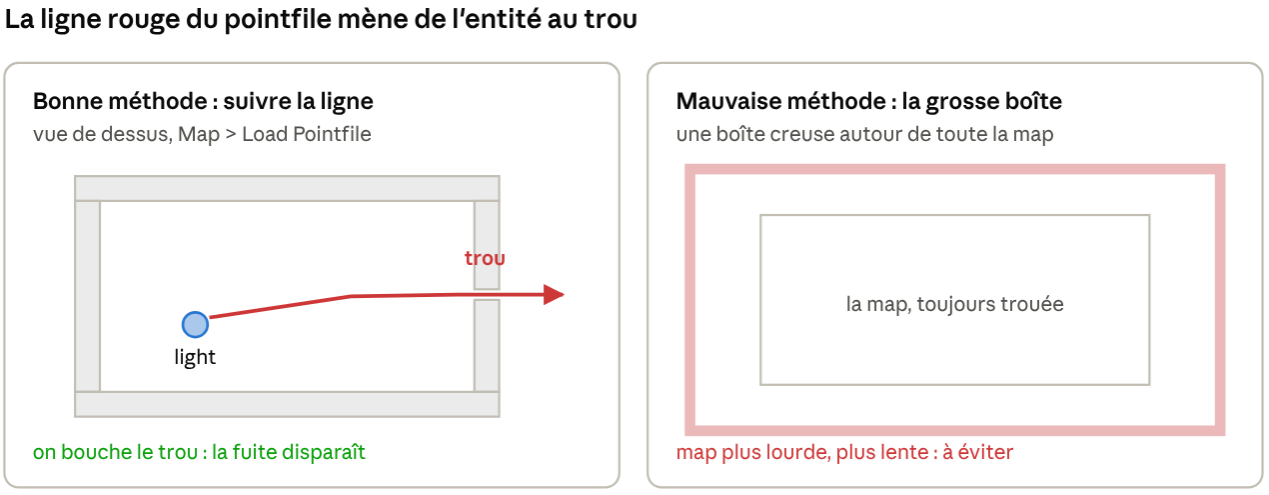
Chaque compilation écrit un fichier `ma_map.log` à côté de la map (et `ma_map.err` en cas d'erreur). Lisez-le **depuis le début** : la première erreur entraîne souvent toutes les suivantes (TWHL : setup TrenchBroom, section compilation).

Les blocs invalides

Un bloc est invalide s'il n'est pas convexe ou si une de ses faces n'est pas plane, le plus souvent après une manipulation de sommets. Le compilateur le « corrige » à sa façon, ce qui peut ouvrir un trou (TWHL : How to fix those leaks).

1. **Map > Check for Problems** (Alt+P) liste les blocs invalides et propose souvent de les réparer.
2. Si le log cite un numéro de bloc, **Go to Brush Number** (Ctrl+Shift+G) vous y emmène.
3. Pour une face tordue, la **triangulation** de J.A.C.K. la découpe en faces planes.

Les fuites (leaks)



fuite · pointfile contre grosse boîte

Une fuite signifie qu'une entité « voit » le vide par un trou de la map. HLBSP ne sait alors plus ce qui est dedans ou dehors : VIS et RAD ne fonctionnent pas, la map est entièrement éclairée et lente.

La bonne méthode (TWHL : Pointfiles) :

1. Après une compilation avec fuite, ouvrez **Map > Load Pointfile**. En GoldSrc, choisissez le fichier **.lin** plutôt que .pts.
2. Une ligne rouge apparaît dans les vues : elle part d'une entité et traverse le trou. Suivez-la.
3. Bouchez le trou avec un bloc du monde, et recompilez.

Les causes fréquentes (TWHL) :

Cause	Solution
Un petit trou entre deux blocs	Travailler sur la grille, recaler avec Snap to Grid (Ctrl+B)
Un mur extérieur transformé en entité (`func_wall`, porte)	Les entités ne ferment pas la map : remettre un bloc du monde derrière

Cause	Solution
Une entité hors de la map	Seul son point d'origine compte : la ramener à l'intérieur
Un bloc ORIGIN qui dépasse dans le vide	Déplacer le bloc ORIGIN à l'intérieur
Un bloc invalide	Le réparer (voir plus haut)

> **Erreur fréquente :** « réparer » une fuite en mettant toute la map dans une grande boîte creuse. La fuite est cachée, pas corrigée : la map devient plus lourde, plus longue à compiler et plus lente en jeu (TWHL).

> **Info :** le compilateur ne signale qu'une fuite à la fois. Recompilez après chaque correction. Pour aller vite, `-leakonly` sur hlbsp s'arrête dès la recherche de fuite terminée (VDC : HLBSP).

Le ciel noir ou absent

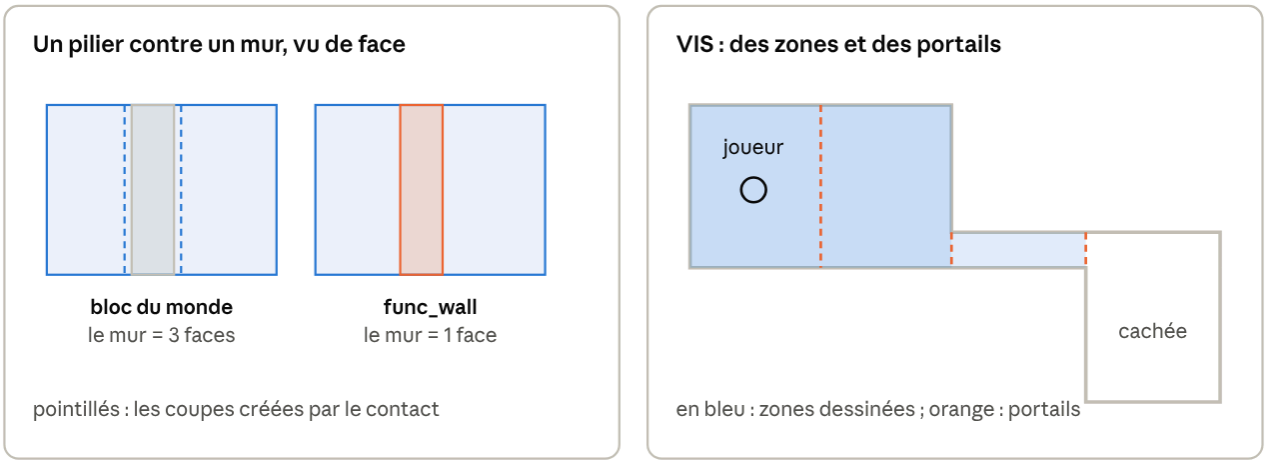
Si le ciel s'affiche mal, vérifiez que le nom du ciel dans **Map > Map Properties** (clé `skyname` du worldspawn) correspond bien à six images présentes dans `gfx/env/`. Un ciel personnalisé doit aussi figurer dans le fichier .res (Partie 2, chapitre 19).

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

3. Optimisation (A : les lois de VIS)

Une map fluide n'est pas une map vide : c'est une map où le moteur ne dessine que ce qui est utile. Pour l'aider, il faut comprendre comment il travaille.

Les blocs qui se touchent découpent les faces, VIS découpe la map en zones



optimisation · découpage des faces et zones de VIS

Le découpage des polygones

Le moteur ne dessine pas des blocs, mais des **faces** (polygones). Plus il y en a à l'écran, plus la carte graphique travaille. Deux choses multiplient les faces (TWHL : Total Map Optimisation) :

- **Les blocs du monde qui se touchent** se découpent mutuellement : un pilier collé contre un mur coupe le mur en plusieurs faces. En faisant du pilier un `func_wall` (ou un `func_detail` avec VHLT), le mur reste d'un seul tenant. Dans l'exemple de TWHL, ce seul réflexe a économisé près de 100 polygones.
- **Les grandes faces** sont redécoupées tous les **240 pixels de texture** environ, à l'échelle 1. Une texture à l'échelle 2 n'est coupée que tous les 480 pixels.

Le travail du moteur

1. **hlbsp** découpe l'espace vide de la map en volumes convexes, les **visleafs** (zones).
2. Entre deux zones voisines se trouve un **portail**, une ouverture par laquelle on peut voir.
3. **hlvis** calcule, pour chaque zone, la liste des zones visibles à travers les portails.
4. En jeu, le moteur ne dessine que les zones visibles depuis celle où se trouve le joueur, même si une partie est cachée par un angle.

C'est pourquoi un couloir droit qui traverse toute la map est coûteux : depuis un bout, on « voit » tout jusqu'à l'autre.

Mesurer : les r_speeds

Dans la console du jeu (TWHL) :

```
developer 1
sv_cheats 1
r_speeds 1
map ma_map
```

En haut à gauche s'affichent :

- **wpoly** : les polygones du monde (blocs) dessinés ;
- **epoly** : les polygones des entités et des modèles (joueurs, armes, models).

Deux commandes complètent l'analyse : `gl_wireframe 1` montre le découpage des faces, `gl_wireframe 2` montre tout ce que le moteur dessine, y compris derrière les murs.

> **Conseil :** placez-vous aux endroits les plus ouverts de la map, notez les wpoly, puis modifiez et recompilez. C'est la seule façon de savoir si une optimisation a servi.

Source : TWHL : Total Map Optimisation Part 1.

4. Optimisation (B : diminuer les r_speeds)

Vous savez mesurer. Voici comment faire baisser les chiffres, du plus efficace au plus fin.

1. Bien structurer sa map

Aucune astuce ne sauve un mauvais plan (TWHL : Hint Brushes). Pensez la visibilité dès le début :

- **En intérieur** : des couloirs coudés plutôt que droits, des pièces reliées par des passages étroits ou des angles. Chaque angle coupe la vue.
- **En extérieur** : de grands murs, des bâtiments ou du relief qui séparent les zones. Une grande place ouverte où l'on voit tout est le pire cas.

2. Éviter le contact direct des blocs

Tout ce qui est décoratif et touche un mur (piliers, tuyaux, rebords, lampes) passe en `func_wall`, ou en `func_detail` avec VHLT. Le mur n'est plus découpé, et VIS a moins de travail (TWHL : Total Map Optimisation).

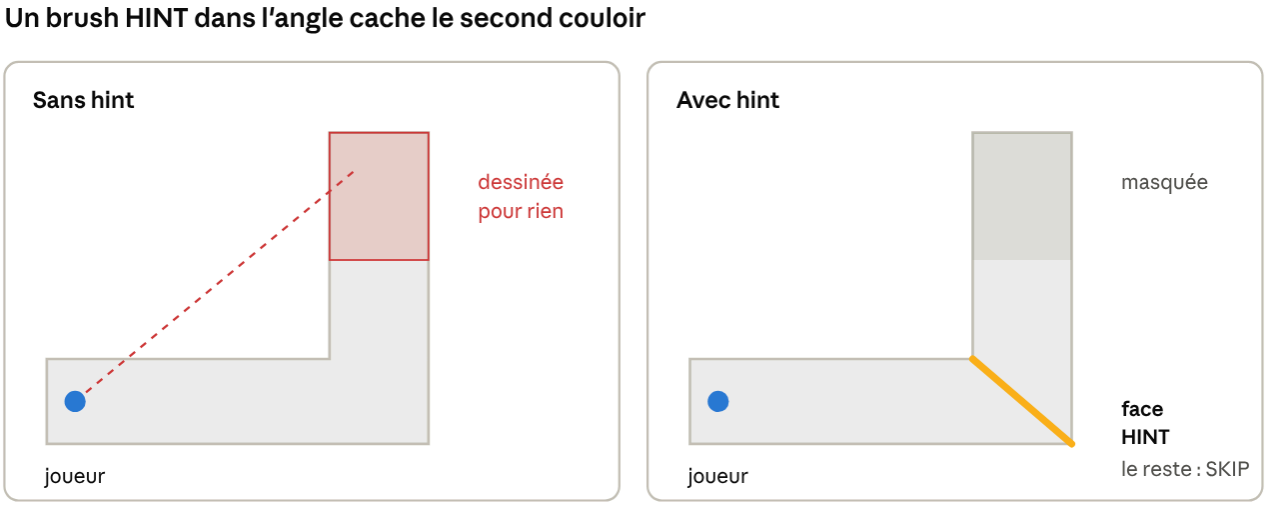
> **Info :** `func_detail` est un bloc du monde (il projette des ombres, il est solide) que VIS ignore (VDC : func_detail (GoldSrc)). C'est la solution moderne pour les détails, à condition d'utiliser VHLT.

> **Attention :** un `func_wall` ou un `func_detail` ne ferme jamais la map. Ne transformez pas un mur extérieur.

3. Soigner les textures

- Mettez **NULL** sur les faces d'entités que le joueur ne voit jamais (dessous d'une caisse posée, dos d'un `func_wall` collé au mur). Le compilateur supprime déjà celles des blocs du monde.
- Une face est redécoupée tous les 240 pixels de texture : une texture de 240 pixels, ou une échelle un peu plus grande, évite des coupes inutiles (TWHL).
- Alignez les textures entre deux blocs voisins : un décalage crée des faces en plus.

4. Les hint brushes



hint brush · couloir en L vu de dessus

Un hint brush force VIS à couper ses zones à un endroit précis (TWHL : Hint Brushes).

1. Créez un bloc fin qui traverse l'angle en diagonale, du coin intérieur au coin extérieur, et du sol au plafond.
2. Texturez tout le bloc en **SKIP**.
3. Avec l'outil Texture Application, mettez **HINT** sur une seule des grandes faces de la diagonale.
4. Compilez avec un VIS complet (pas `-fast`) et comparez les `r_speeds` au même endroit.

Ce qui marche le mieux, d'après les tests de TWHL :

- **couper les angles** : un hint qui passe par l'angle le plus proche du joueur, en touchant murs, sol et plafond ;

- **fermer les embranchements** de couloirs ;
- **un hint horizontal** au-dessus des têtes, quand des murs ne montent pas jusqu'au plafond.

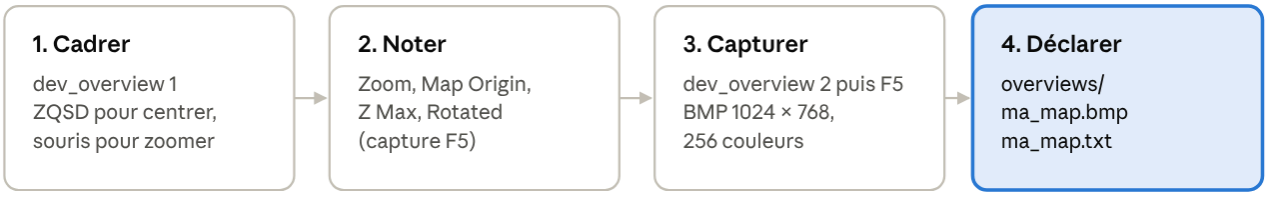
> **Attention :** mal placés, les hints peuvent **augmenter** les r_speeds et allongent la compilation. Testez chaque hint avec `r_speeds 1` et `gl_wireframe 2`.

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

5. Les overviews

L'overview est la carte vue de dessus qu'affichent le mode spectateur et la mini-carte. Sans elle, votre map n'a pas de plan. Elle se fabrique dans le jeu lui-même (TWHL : Creating Overviews).

Une overview en quatre étapes : cadrer, noter, capturer, déclarer



overview · 4 étapes

Récupérer l'overview

1. Dans les options vidéo du jeu, passez en **OpenGL** et en **1024 × 768**. La commande ne marche pas en mode Software.
2. Lancez votre map. Masquez l'interface avec `hud_draw 0`, ou éloignez-vous de la zone d'achat.
3. Dans la console, tapez `dev_overview 1`. Centrez la map avec les touches de déplacement, zoomez avec les boutons de la souris.
4. Notez les valeurs affichées en haut de l'écran : **Zoom**, **Map Origin** (trois nombres), **Z Min**, **Z Max**, **Rotated**. Une capture F5 les enregistre.
5. Tapez `dev_overview 2` pour masquer les chiffres, puis **F5**. Si c'est trop sombre, `r_fullbright 1` avant la capture.

> **Info :** un commentateur de TWHL signale que l'overview ne s'affiche correctement que si le jeu est lancé avec `-dev` et que vous rejoignez les spectateurs avant de taper la commande (TWHL).

Convertir l'overview

La capture apparaît dans le dossier du jeu, sous un nom comme `ma_map0000.bmp`. Ouvrez-la dans un logiciel d'image, retirez les lignes vertes éventuelles, puis enregistrez-la en **BMP 8 bits, 256 couleurs, 1024 × 768**, sous le nom `ma_map.bmp`, dans `cstrike/overviews/`.

Éditer les propriétés

Copiez le fichier .txt d'une map officielle (celui de de_dust par exemple), renommez-le `ma_map.txt` dans le même dossier, et remplacez les valeurs par les vôtres :

```
// overview de ma_map

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

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

- **ZOOM** : la valeur Zoom notée.
- **ORIGIN** : les trois valeurs de Map Origin.
- **ROTATED** : la valeur Rotated.
- **IMAGE** : le chemin de votre BMP.
- **HEIGHT** : la valeur Z Max.

Les nombres ci-dessus ne sont qu'un exemple de format : utilisez ceux affichés pour votre map.

> **Attention :** l'overview ne montre que l'étage le plus haut d'une map à plusieurs niveaux (TWHL). Et pensez à ajouter `overviews/ma_map.bmp` et `overviews/ma_map.txt` au package de la map.

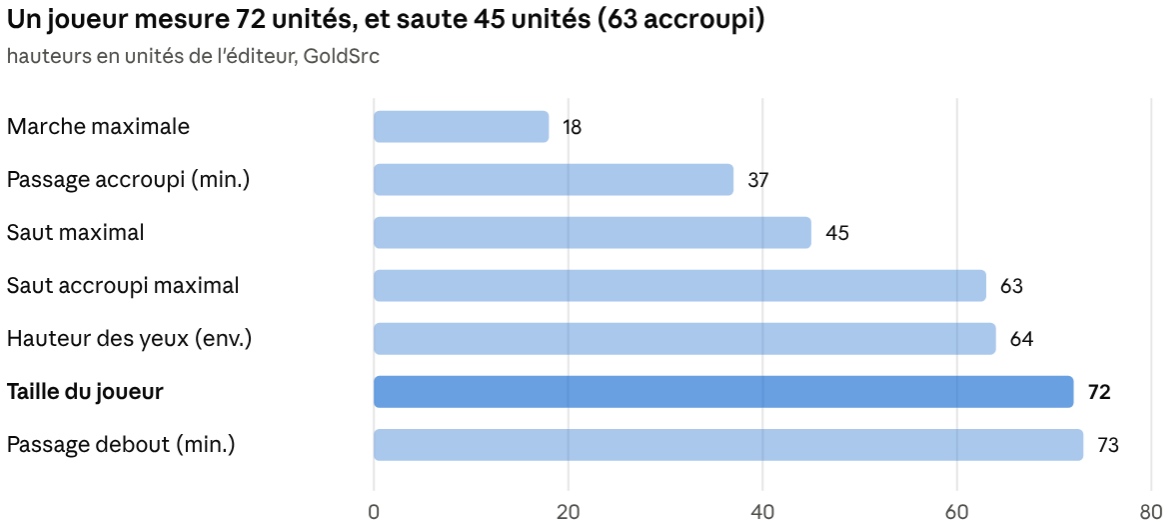
Source : TWHL : Creating Overviews.

Partie 4 : Annexes

Les mémos à garder sous la main.

1. Les longueurs

Tout se mesure en **unités** de l'éditeur. Ces chiffres sont à garder sous la main pour que les joueurs passent, montent et sautent là où vous le voulez (VDC : GoldSrc Dimensions).



dimensions du joueur · 7 hauteurs (VDC)

Les mesures du joueur

Mesure	Unités	Ce que ça veut dire
Marche maximale	18	Au-delà, il faut sauter
Largeur franchissable	32	Un trou plus étroit que 32 se traverse en marchant
Passage minimal en largeur	33	Il faut au moins 33 entre deux murs pour passer
Passage accroupi minimal	37	Hauteur minimale d'un conduit
Saut maximal	45	Hauteur d'un rebord atteignable en sautant
Saut accroupi maximal	63	Saut en s'accroupissant en l'air
Hauteur des yeux	64 environ	Hauteur de la caméra

Mesure	Unités	Ce que ça veut dire
Taille du joueur	72	
Passage debout minimal	73	Hauteur minimale d'une porte ou d'un couloir

> **Conseil :** ne construisez pas au millimètre près. Une porte de 96 à 112 de haut, un couloir de 64 à 128 de large laissent de la marge et paraissent naturels.

Les conversions

Une unité vaut à peu près **un pouce, soit 2,54 cm**. Certains l'estiment plutôt à 0,75 pouce (1,9 cm) : ce sont des approximations (VDC).

Unités	Environ
16	40 cm
64	1,6 m
128	3,3 m
1024	26 m

Calculé avec 1 unité = 2,54 cm.

La taille maximale d'une map

8192 × 8192 × 8192 unités, soit 4096 dans chaque direction depuis le centre de la grille. Au-delà, le jeu refuse de charger la map, même si J.A.C.K. vous laisse construire plus grand (VDC).

Source : VDC : GoldSrc Dimensions.

2. Les bots et waypoints

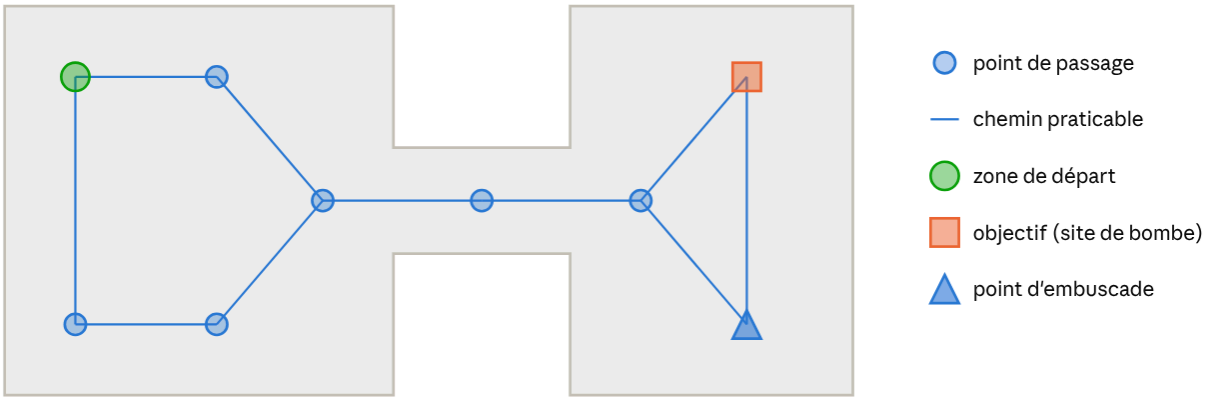
Tester une map seul ne dit pas grand-chose : il faut des adversaires. Or **Counter-Strike 1.6 n'a pas de bots officiels** (VDC : Counter-Strike).

Les solutions

Solution	Principe
Condition Zero	Les maps de CS 1.6 se lancent dans Counter-Strike: Condition Zero, qui a ses propres bots (CS_Bot) (VDC)
YaPB	Bots tiers, libres (licence MIT), pour CS 1.6 et Condition Zero
Autres bots tiers	Listés par la communauté bots-united, citée par VDC

Qu'est-ce qu'un waypoint ?

Les bots se déplacent sur un réseau de points reliés



Vue de dessus : sans ce réseau, un bot ne sait pas où aller.

waypoints · réseau de navigation des bots

Un bot ne « voit » pas la map comme un joueur. Il se déplace sur un réseau de **points de passage** (waypoints, ou « graphe de navigation » chez YaPB) reliés par des chemins. Certains points ont un rôle : zone de départ, objectif (site de bombe, otages), point d'embuscade, échelle, saut.

Placer et lier des waypoints

Le principe est le même pour tous les bots tiers :

1. Lancez la map avec le bot installé et activez son éditeur de waypoints (une commande de console propre à chaque bot).
2. Parcourez la map et posez un point à chaque endroit important : carrefours, portes, haut et bas des escaliers et des échelles.

3. Vérifiez que les liens entre les points passent bien là où un joueur peut aller.
4. Marquez les points spéciaux (objectifs, embuscades, échelles).
5. Enregistrez : un fichier de waypoints est créé pour la map.

Les commandes exactes changent d'un bot à l'autre : suivez la documentation du bot choisi. Pour YaPB, les graphes de navigation sont gérés dans un dépôt à part, où l'on peut aussi en demander pour une map (GitHub : YaPB).

Livrer ses waypoints

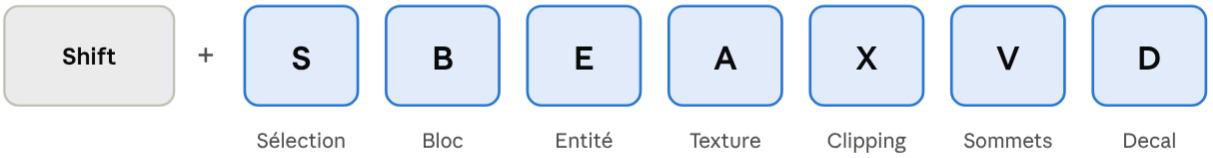
Joignez le fichier de waypoints à votre package (Partie 2, chapitre 19) : les joueurs qui utilisent le même bot n'auront pas à le refaire.

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

3. Raccourcis clavier de J.A.C.K.

J.A.C.K. reprend les raccourcis de Hammer (VDC : Hammer Hotkey Reference). Selon votre version, certains peuvent différer : vérifiez dans les menus, où chaque raccourci est affiché à côté de sa commande.

Shift + une lettre : les outils de la barre de gauche



Raccourcis hérités de Hammer ; ils se modifient dans Tools > Options.

outils · 7 raccourcis Shift

Les outils

Raccourci	Outil
Shift+S	Sélection
Shift+G	Loupe (zoom)
Shift+C	Caméra
Shift+E	Entité
Shift+B	Bloc

Raccourci	Outil
Shift+A	Application de texture
Shift+T	Appliquer la texture courante à la sélection
Shift+D	Decal
Shift+X	Clipping
Shift+V	Sommets (vertex)

La sélection

Raccourci	Action
Ctrl+clic	Ajouter ou retirer un objet de la sélection
Échap ou Shift+Q	Vider la sélection
Page précédente / suivante	Passer d'un objet à l'autre sous le curseur
Shift + glisser	Cloner
Flèches (vue 2D)	Déplacer d'un pas de grille

L'application de texture (dans la vue 3D)

Raccourci	Action
Clic gauche	Sélectionner une face
Ctrl+clic	Ajouter une face
Shift+clic	Sélectionner toutes les faces du bloc
Clic droit	Appliquer la texture courante à la face visée
Alt+clic droit	Appliquer texture et alignement

Les menus (utilisés dans ce cours)

Raccourci	Action
Alt+Entrée	Propriétés de l'objet

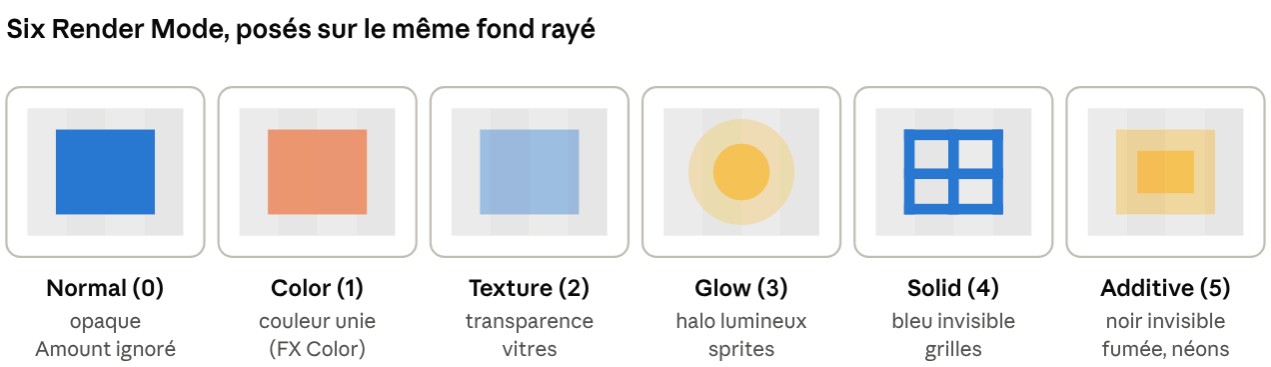
Raccourci	Action
Ctrl+T / Ctrl+W	Lier à une entité / rendre au monde
Ctrl+G / Ctrl+U	Grouper / dégrouper
Ctrl+Shift+H	Hollow (creuser)
Ctrl+Shift+C	Carve (découper)
Ctrl+M	Transformation précise
Ctrl+Shift+V	Collage spécial
Ctrl+B	Caler sur la grille
Ctrl+Shift+G	Aller au bloc numéro…
Alt+P	Vérifier les problèmes
[et]	Grille plus fine / plus large
Z	Vol libre dans la vue 3D
F9	Compiler (Run Map)

Source : VDC : Hammer Hotkey Reference.

4. Propriétés de rendu

Presque toutes les entités visibles (blocs, models, sprites) ont quatre propriétés de rendu (TWHL : Render Properties) :

- **Render Mode** (`rendermode`) : la façon de dessiner l'entité ;
- **FX Amount** (`renderamt`) : de 0 (invisible) à 255 (opaque), selon le mode ;
- **FX Color** (`rendercolor`) : une couleur R V B ;
- **Render FX** (`renderfx`) : un effet animé par-dessus.



Render Mode · 6 modes illustrés

Render Mode

Mode	Valeur	Formule (TWHL)	Usage
Normal	0	src	Rendu opaque, FX Amount ignoré
Color	1	c × a + fond × (1 − a)	Remplace la texture par FX Color
Texture	2	src × a + fond × (1 − a)	Vitres (FX Amount autour de 100)
Glow	3	src × a + fond, sans test de profondeur	Halos de sprites, qui grossissent et s'estompent avec la distance
Solid	4	src × alpha de la texture + fond × (1 − alpha)	Textures masquées `{` (FX Amount 255)
Additive	5	src × a + fond	Fumée, hologrammes, sprites lumineux : le noir devient transparent

Dans les formules, *src* est l'entité, *fond* ce qu'il y a derrière, *a* le FX Amount et *c* le FX Color.

Render FX

Valeur	Effet
0	Normal
1 à 4	Pulsations lentes ou rapides, étroites ou larges
5, 6	Disparaît en fondu, lentement ou vite
7, 8	Apparaît en fondu, lentement ou vite
9 à 11	Stroboscope de plus en plus rapide

Valeur	Effet
12, 13	Scintillement lent ou rapide
14	Constant Glow : annule le grossissement du mode Glow
15	Distort
16	Hologram (distorsion et fondu)
19	Glow Shell : aura autour d'un model

Valeurs d'après TWHL : func_wall et TWHL : Render Properties.

Les réglages à retenir

Effet voulu	Render Mode	FX Amount
Vitre	Texture	100 environ
Grille, clôture, feuillage (texture `{`)	Solid	255
Halo de lampe (env_sprite)	Glow	255, avec Render FX Constant Glow si la taille ne doit pas changer
Faisceau, fumée	Additive	100 à 200
Objet invisible (à révéler par env_render)	Texture	0

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

5. Sources et liens utiles

Toutes les pages consultées pour ce cours, regroupées par site. Le plan et l'esprit du cours viennent du tutoriel « Une map avec Hammer ! » de Loulimi et Mathieu Nebra (OpenClassrooms, licence Creative Commons, mise à jour du 24/08/2011).

Outils

- 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

Configuration et compilation

- TWHL : Setup Guide Hammer/J.A.C.K.
- TWHL : Compile
- TWHL : Compiling 101
- TWHL : Setting up TrenchBroom (section compilation)
- VDC : HLBSP
- VDC : compile settings de SirYodaJedi
- 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

Entités

- TWHL : Doors
- TWHL : func_door_rotating
- TWHL : func_wall
- TWHL : Ladders et 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