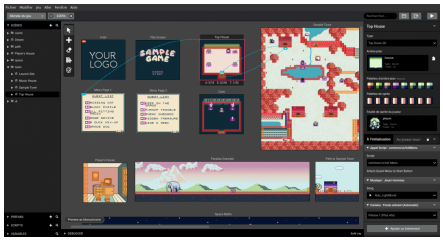
gameboy, jeu-video, homebrew, gamejam, em

# GB Studio



GB Studio est un outil (*visual game builder*) qui permet de créer des jeux vidéo, pour Game Boy ou Game Boy Color, sans nécessiter de compétences en programmation. Les jeux créés sont exportables facilement pour le web ou sous forme de «rom» qui peut être inscrite sur cartouche et joués sur une véritable console (cf. GBxCart RW et EverDrive-GB X5)

GB Studio est un logiciel libre qui peut être installé sur Linux, MacOS ou Windows, développé à l'origine par Chris Maltby.

- Site principal : https://www.gbstudio.dev/
- Téléchargement : https://chrismaltby.itch.io/gb-studio
- Code source : https://github.com/chrismaltby/gb-studio

## Caractéristiques des jeux

- Définition : 160 x 144 px
- 3 couleurs pour les sprites
- 4 couleurs pour les backgrounds
- musique au format UGE composable dans l'éditeur
- effets sonores : .wav, 3.64 seconds max, 8-bit mono, à placer dans le dossier /assets/sounds

Pour chaque scene, il y a un nombre max de sprites affichables et de tuiles utilisables pour le background

## Utilisation : assets

Dans le contexte du jeu vidéo, le terme assets désigne éléments qui composent un jeu :

- éléments graphiques (sprites, images de fond, textures)
- audio (musiques, effets sonores)
- texte (scripts, dialogues)
- etc.

### Palettes

En mode Game Boy, la palette de background (à gauche) comporte 4 couleurs, la palette de sprites (à droite) en comporte 3, la dernière couleur indique la transparence.
Palette background : #071821, #306850, #86c06c, #e0f8cf
Palette sprites : #071821, #86c06c, #e0f8cf, #65ff00 (transparent)




- #071821 : rgb(7, 24, 33)
- #306850 : rgb(48, 104, 80)
- #86c06c : rgb(134, 192, 108)
- #e0f8cf : rgb(224, 248, 207)
- #65ff00 : rgb(101, 255, 0)

En mode Game Boy Color (TODO)

## Sprites

### Types de sprites

- statiques (16 x 16 px)
- acteurs : 3 positions (48 x 16 px)
- animés : entre 2 et 25 frames (= entre 32 x 16 px et 400 x 16 px)
- acteurs animés : 6 positions ( 96 x 16 px)

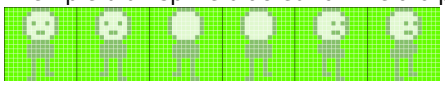
Pour dessiner les sprites, on utilise LibreSprite, attention la palette Gameboy de LibreSprite n'a pas les bonnes couleurs, on les a refaites :

gb_palette.zip

### Utiliser l'éditeur de sprites de GB Studio

- https://www.gbstudio.dev/docs/assets/sprites
- https://gbstudiocentral.com/tips/basics-the-sprite-editor/
- https://gbstudiocentral.com/tips/managing-sprite-tiles/

Exemple d'un sprite d'acteur animé à 6 positions (fichier .PNG de 96×16 pixels) :



Positions : 2 frames «de face en mouvement», 2 frames «avançant vers le haut», 2 frames «avançant à droite» (dont seront déduites les positions «avançant à gauche»). Le vert vif clair correspond à la couleur transparente.

## Images de fond (backgrounds)

Plusieurs règles pour les images de fond :

- L'image de background sera divisée en jeu de tuiles de 8x8px, sa définition doit donc être un multiple de 8 en largeur et en hauteur.
- L'image de background a une définition minimale de 160x144px (c'est la taille de l'écran d'une Game Boy!)
- La largeur et la hauteur de l'image de fond ne doivent pas dépasser 2040px.
- Le résultat de la largeur multipliée par la hauteur doit être inférieur ou égal à 1 048 320. PAr exemple une image de 2016 pixels de large multiplié par 520 pixels de haut (2016 x 520 = 1048320)
- L'image de fond doit comporter au maximum 192 tuiles différentes (ou 256 si la scène ne comporte pas de dialogues). Par exemple, une image de 160x 144px comporte 360 tuiles de 8x8px, c'est trop! Il faut donc réduire le

nombre de tuiles (voir ci-dessous)

Optimiser ses images

- **outil en ligne pour compter les tuiles** : http://gb-studio-tile-count.glitch.me/
- **application pour compter les tuiles** : https://momeka.itch.io/gameboy-tools (application .exe)
- https://gbstudiocentral.com/tips/basics-backgrounds/
- https://gbstudiocentral.com/tips/converting-images-for-gb-studio/

Pour les scènes de type «Logo» : 160 x 144 px, 4 couleurs, pas nécessaire de faire des tuiles mais pas de player sur la scène

## Musique

La musique peut être au format .MOD ou .UGE, mais un projet ne peut comporter qu'un seul de ces formats.
L'éditeur intégré permet de réaliser des musiques au format .UGE. Pour le format .MOD, il faut utiliser un éditeur externe.

- Utilisation de l'éditeur .UGE intégré : https://www.gbstudio.dev/docs/assets/music/music-huge/
- Utilisation du format .MOD : https://www.gbstudio.dev/docs/assets/music/music-gbt

4 canaux (2 ondes carrées pour les mélodies, 1 noise, 1 forme d'onde sawtooth pour les basses), chacun de ces canaux a ses propres caractéristiques (instruments, etc.)
L'édition peut se faire sous la forme d'un piano roll, ou d'un tracker.
Pas de BPM, juste une vitesse indiquant combien de frames par pulsation.
Des effets peuvent être ajoutés sur chaque note

Ressources musique :

- https://gbstudiocentral.com/tips/faqs-for-gb-studio-music-composers/
- https://nickfa.ro/wiki/HUGETracker
- https://tronimal.de/game-boy-music-software/
- https://gbstudiocentral.com/resources/

## Effets sonores

Les effets sonores sont à placer dans le dossier /assets/sounds, plusieurs types de fichiers sont possibles :

- **.wav**, format .WAV, 3.64 secondes max, 8-bit mono, à placer dans le dossier
- **.vgm**, fichier audio VGM audio file, uniquement pour le format de jeu Game Boy
- **.sav**, effet sonore FX HAMMER (FX Hammer est un logiciel qui fonctionne directement sur cartouche : guide fx hammer).

Le format .sav est le seul format natif, à privilégier!

Ressources effets sonores

- https://gbstudiocentral.com/tips/sound-effects-and-music-in-gb-studio-3-1/

### Conversion des sons pour effets sonores

```
# Créer un dossier "8bit" s'il n'existe pas
mkdir -p 8bit

# Boucle sur chaque fichier .wav dans le dossier actuel
for file in *.wav; do
  # Transformer le fichier en wav mono 8 bits avec sox
  sox "$file" -b 8 -r 44100 "8bit/${file%.*}_8bit_44100_mono.wav" channels 1
done
```

La boucle `for` parcourt tous les fichiers .wav dans le dossier actuel. La commande sox utilisée à l'intérieur de la boucle convertit chaque fichier en wav mono 8 bits 44100. La sortie est enregistrée dans le dossier “8bit”. La substitution de variables `${file%.*}` retire l'extension “.wav” du nom de fichier.

# Utilisation : composer un jeu

Un jeu est composé de plusieurs scènes, reliées par des évènements déclencheurs, dans lequelles des acteurs interagissent. La logique du jeu est assurée par des scripts associés aux scènes, aux acteurs ou aux évènements.

Les zones de collision des scènes sont réglées dans l'éditeur.

A chaque scène, on peut associer un arrière plan, un type, une feuille de sprite au personnage et des scripts à l'initialisation (par exemple : changer la musique)

Les acteurs sont soit des personnages, soit des objets avec lesquels le joueur interagit

### Scenes

Les scènes composent le jeu, chacune est un «écran» ou un inviers du jeu, elle ssont reliées par des déclencheurs (*triggers*) qui déclenche un évènement «change scene». Dans l'éditeur ces liens entre scènes sont représentés par des flèches bleues en pointillés.

Plusieurs types de scènes sont proposés qui correspondent à un type de jeu :

- Top Down 2D
- Platformer
- Aventure
- Point'n'click
- Shoot'em'up
- Logo

Chaque scène peut avoir un maximum de 20 acteurs, 30 triggers/déclencheurs et entre 64 et 192 tuiles de sprites selon la complexité du projet. En sélectionnant une scène, dans la barre grise sous la scène, on retrouve ces infos :

- A : représente le nombre d'acteurs de la scène
- F : nombre de frames de sprites utilisées par les différents acteurs
- T : nombre de triggers de la scène

Tout ça est à nuancer, voir la doc : https://www.gbstudio.dev/docs/project-editor/scenes

# Utilisation : scripting

TODO

Les scripts permettent de contrôler le déroulement du jeu en fonctions des interactions du joueur. Par exemple, ils sont utilisés pour relier les scènes entre elles, modifier des variables, ajouter des dialogues, etc.

Les scripts peuvent être ajoutés aux scènes, aux acteurs ou aux déclencheurs (triggers). Pour afficher les scripts liès à un de ces éléments, le sélectionner dans la vue générale.

**scène**

- «à l'initialisation»
- «au joueur touché»

**acteur**

- «à l'interaction»
- «à l'initialisation»
- «à l'actualisation»
- «au touché»

**déclencheur**

- «à l'entrée»
- «à la sortie»

Ressources scripts

- https://www.gbstudio.dev/docs/scripting/custom-scripts/
- https://gbstudiocentral.com/tips/basics-enemy-movement/
- https://gbstudiocentral.com/tips/zen-and-the-art-of-retro-level-design-in-kudzu/
- https://gbstudiocentral.com/tips/dwf-chapter1-research/
- https://gbstudiocentral.com/tips/basics-custom-events/
- https://gbstudiocentral.com/tips/basics-switches/

# Utilisation : build & export

Le jeu peut être «buildé» à tout moment avec le bouton «play» en haut à droite, il s'ouvrira alors dans un émulateur (qui a quelques difficultés avec le son).
On peut aussi l'exporter sous différente forme depuis le menu «jeu/exporter sous» :

- Encapsulé dans un fichier HTML, ce qui permet d'y jouer dans n'importe quel navigateur et de le mettre sur le web. Dans ce cas un dossier «web» est créé dans le dossier «build» du projet. Le dossier inclut l'émulateur BinjGB.
- Sous forme de ROM qui peut ensuite être jouée dans un émulateur ou mis sur cartouche ou sous forme de ROM pour console Analogue Pocket.

Pour jouer au jeu créé, on peut utiliser :

- sur ordi : émulateur, navigateur web
- sur mobile : émulateur, navigateur web
- sur console : cartouche (cf. GBxCart RW), cartouche à carte microSD (cf. Krikkz EverDrive-GB X5)

## Émulateur intégré

Quand on «build» le jeu, il s'affiche dans une fenêtre d'émulation. C'est BinjGB qui assure l'émulation.

**Haut, Bas, Gauche, Droite** : flèches directionnelles
**Bouton A** : Z
**Bouton B** : X
**Bouton Start** : ↵ Enter
**Bouton Select** : ⇧ Shift

## Export HTML

Quand on **exporte un jeu en html**, il faut lancer un serveur pour le tester en local (et ne pas se confronter aux règles CORS du navigateur), par exemple en utilisant le serveur intégré à python

- se placer dans le dossier du jeu (où se trouve le fichier index.html)
- `python3 -V` (si python 3.x), ou `python -V` (si python 2.x)
- `python3 -m http.server` (si python 3.x) ou `python -m SimpleHTTPServer` (si python 2.x)

Pour publier sur itch.io :

- exporter en html, puis
- zipper le dossier build/web, uploader sur itch.io comme un jeu HTML, taille de viewport recommandée : 480px x 432px.

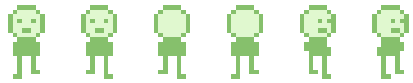
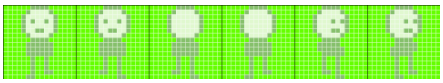
# FAQ / Tips

**Jouer de la musique une seule fois**
Apparemment le player n'est pas fait pour ça, il faut créer un pattern vide et à la fin de celui-ci renvoyer vers le pattern vide, cf. https://github.com/chrismaltby/gb-studio/issues/1191#issuecomment-1284881468

**Représenter des sprites sur une grille**
Utile d'un point de vue pédagogique! Les images des sprites sont très petites (16x16px), il faut les agrandir et ajouter une grille (avec le script grid de Fred Weinhaus)

```
convert frogue.png -filter point -resize 1000% f1.png          # multiplier la définition par 10 sans interpolation
```

```
convert f1.png -background '#65ff00' -layers flatten f2.png     # ajouter un fond vert
sh ./grid -s 10 -c white -o 0.8 f2.png f3.png                   # ajouter une grille blanche autour de chaque pixel
sh ./grid -s 160 -c black f3.png frogue_x10_grille.png          # ajouter une grille noire autour de chaque frame
```

Original  Résultat 

# Advanced mode

Explaining VRAM and tile limits in GB Studio. https://gbstudiolab.neocities.org/guides/vram-tile-limits
Game size checker : https://gbstudiolab.neocities.org/resources/game-size-checker (utile avant optimisation)

### Détails techniques croustillants sur GB Studio

Le projet est open-source, développée principalement par Chris Maltby depuis 2019. Il est réalisé en React à l'aide de l'outil Electron ainsi qu'une architecture Redux et est disponible sur Github.

GB Studio fonctionne sur la machine virtuelle GBVM

- https://github.com/chrismaltby/gbvm
- https://www.gbstudio.dev/docs/scripting/gbvm/
- https://gbstudiocentral.com/tips/understanding-gbvm/
- https://gbstudiocentral.com/tips/learning-gbvm/

GB Studio fonctionne comme une interface visuelle avec GBDK (Game Boy Developer's Kit) qui est un set non-officiel d'outils et de bibliothèques en langage C pour développer des logiciels sur Game Boy (et pour d'autres consoles).

- https://github.com/gbdk-2020/gbdk-2020
- **exemples de jeux réalisés avec GBDK / GBDK-2020** : https://gbdk-2020.github.io/gbdk-2020-gallery/

# Ressources

- https://chrismaltby.itch.io/gb-studio
- https://www.gbstudio.dev/docs/
- https://marcosecchi.github.io/resources/pages/retrogaming_gbstudio.html
- https://toxworks.itch.io/portal-gun-system-in-gb
- https://itch.io/game-assets/tag-gb-studio
- https://gbstudiocentral.com/topics/tips/
- https://docs.google.com/spreadsheets/d/1d2F5hSEMt6nkacw-qVnYlT3IPHqmCCaLFhRboC5xxc0/edit#gid=0
- exemple jeu complet : https://halmic.medium.com/taco-neko-development-5c1fb9e6372c
- plugins : https://github.com/shin-gamedev/gbs-plugins

#### Assets

- **ressources graphiques et sonores à utiliser dans les jeux** : https://opengameart.org/
- **ressources graphiques** :
  - https://www.spriters-resource.com/
  - https://kenney.nl/
  - https://craftpix.net/
- type RPG (top down 2D) https://kenney.nl/assets/monochrome-rpg
- https://github.com/DeerTears/GB-Studio-Community-Assets

#### Tutos vidéo

- https://www.youtube.com/watch?v=0xO5jCqs3As
- https://www.youtube.com/playlist?list=PLmac3HPrav--Q4QKUVknwwMSNk1YECFKT
- https://www.youtube.com/watch?v=JMZkrucEkBU&list=PLEeET_LWnAgsE0hD_kBSz1QoE4XnuwqoG

Article extrait de : http://www.lesporteslogiques.net/wiki/ - **WIKI Les Portes Logiques**
Adresse : **http://www.lesporteslogiques.net/wiki/ressource/logiciel/gb_studio/start**
Article mis à jour: **2026/02/20 10:59**