

Sylvain Seccia



ÉDITIONS SECCIA

TABLE DES MATIÈRES

INTRODUCTION	1
INSTALLATION DU LOGICIEL	2
LICENCE	2
COMPILATION DU PROJET UNITY	2
PREFERENCES.....	3
DOCUMENTATION.....	3
MES JEUX CREEES AVEC SECCIA.DEV	4
GAMEPLAY	7
QU'EST-CE QU'UN JEU VIDEO ?	7
QU'EST-CE QU'UN JEU D'AVENTURE NARRATIF ?	8
LES QUATRE PROFILS DE GAMEPLAY	8
INTERFACE.....	11
MENU PRINCIPAL.....	12
BARRE D'OUTILS	12
BARRE DE NAVIGATION	13
PROJET	15
STRUCTURE	15
PARAMETRES	17
PARAMETRES DES CONFIGURATIONS	23
CREDITS	25
TERMINAL	26
SCÉNARIO.....	27
BOITE NODALE	27
SECTION.....	32
SCRIPT.....	33
EXPORTATION	33
LES BONNES PRATIQUES	34
PLAYER.....	39
ICONE.....	39

DEFILEMENT & ZOOM	40
TRAJECTOIRE	40
INTEGRATION PAR LE CODE	41
OBJET	43
OBJET	43
ANIMATION.....	47
SOUS-OBJET	50
INVENTAIRE	50
CLONE.....	51
MASQUE	51
PAROLE	53
INTEGRATION PAR LE CODE	54
LES BONNES PRATIQUES	54
SCÈNE	57
SCENE	57
STILL	64
OBJET DE SCENE	65
LABEL.....	69
DOOR	72
SHOT	74
WALL.....	75
GRID & CELL	76
PATH & SPOT	81
DRAG & DROP.....	83
TIMELINE	84
INTEGRATION PAR LE CODE	88
GESTIONNAIRE DE SCENES	88
LES BONNES PRATIQUES	89
DIALOGUE	91
DIALOGUE.....	91
BOITE ROUGE « CHOICE »	92
BOITE BLEUE « SENTENCE »	95
INTERFACE	98

STYLE	99
INTEGRATION PAR LE CODE	103
RÔLE.....	105
LANGAGE	105
ACTIONS	107
CONDITIONS	117
ÉVÉNEMENTS	122
INTEGRATION PAR LE CODE	129
EFFET.....	131
OUTPUT.....	131
BRIGHTNESS, CONTRAST, SATURATION	132
BLUR.....	132
COLORIZATION	133
CURVE.....	133
FIRE	133
GRAIN	133
GRAYSCALE.....	134
LOOK UP TABLE.....	134
PARALLAX.....	134
RAIN	134
INTEGRATION PAR LES EDITEURS.....	135
INTEGRATION PAR LE CODE	135
CINÉMATIQUE	137
VIDEO, AUDIO ET SOUS-TITRE	137
ENCODAGE D'UN FICHER VIDEO MP4	138
INTEGRATION PAR LE CODE	138
AUDIO	139
MUSIQUE ET AMBIANCE SONORE	139
BRUITAGE	139
VOIX	140
SAUVEGARDE	141
SAUVEGARDE LOCALE	141

SAUVEGARDE EN LIGNE142

CHAPITRES143

DONNEES PARTAGEES.....143

LOCALISATION 145

ÉDITION INTERNE.....145

ÉDITION EXTERNE146

ALPHABET LATIN, CYRILLIQUE ET AUTRES.....147

SCRIPT 149

SYNTAXE149

VARIABLE152

ASSISTED SCRIPTING.....154

BUILD 157

LIVE, DEBUG ET RELEASE.....157

ANDROID158

IOS.....158

WEBGL.....158

POST-BUILD.....159

LIGNE DE COMMANDE159

PROJET UNITY..... 161

FONCTIONS EXPOSEES.....161

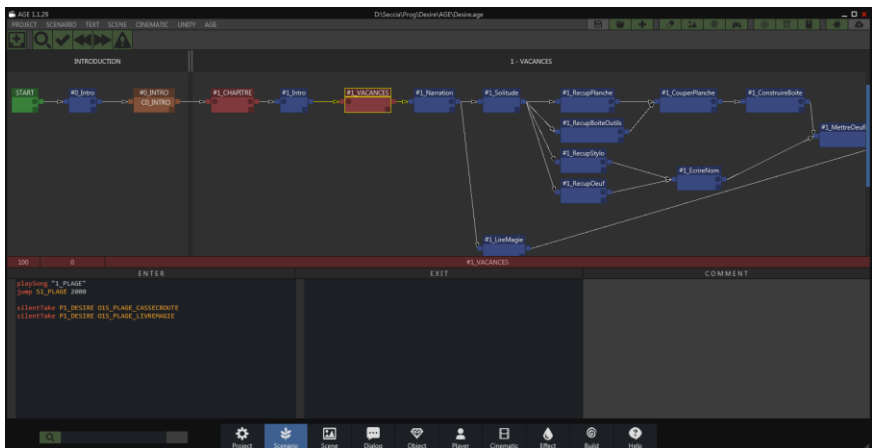
CALLBACK162

INTERLUDE162

INTRODUCTION

seccia.dev est un logiciel gratuit permettant de créer des jeux d'aventure 2D sans avoir besoin au préalable de compétences techniques en matière de programmation. Quelques notions de Scripting suffisent pour maîtriser toutes les fonctionnalités. Les outils proposés sont très simples d'utilisation et s'adressent avant tout aux non-programmeurs, et plus précisément aux scénaristes, artistes et game designers voulant créer leurs propres jeux d'aventure. À l'origine, je l'ai conçu comme outil interne pour faciliter le développement de mon premier jeu d'aventure **Désiré** de 2012 à 2016. Mon but était d'adopter une approche scénaristique pour me rapprocher le plus possible de la nature du jeu que je souhaitais réaliser.

Bien que **seccia.dev** soit un **IDE** propriétaire développé en **C++** pour **Windows** uniquement, les jeux générés peuvent tourner sur de nombreux systèmes d'exploitation puisque que le runtime a été développé avec **Unity**. Ce qui signifie que vous devez au préalable générer les builds du projet **Unity** présent dans le dossier d'installation du logiciel. Afin d'économiser du temps précieux sur vos projets, **seccia.dev** intègre un outil pour lancer la compilation du projet **Unity**, ainsi que l'intégration automatique des builds générés. Cette contrainte est due aux CGU qui imposent au développeur d'utiliser sa propre licence **Unity**. Cela dit, à l'installation, seule la version **Windows** est précompilée et ne doit être utilisée qu'à titre d'essai en phase de développement de vos projets.



Installation du logiciel

Avant de pouvoir profiter des fonctionnalités du logiciel, il est nécessaire de suivre quelques consignes afin de s'assurer que tous les composants soient correctement installés.

Le logiciel requiert :

Windows 11 à jour de préférence
Une résolution **Full HD** ou plus
Et idéalement un **SSD**

Et l'installation de l'application **Seccia Launcher** disponible à l'adresse suivante :

- <https://www.seccia.com/download/launcher>

Les mises à jour seront disponibles gratuitement. Vous serez tenu informé par une notification qui apparaîtra au sein de l'interface de l'application **Seccia Launcher** à chaque fois qu'une nouvelle version sera publiée.

Cet ouvrage fait référence à la version **2.0.7**.

Licence

Je tiens à préciser que le logiciel est entièrement gratuit. Aucune limitation fonctionnelle n'est appliquée volontairement dans le but de brider le produit. En revanche les services proposés peuvent être payants. Vous pouvez développer et finaliser vos jeux gratuitement pour une utilisation personnelle ou commerciale. Pour plus d'informations à ce sujet, je vous invite à vous rendre sur le site officiel.

Compilation du projet Unity

La première étape consiste à télécharger et installer, dans le dossier par défaut, la version actuelle du programme **Unity Hub** à l'adresse suivante :

- <https://www.unity.com>

Parmi la liste de modules proposée lors de l'installation, sélectionnez **Android** (sans oublier les sous-modules contenant les **SDK**), **iOS**, **WebGL** et **Windows**.

Une fois l'installation réussie, la deuxième étape consiste à lancer **seccia.dev** et ouvrir le menu **UNITY/Unity project** pour sélectionner les plateformes

souhaitées et lancer la compilation en cliquant sur le bouton **Build selected platforms**. Puis patientez...

Notez que **seccia.dev** pointera toujours vers la version d'**Unity** la plus récente installée sur votre machine, à moins de définir un chemin spécifique dans les préférences du logiciel.

Si vous êtes amené à effectuer des modifications sur le projet **Unity** pour apporter de nouvelles fonctionnalités à votre jeu, ce qui est tout à fait autorisé par les conditions d'utilisation, je vous déconseille fortement de modifier les propriétés du projet **Unity** au risque de provoquer des erreurs de compilation.

Préférences

Les préférences sont accessibles depuis le menu **IDE/Preferences**. Si vous rencontrez des problèmes de stabilité, vous pourrez désactiver la prise en charge du multithreading.

Documentation

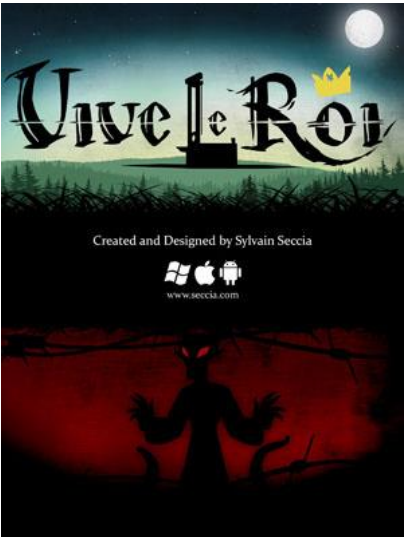
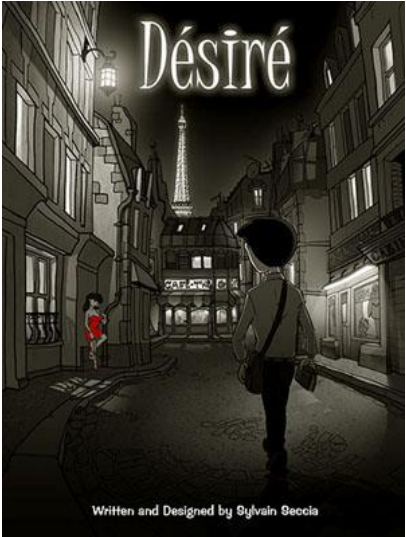
Toutes les fonctions du langage sont documentées et accessibles depuis la page d'aide du logiciel.

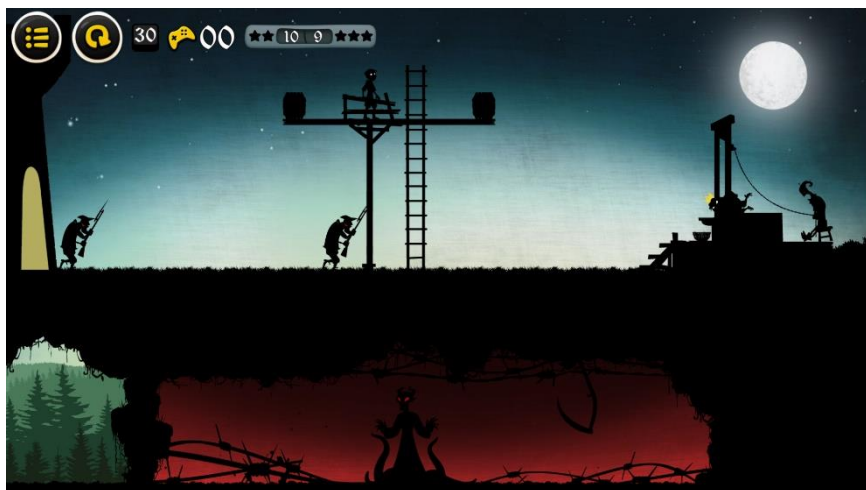
La touche **F1** est un raccourci et peut même interagir avec les scripts. Il vous suffit de placer le curseur de la souris sur le nom d'une fonction ou d'un mot clé et d'appuyer sur la touche **F1** pour afficher la page d'aide en question.

Sur cette page, vous trouverez également les réseaux sociaux, les sites web de la communauté **seccia.dev** et les tutoriels qui vous aideront à progresser dans votre apprentissage et je l'espère, vous inciteront à faire évoluer la communauté.

N'hésitez pas à consulter la page régulièrement pour vous familiariser avec le langage et être au courant des nouveaux contenus.

Mes jeux créés avec seccia.dev

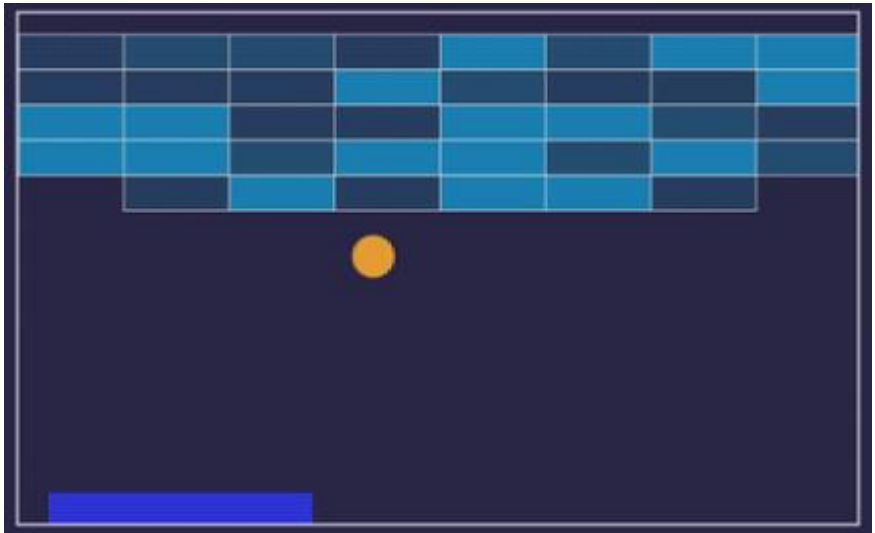




GAMEPLAY

Selon Marc Goetzmann et Thibaud Zuppinger (2016) : « le gameplay est l'articulation entre le *game*, les structures et règles de jeu, et le *play*, la façon dont le joueur s'approprie les possibilités du jeu en mettant au point ses propres stratégies, pour répondre aux contraintes que les règles constitutives du jeu lui imposent »

Avant de vous précipiter dans la réalisation de votre premier projet **seccia.dev**, il est important de bien discerner les caractéristiques d'un jeu vidéo d'aventure narratif pour mieux comprendre l'intérêt du gameplay et comment son rôle influencera fortement l'expérience du joueur.



Qu'est-ce qu'un jeu vidéo ?

Un jeu vidéo doit avant tout être bâti autour d'un gameplay. C'est la composante principale qui caractérise un jeu vidéo et qui devrait retenir toute votre attention dès le début du projet. De plus, les mécaniques et la jouabilité d'un jeu se construisent par itération en testant et retestant en boucle jusqu'à trouver le bon équilibre. Un jeu vidéo n'a ainsi pas besoin d'être soigné artistiquement pour être fun. Cette phase est souvent sous-estimée et les déceptions surgissent malheureusement en cours voire en fin de développement.

De plus, le jeu vidéo est une œuvre artistique parce qu’il se sert de plusieurs arts et disciplines pour exister. Il s’agit en l’occurrence du dessin, de l’architecture, de la photographie, de l’écriture, de la mise en scène, de la musique, des bruitages, des effets spéciaux et de nombreuses autres qualités. Ces éléments sont donc importants mais ne sont en fait qu’un habillage pour enrichir votre jeu, le rendre unique et parvenir à transmettre des émotions au joueur tout en donnant du sens à votre œuvre.

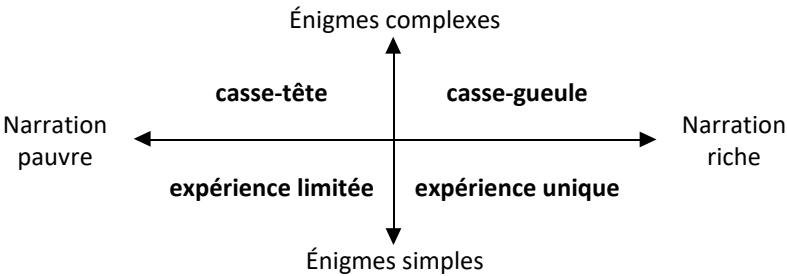
Qu’est-ce qu’un jeu d’aventure narratif ?

Le jeu d’aventure narratif tient une place particulière dans le monde du jeu vidéo. Est-ce que la narration d’un jeu d’aventure aurait plus d’importance que le gameplay ? C’est souvent le sentiment que l’on pourrait ressentir en jouant à un jeu narratif. D’ailleurs certains joueurs pensent même que le jeu d’aventure essentiellement basé sur un schéma narratif se rapprocherait plus d’un film interactif à cause d’un gameplay trop en retrait. Je pense qu’ils ont tort.

Je parle d’un genre à part parce que la narration d’un jeu d’aventure fait partie et ne peut être dissociée du gameplay. Si la trame de l’histoire est inintéressante ou maladroitement écrite, les mécaniques ne suffiront pas à rendre le gameplay suffisamment fun pour le joueur. Le sentiment que nous éprouvons est donc juste, parce que le gameplay d’un jeu d’aventure narratif repose à la fois sur une mise en scène et sur des énigmes.

Les quatre profils de gameplay

Si le gameplay d’un jeu d’aventure possède deux composantes que nous venons d’évoquer, c’est-à-dire la narration et les énigmes, nous pouvons alors proposer un modèle illustré de quatre profils à travers le schéma suivant :



Selon le niveau de complexité du gameplay en matière de narration et d'énigmes, vous obtiendrez un de ces quatre profils :

Expérience limitée	La narration est pauvre et les énigmes sont trop simples pour stimuler le joueur. Cette configuration débouchera probablement sur un jeu fade sans réel défi.
Casse-tête	Votre jeu est peu narratif et contient beaucoup d'énigmes complexes. Il s'agirait plutôt d'un puzzle et non d'un jeu d'aventure. Pour un Escape Game, vous pourriez être dans la bonne catégorie. Cela dit, des phases plus orientées puzzles au sein de votre aventure contrasteraient l'expérience du joueur.
Casse-gueule	C'est sûrement le profil le plus difficile à maîtriser car la narration est riche et les énigmes complexes. Je le déconseille à moins que les énigmes soient justifiées et servent réellement la mise en scène. Comme pour le profil précédent, vous pouvez l'utiliser à des moments clés de votre histoire. Trouvez le bon équilibre.
Expérience unique	Une narration riche avec des énigmes simples reste sans aucun doute le profil idéal pour réussir un jeu d'aventure. Écrire des énigmes simples ne signifie pas que vous devez tomber dans la facilité et vous appuyer sur des banalités. Cela ne signifie pas non plus que les énigmes doivent être faciles à résoudre. En fait, les tâches à effectuer doivent être courtes, simples et claires pour le joueur, et c'est la combinaison de ces petites énigmes qui l'incitera à se creuser les méninges pour trouver la solution sans tricher. La frustration intervient lorsqu'on a l'impression de ne plus avancer sur une longue période.

INTERFACE

seccia.dev propose une approche d'interface basée sur un concept de pages plutôt original et peu exploité dans les logiciels de développement d'applications. A contrario, le modèle de fenêtre par asset est un modèle plus répandu dans le domaine des **ID** et s'impose naturellement parce qu'il est relativement pratique et efficace dans la majorité des situations. Mais est-il efficace pour tous les environnements de développement ? Car autant pour un outil de programmation, il serait inimaginable de ne pas pouvoir ouvrir plusieurs codes sources à la fois sans perdre l'historique des modifications en cours ; autant pour un logiciel de montage vidéo, cela s'avère nettement moins pertinent de travailler simultanément sur plusieurs timelines. Il est vrai que **seccia.dev** semble être plus proche d'un logiciel de programmation que d'un logiciel de montage vidéo, pourtant j'ai décidé de m'inspirer de ce dernier pour élaborer l'ergonomie de mon interface. Tout simplement parce qu'à mon avis, cette approche décrit mieux les étapes de production d'un projet **seccia.dev** et tout compte fait, facilite sans aucun doute la mise en œuvre de votre jeu.

L'interface se présente donc en deux parties : la partie commune et la partie réservée aux pages. Dans la première partie, vous pouvez accéder au menu principal en haut à gauche, à la barre d'outils en haut à droite et à la navigation des pages en bas de la fenêtre. Ces zones sont fixes et omniprésentes, et partagent des fonctionnalités avec plusieurs éditeurs. La seconde partie est au contraire dynamique et elle est réservée au projet, au scénario, à la génération des exécutables, aux ressources pédagogiques et bien entendu aux éditeurs tels que l'éditeur de **scène** et de **rôle** que vous utiliserez le plus souvent.



Les éditeurs sont consacrés à la production d'assets, c'est-à-dire à la création et à l'édition d'**objet** (incluant les personnages), de **scène** (niveaux), de **rôle** (logique

gameplay), de **dialogue** (conversations entre les personnages), de **player** (incluant les inventaires), de **cinématique** (vidéo mp4) et d'**effet (shaders)**.

Menu principal

Le menu principal, situé en haut à gauche de la fenêtre, permet d'accéder aux fonctionnalités plus avancées du logiciel ou moins souvent sollicitées. Elles sont donc mises en retrait et regroupées par étape de production.

PROJECT	Création et gestion du projet.
SCENARIO	Graphe nodal, énigmes, gameplay...
TEXT	Dialogues, polices, palette de couleurs, localisation...
SCENE	Traitement par lots, exportation des aperçus...












Barre d'outils

La barre d'outils, située en haut à droite de la fenêtre, permet d'accéder rapidement aux fonctionnalités courantes du logiciel et communes aux éditeurs. Elle se présente sous la forme de trois groupes :



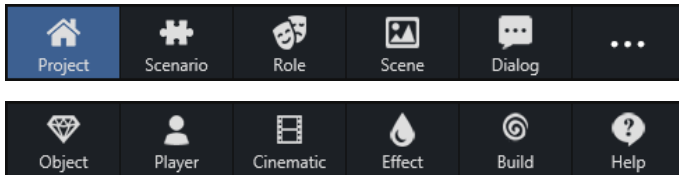
À l'exception du premier groupe, ces icônes restent accessibles quelle que soit la page sélectionnée, et pour certaines un raccourci clavier leur est attribué.

	Créer une nouvelle boîte (raccourci avec les touches P, S, D)
	Considérer une boîte comme résolue
	Prévisualiser l'ensemble du scénario pour vous déplacer
	Sélectionner la boîte START du scénario
	Sélectionner la boîte contenant une erreur
	Sauvegarder le projet en cours

	Afficher le dossier du projet dans l'explorateur de Windows
	Créer un nouvel asset (raccourci avec la touche ALT)
	Lancer la scène ouverte ou une scène spécifique en mode Live
	Lancer le jeu en mode Live
	Lancer le jeu en mode Debug
	Lancer le jeu en mode Release pour play_windows ou play_web
	Activer ou désactiver le rendu du jeu dans les éditeurs
	Activer ou désactiver le rendu temps-réel dans les éditeurs
	Activer ou désactiver le mode Assisted Scripting
	Accéder aux assets graphiques
	Accéder aux réglages du logiciel

Barre de navigation

La barre de navigation, située en bas de la fenêtre, permet d'accéder aux pages et d'effectuer des recherches dans le projet.



Les pages sont navigables également à l'aide d'un raccourci clavier.

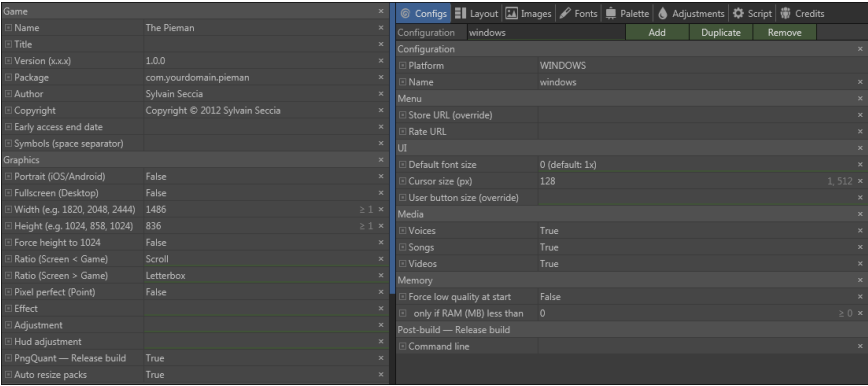
CTRL+Tab	Pour accéder à la page précédemment ouverte.
ALT+Left	Pour accéder à la page précédente de la liste.
ALT+Right	Pour accéder à la page suivante de la liste.

PROJET

Un projet **seccia.dev** est composé d'un fichier **JSON** centralisant les données du jeu, et en externe d'une arborescence de dossiers et de fichiers. Les fichiers sont principalement des textes et des données multimédias. Les principaux fichiers binaires produits par le logiciel sont en quelque sorte les données de compilation du jeu.

Le principal avantage de cette approche est d'apporter une meilleure flexibilité au projet via les outils d'édition et de versioning que vous emploieriez éventuellement. En revanche, il est important d'éviter l'édition manuelle du fichier portant l'extension **seccia** afin de préserver les liens des identifiants uniques. En d'autres termes, rien ne vous empêche de le modifier à votre guise à condition de maîtriser ce que vous faites.

seccia.dev utilise le format **PNG** pour stocker en fichiers externes toutes les images du jeu. Le format **JPEG** est toutefois accepté pour importer des images qui seront automatiquement converties.



Structure

Le projet contient un fichier définissant l'identité du projet et une liste de dossiers :

ASSET

Un sous-dossier par asset contenant tous les fichiers dépendants : textes, images, sons...

BINARIES	L'emplacement des builds générés et rangés par configuration.
IMAGE	Les images de l'interface du jeu. Ce dossier est aussi accessible depuis la page Project .
LIBRARY	Le dossier des interfaces permettant d'exporter des librairies et devant contenir un sous-dossier par interface.
PLATFORM	Les fichiers relatifs aux plateformes (principalement des icônes).
SCENARIO	Le scénario exporté en PNG faisant office de GDD et les sauvegardes liées aux chapitres.
SNAPSHOT	Aperçus des scènes .
SONG	Les musiques au format OGG pour Windows et au format MP3 pour les autres plateformes.
SOUND	Les bruitages au format WAV .
TEMP	L'emplacement des fichiers temporaires. En cas de suppression, seccia.dev devra régénérer ces fichiers.
TEXT	Les textes à traduire au format CSV et HTML , la police personnalisable et autres textes.

Chaque asset possède un identifiant unique attribué à sa création et les dossiers reprennent cet identifiant. Encore une fois, il est fortement déconseillé de renommer les dossiers par l'explorateur de **Windows**. Lorsque le nom d'un asset est modifié au sein du logiciel, tous les fichiers dépendants ainsi que les liens sont mis à jour automatiquement. Il va de même pour la suppression.

Il existe sept types d'asset : **Role**, **Scene**, **Object**, **Dialog**, **Player**, **Cinematic** et **Effect**. La première lettre de chaque type est reprise dans le nom des assets. C'est-à-dire qu'un asset de type **Scene** commence forcément par la lettre S, par exemple **S_HOME** et par déduction : R pour **Role**, O pour **Object**, D pour **Dialog**, C pour **Cinematic**, P pour **Player** et E pour **Effect**.

Concernant les images, vous pouvez à tout moment ouvrir la page **Project** pour visualiser la liste exhaustive des images de l'interface du jeu. Toutes les images sont regroupées dans le dossier **IMAGE**.

Paramètres

Tous les paramètres du projet sont regroupés et centralisés sur la page **Project**.

Game	
Name	Le nom du jeu, utilisé aussi pour nommer les fichiers.
Title	Le titre du jeu. Si le champ est vide, le titre sera le nom du jeu. Pour écrire en UTF-8 , préfixez-le par @ .
Version	Version du jeu sous la forme X.X.X (1.0.0).
Package	Nom du package de votre jeu (com.domain.name)
Author	Nom du développeur.
Copyright	Information sur les droits réservés (Copyright © Année Nom)
Early access end date	Il est possible de débloquent des actions par le code après une date prédéfinie via la fonction early_access .
Symbols	Permet de définir une liste de symboles (mots séparés par des espaces) à la génération du jeu et de tester la présence des symboles par le code via la fonction symbol .
Graphics	
Portrait	Uniquement disponible pour les mobiles, cette option permet de forcer l'orientation portrait au démarrage.
Fullscreen	Exclusivement pour les plateformes Desktop, cette option permet de forcer le mode plein écran au démarrage du jeu. Dans ce cas, le joueur ne pourra plus modifier la résolution dans les options.
Width & Height	La dimension du jeu en pixels. La taille maximale est 4096. Les scènes peuvent avoir une taille différente. Si votre jeu est au format 16/9, la résolution idéale est 1820x1024 (2 textures). Si votre jeu est au format 21/9, la résolution idéale est 2048x858 (2 textures). Dans ce dernier cas, il pourrait être préférable d'utiliser la résolution 2444x1024 (3 textures) si vous avez l'intention de zoomer fréquemment afin de garder une qualité optimale tout en optimisant la mémoire puisque les décors utilisent des textures de

	1024x1024. Si le ratio n'est pas un critère prioritaire et que vous souhaitez optimiser toute la surface des textures, vous pouvez choisir la résolution 2048x1024 (2 textures) correspondant à du 18/9 ou 2/1.
Force height to 1024	Permet de redimensionner toutes les images en 1024 pixels de hauteur si vous avez fait l'erreur de créer vos décors en 1080 par exemple. Cette option est particulièrement utile pour les mobiles.
Ratio (Screen < Game)	<p>Lorsque le ratio de la résolution de l'écran est plus petit que le ratio de la résolution de votre jeu, plusieurs choix s'offrent à vous pour compenser l'espace perdu verticalement :</p> <p>Scroll : La hauteur de la scène est égale à la hauteur de l'écran et un défilement horizontal est activé</p> <p>Letterbox : Des bandes noires sont ajoutées en haut et/ou en bas</p> <p>Crop : La scène est rognée en haut et/ou en bas</p>
Ratio (Screen > Game)	<p>Lorsque le ratio de la résolution de l'écran est plus grand que le ratio de la résolution de votre jeu, plusieurs choix s'offrent à vous pour compenser l'espace perdu horizontalement :</p> <p>Letterbox : Des bandes noires sont ajoutées à gauche et/ou à droite</p> <p>Crop : La scène est rognée à gauche et/ou à droite</p>
Pixel perfect	Si activé, Unity utilisera la valeur <code>FilterMode.Point</code> .
PngQuant	Cette option n'est valable que pour le build Release et permet d'encoder les PNG transparents en 8 bits au lieu de 32 bits grâce à la librairie externe PngQuant .
Auto resize packs	Si l'option est validée, seccia.dev parcourt toutes les scènes pour connaître la plus grande taille de l' objet . Si elle est inférieure à 100% de l' objet , les images sont redimensionnées pour optimiser le nombre de textures à générer. En cas de changement, il est nécessaire de régénérer les textures. Si vous avez l'intention de zoomer fréquemment dans vos scènes et ainsi garder une qualité optimale de vos textures, il est préférable de désactiver l'option et de vous assurer de la bonne dimension des fichiers PNG .

Filter	
Effect	Un effet peut être appliqué à la scène entière. Cette option est ignorée si elle est surdéfinie dans l'éditeur de scène .
Adjustment	L'ajustement est appliqué au moment de la génération du jeu. Cette option ne tient pas compte des HUD .
Hud adjustment	Cette option ne modifie que les images liées aux HUD .
Light	
Baked	Cette option permet d'optimiser le rendu en effectuant un prétraitement en cas de changement de lumières. Ce pré-rendu est alors stocké dans une autre texture. L'avantage de cette méthode est d'augmenter le framerate de votre jeu, l'inconvénient est d'utiliser plus de ressources graphiques. Si la scène nécessite de changer constamment l'état des lumières, il ne faut pas activer l'option. Je vous conseille de l'utiliser pour des lumières statiques ou qui changent occasionnellement.
Ambient	Cette valeur définit la lumière ambiante de la scène entre 0 et 255 . Si la valeur est à 0 , l'écran du jeu sera complètement noir. Si la valeur est à 255 , les pixels auront la couleur d'origine. Il n'est pas possible d'éclaircir davantage la scène par cette propriété.
Blur scale	Cette option permet d'appliquer un effet de flou sur la lumière pour adoucir le rayon. Plus la valeur est élevée et plus les performances seront impactées.
Low quality	Pour des gains de performance, activez cette option pour réduire la taille des textures réservées au rendu des lumières. Le résultat sera moins détaillé et plus pixelisé.
Bokeh	
Shape width/height	Cette option permet de définir la taille maximale de la forme d'un bokeh afin de reproduire l'effet d'une optique. Une optique sphérique produit des formes rondes alors que des objectifs anamorphiques utilisés dans le cinéma produisent des formes ovales. Les valeurs par défaut sont 128 pour la largeur et 256 pour la hauteur.

Menu	
Custom menu	Cette option permet de personnaliser l'interface en indiquant la scène qui deviendra l'écran d'accueil du menu. La fonction jump_menu permettra de naviguer d'un écran à un autre au sein du menu.
Effect	Un effet peut être appliqué au menu natif. Cette option ne concerne pas le menu personnalisé.
Splash durations	La durée d'attente en secondes. Si la valeur est à 0, le joueur doit cliquer ou toucher l'écran pour continuer.
Options menu	Affichage du menu Options .
Quality menu	Affichage du menu pour changer la qualité des textures.
Merge Audio/Text menu	Permet de différencier la langue des sous-titres et la langue audio.
Font size menu	Affichage du menu pour changer la taille de la police.
Subtitle menu	Affichage du menu Sous-titres pour changer le mode et la vitesse de défilement du texte.
Privacy policy URL	Lien qui renvoie vers votre page web contenant votre politique de confidentialité. Ce lien est obligatoire si vous utilisez les sauvegardes en ligne.
Store URL	Lien qui renvoie vers la page de vos jeux.
User button position	Position des boutons personnalisables.
User button size	Taille des boutons personnalisables du menu. À l'échelle 1, la taille est de 96 pixels.
User button URL	Lien qui renvoie vers une page web.
User button show	Par défaut tous les boutons sont visibles. Pour n'afficher que les trois premiers boutons, il suffit d'écrire « 123 ».
Text	
Font	Le nom de la police proposée par seccia.dev . Il est possible de fournir une police personnalisée.

CJK font	Le nom de la police asiatique utilisée pour la génération des textures. La police doit être installée sur l'ordinateur au moment de la génération. Si le champ est vide ou si la police n'a pas pu être chargée, le logiciel tentera d'utiliser une police par défaut.
Font color	La couleur du texte par défaut.
Font color (Highlight)	La couleur du texte lorsque le joueur pointe sur un choix de réplique lors d'une conversation.
Font color (Door)	La couleur du texte lorsque le joueur pointe sur une door .
Font color (Menu)	La couleur du texte des menus.
Font color (Menu/Rate)	La couleur du texte du menu Evaluate the game .
Font color (Menu/Highlight)	La couleur du texte lorsque le joueur pointe sur un menu.
Font color (Menu/Values)	La couleur du texte indiquant les valeurs des options.
Font color (Credits)	La couleur du texte du générique.
Font color (Credits/Title)	La couleur du texte du générique lorsque la ligne commence par un arobase.
Back color	Affichage d'un rectangle derrière le texte pour faciliter la lecture. La couleur est au format RGBA en hexadécimal.
Default subtitles speed	Définit le mode et la vitesse par défaut.
Subtitles margin	Définit la marge entre le texte et le bas de l'écran.
Typewriter	Si cette option est activée, les dialogues s'écritont à l'écran lettre par lettre selon la vitesse des options du jeu.
Vibration	Permet d'animer le texte avec un effet d'ondulation.

Randomize dialogs	Permet de mélanger l'ordre des choix des répliques de toutes les conversations au lancement d'une nouvelle partie.
Balloon: image	Les marges intérieures en pixels séparées par une virgule pour définir la zone variable de l'image lors de l'ajustement de la bulle par rapport au texte dans l'ordre suivant : gauche, haut, droite et bas. Les valeurs doivent être positives. Il est nécessaire d'inclure une image ITEM_BALLOON de 128x128 pixels pour utiliser le mode COMICS .
Balloon: text	Les marges intérieures en pixels séparées par une virgule pour définir la zone de texte dans l'ordre suivant : gauche, haut, droite et bas. Les valeurs doivent être positives.
Choice: size	Taille en pixels de la bulle où sera affiché le texte du choix proposé au joueur. Il est nécessaire d'inclure une image ITEM_CHOICE de 256x256 pixels pour utiliser le mode VISUAL NOVEL .
Choice: icon size	Taille en pixels de l'icône.
Choice: horz space	Espace en pixels entre deux bulles.
Choice: vert space	Espace en pixels entre les bulles et le bord de l'écran.
Choice: margin	Les marges intérieures en pixels séparées par une virgule pour définir la zone de texte dans l'ordre suivant : gauche, haut, droite et bas. Les valeurs doivent être positives.
Audio	
Level menu	Permet de fusionner le volume des sources audio en un seul menu ou de désactiver le menu.
Preload sounds	Permet de charger en mémoire tous les sons (fichiers WAV) de votre jeu au lancement de l'application.
Savegame	
Menu	Permet d'activer ou de désactiver la gestion des sauvegardes manuelles.

Server URL	Le lien du dossier contenant les scripts seccia.dev permettant de gérer les sauvegardes en ligne via votre propre site. Si le champ est vide, le menu de la sauvegarde en ligne sera inaccessible.
Server game	Nom du jeu et par conséquent du sous-dossier à créer sur votre serveur.
Version	La version de la sauvegarde du jeu. Si deux versions du jeu sont trop différentes, il est préférable d'incrémenter cette valeur. Les anciennes sauvegardes ne seront alors plus compatibles mais cela évitera de faire planter l'application ou d'obtenir des résultats hasardeux. Si la valeur est à 0 , les sauvegardes ne seront jamais compatibles entre deux versions.
Autosave	Permet de lancer une sauvegarde automatique à chaque fois qu'une boîte puzzle est résolue.
Languages	
Native	La langue par défaut utilisée dans l'éditeur. C'est généralement la langue native du développeur ou du jeu.
Default	La langue par défaut au premier lancement du jeu.
OS if possible	Si l'option est activée, l'application tentera de récupérer la langue de l'appareil. En cas d'échec, la langue par défaut sera prise en compte.
Language	Avant de pouvoir utiliser une langue, il est important de l'activer pour avoir accès aux champs. Lorsque vous désactivez une langue, les textes ne sont pas effacés. Utilisez le menu TEXT/Purge unused languages pour effectuer un nettoyage complet (irréversible).
Post-build command lines	
Zip	Il est possible de compresser l'ensemble des builds à la fin de la génération en activant cette option.

Paramètres des configurations

Vous pouvez créer des configurations de build ayant des paramètres différents en fonction de la plateforme et de la version à distribuer. Par défaut, il existe

deux configurations nécessaires aux tests au sein du logiciel : **play_windows** et **play_web**. Les plateformes possèdent leurs propres paramètres.

Paramètres communs à toutes les plateformes	
Store URL	Lien qui renvoie vers la page de vos jeux. Si le champ est utilisé, le lien équivalent du groupe Menu est ignoré.
Rate URL	Lien qui renvoie vers la page du store permettant d'évaluer le jeu.
Default font size	La taille de la police au premier lancement de l'application. Si la valeur est à 0, une valeur par défaut est utilisée.
Cursor size	La taille du curseur en pixels.
User button size	La taille des boutons personnalisables du menu. Si le champ est utilisé, le lien équivalent du groupe Menu est ignoré.
Media	Permet d'inclure les voix, musiques et vidéos.
Force low quality	Active le mode en basse qualité lors du premier lancement de l'application si le menu Quality est accessible. Par défaut l'application se base sur les performances de l'appareil pour activer le mode.
Command line	Il est possible d'exécuter un programme en ligne de commande à la fin de la génération. Cette fonctionnalité ne s'applique qu'au build Release .
Paramètres spécifiques à Android	
Package	Nom du package spécifique à la configuration. Si le champ est vide, le package du groupe Game est utilisé.
Version code	La version du fichier APK ou AAB sous forme d'un entier commençant par 1.
Bundle	Par défaut, un fichier APK est généré. Pour générer un fichier AAB afin de publier sur le store Google Play , il est nécessaire d'activer l'option.
File	Le fichier Keystore généré par le SDK Android pour signer le fichier APK ou AAB .

Store password	Le mot de passe qui a été utilisé pour générer le fichier.
Alias	Le nom qui a été utilisé pour générer le fichier.
Password	Le mot de passe qui a été utilisé pour générer le fichier. Si les deux mots de passe ont identiques, vous devez compléter les deux champs.
Paramètres spécifiques à WebGL	
Distribution	Permet de spécifier le type de distribution. Choisissez Local pour tester votre jeu au sein du logiciel.
URL	Lien du dossier Content si les assets sont accessibles depuis un autre emplacement.
Check domain	Permet de vérifier la validité du nom de domaine du site où est hébergée l'application. Si l'option est activée, il est nécessaire de fournir une liste de domaines autorisés.
Allowed domains	La liste des noms de domaine acceptés. L'application ne se lancera pas si le site qui héberge le jeu n'est pas autorisé.
Width & Height	La résolution du jeu à l'intérieur de la page web.

Crédits

Les crédits défilent à la fin du jeu et sont également visibles depuis le menu principal ou lors de l'appel de la fonction **show_credits**. Chaque langue possède son propre texte.

La couleur du texte est définie par la propriété **Font color (Credits)**. Pour changer la couleur d'une ligne entière, il suffit d'ajouter le caractère **@** en début de ligne. Cette couleur est personnalisable via la propriété **Font color (Credits/Title)**.

Vous avez aussi la possibilité d'inclure des objets animés sur une ligne en écrivant le nom de l'objet précédé par le caractère arobase. Seule l'animation **STOP** sera jouée.

Vous pouvez ajouter des mots clés prédéfinis pour récupérer certaines propriétés de votre projet. Pour cela **seccia.dev** propose une liste de constantes accessibles depuis les **dialogues**, les champs textes et les variables de vos scripts.

{OS}	La plateforme du build actuel : windows, web, android, ios
{LANG}	La langue sélectionnée par le joueur : French, English, German, Spanish, Italian, SimplifiedChinese...
{NAME}	Le nom du jeu sans espace ni caractères spéciaux
{TITLE}	Le titre du jeu
{VERSION}	La version du jeu
{AUTHOR}	Le nom du développeur
{COPYRIGHT}	Le champ Copyright renseigné dans les propriétés

Terminal

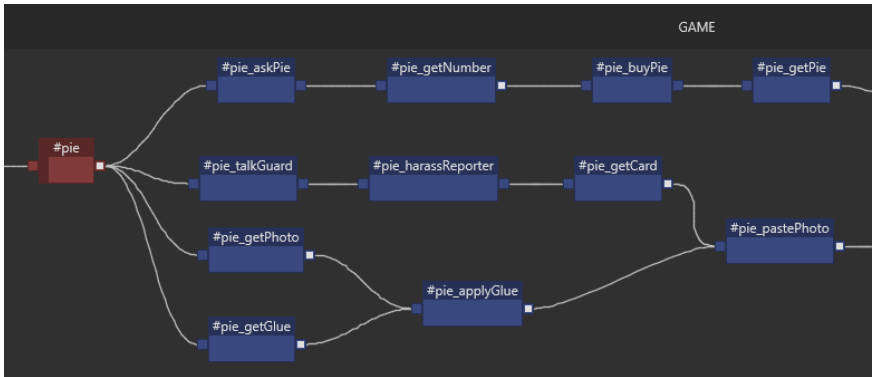
Le terminal, accessible depuis le menu principal **FILE**, vous permet d'appliquer un ensemble de modifications à votre projet via une interface en ligne de commande. Vous pouvez facilement effectuer des traitements par lot en insérant une liste de commandes dans le champ texte. Toutes modifications seront définitives. N'hésitez pas à dupliquer votre projet avant d'accéder au terminal si vous procédez par tâtonnement.

```
object_copyFrame PNG ASSET ANIM DIR index
object_setTitle ASSET "title" LANGUAGE {...}
object_update ASSET {...}
project_deleteAsset ASSET {...}
project_newAsset ASSET {...}
project_renameAsset OLDASSET NEWASSET {...}
```


SCÉNARIO

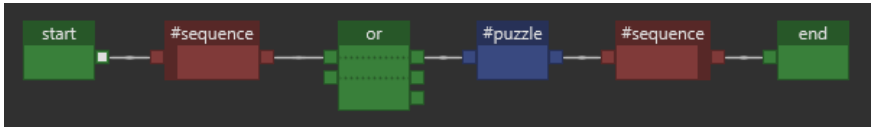
L'éditeur de scénario est au cœur de l'environnement, permettant de définir le déroulement narratif du jeu. Il n'est donc pas nécessaire de programmer une machine à état pour contrôler les énigmes puisque **seccia.dev** propose déjà un outil prêt à l'emploi. Il suffit d'ajouter des boîtes nodales au scénario et de les connecter entre elles. Au runtime, un interpréteur se charge d'actualiser les états lorsque le créateur du jeu décide de valider une énigme à la suite d'une action du joueur. En d'autres termes, le créateur n'a besoin que de notifier la validation d'une boîte lorsque le joueur résout l'énigme en question. Cela dit, l'éditeur de scénario n'est qu'une représentation abstraite de vos énigmes qui permet surtout de structurer la narration et de faciliter le travail en amont.

Les scripts s'intègrent aux boîtes nodales du scénario, des rôles et des timelines. Vous trouverez la liste exhaustive des fonctions du langage en cliquant sur le bouton de la page d'aide en bas de la fenêtre.



Boîte nodule

Il existe seulement cinq types de boîtes nodales : **puzzle**, **sequence**, **or**, **start** et **end**. Chaque boîte possède des propriétés spécifiques à retenir. Pour ajouter une boîte au scénario, il suffit soit de cliquer sur le bouton **[+]** de la barre d'outils principale pour faire apparaître la liste des types, soit d'utiliser les touches du clavier : **P**, **S**, **O** ou **E**. Vous l'aurez certainement remarqué, la touche représente la première lettre du type. Seule la boîte **start** est unique. Les noms que vous donnerez aux boîtes **puzzle** et **sequence** devront toujours commencer par le caractère **#**.



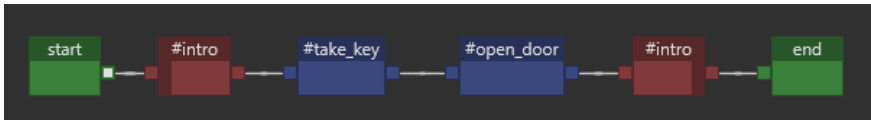
Les boîtes **puzzle** et **sequence** ont deux connecteurs : un en entrée et un en sortie. La boîte **start** ne possède qu’une sortie et la boîte **end** qu’une entrée. En revanche la boîte **or** est légèrement plus complexe avec ses deux entrées et trois sorties mais ne vous inquiétez pas, nous y reviendrons. Un connecteur d’entrée peut recevoir plusieurs connexions et un connecteur de sortie peut aussi transmettre plusieurs signaux. De plus, les boîtes peuvent posséder un script entrant et un script sortant comme une machine à état classique à l’exception du script exécuté en boucle qui n’existe pas. Avant d’étudier les branchements, il est important de décrire les particularités des cinq types évoqués et de bien comprendre que la progression ne peut se dérouler que de gauche à droite, autrement dit du passé au futur sans la possibilité de revenir en arrière.



Les boîtes **start** et **end** sont obligatoires. Elles ne peuvent être retirées (il doit rester au moins une boîte **end**). Lorsqu’une nouvelle partie est lancée depuis le menu du jeu, la première boîte exécutée est **start** comme son nom l’indique. Par définition, il ne peut y avoir de boîtes situées avant **start** et après **end**. Toutes les boîtes **end** mèneront vers les crédits et termineront définitivement la partie. Distinguez tout de même la persistance de données des parties et du jeu. Des valeurs peuvent coexister entre toutes les parties, vous donnant la possibilité d’étendre la rejouabilité de votre jeu en incitant le joueur à atteindre par exemple toutes les fins possibles.



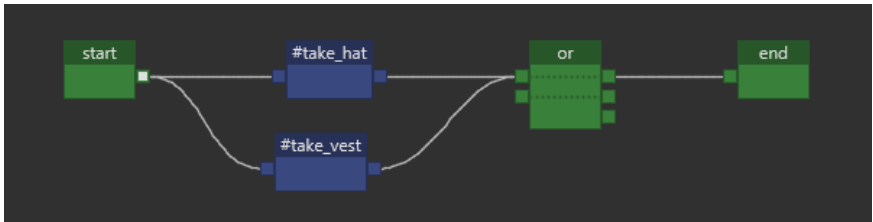
La boîte **puzzle** est la plus courante car elle sert à définir les actions du joueur. Il faut imaginer la boîte bleue comme une mini boucle infinie où la seule façon d’en sortir est de valider l’énigme. Il existe plusieurs manières de le faire : par le code via la fonction **success**, par l’éditeur de **rôle** via l’action **success** ou par les propriétés de certains éditeurs.



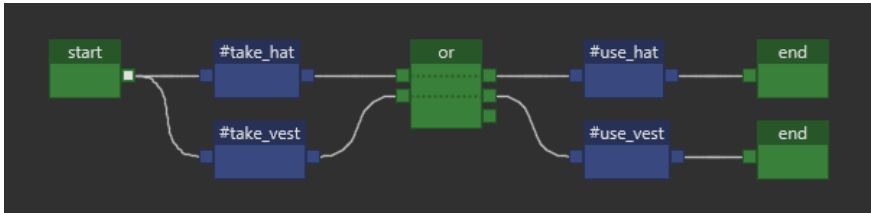
Contrairement aux **puzzles**, les **séquences** ne sont pas bloquantes et le script sortant est donc exécuté immédiatement sans l'interaction du joueur. L'intérêt de la boîte **séquence** est d'améliorer la structure des énigmes pour gagner en flexibilité et en lisibilité, et ainsi de mieux séquencer le scénario. Cela fait partie des bonnes pratiques. Plus intéressant encore, une séquence définit automatiquement une plage d'énigmes et il suffit ensuite d'appeler les fonctions **started**, **unstarted**, **ended** et **playing** pour savoir si le joueur se trouve à l'intérieur ou si la plage entière a déjà été résolue.

```
if playing #intro
  // code here
end
```

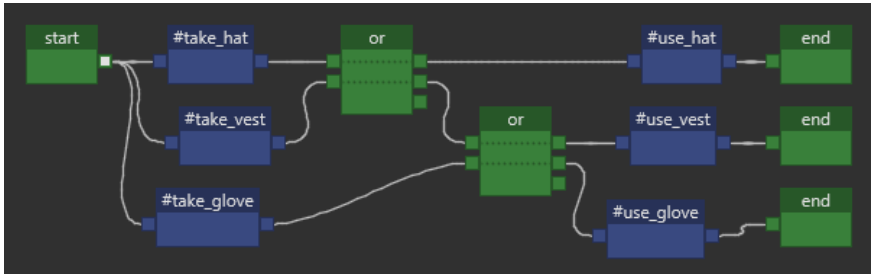
Il est bien évidemment possible d'imbriquer des plages à condition de respecter la notion de **PEPS** (premier entré premier sorti). Cette particularité est avant tout employée pour contraindre l'évolution de la narration. À priori le joueur ne devrait pas pouvoir résoudre des énigmes situées à l'extérieur d'une plage tant qu'il n'en est pas sorti. Si cette contrainte ne vous satisfait pas, cela signifie que le concept des plages n'est finalement pas adapté à vos besoins. Pour illustrer ce cas de figure, prenons un exemple concret très simple : le joueur se retrouve dans un guet-apens qui le limiterait momentanément dans ses déplacements tant qu'il n'aurait pas réussi à s'échapper.



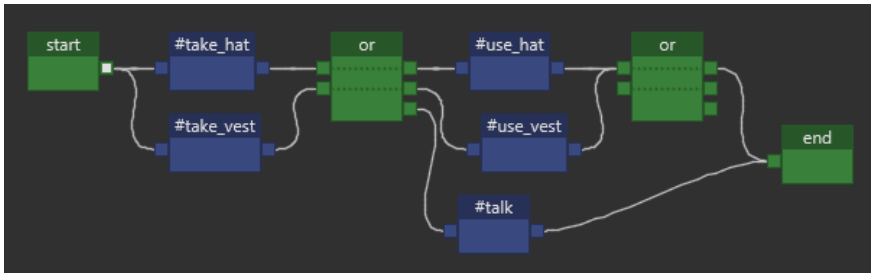
Enfin parmi les cinq types, la boîte **or** permet de proposer au joueur de résoudre une énigme dans une liste prédéfinie en produisant éventuellement un impact sur l'histoire. Dans cette configuration, le joueur n'est plus censé résoudre toutes les énigmes qui lui sont proposées pour passer à l'étape suivante. Ainsi, la première énigme qui mènera à la boîte **or** deviendra l'énigme sélectionnée et invalidera les autres. Depuis l'éditeur, vous pouvez invalider des énigmes en les marquant comme **lost** afin de simuler une partie sans devoir tout rejouer depuis le début. Pour utiliser la boîte de cette manière, il suffit de connecter vos **puzzles** sur la première entrée et de faire sortir le signal via la première sortie comme le montre le schéma ci-dessus. Gardez en tête que la condition s'effectue toujours à l'entrée et jamais à la sortie.



La boîte **or** permet également de faire évoluer la narration en fonction des choix du joueur grâce au concept de branches. Pour l'utiliser de cette manière, vous devez vous servir des deux premiers circuits. Le premier circuit relie la première entrée avec la première sortie, et le deuxième circuit relie la deuxième entrée avec la deuxième sortie. Ce sont des circuits distincts : si le signal entre par un circuit, il sortira forcément par le même circuit. Le premier signal entré définira donc le circuit de sortie. C'est fort pratique pour deux branches mais qu'en est-il avec trois branches puisque la boîte ne propose que deux circuits ? Jetez un coup d'œil à l'image ci-dessous.



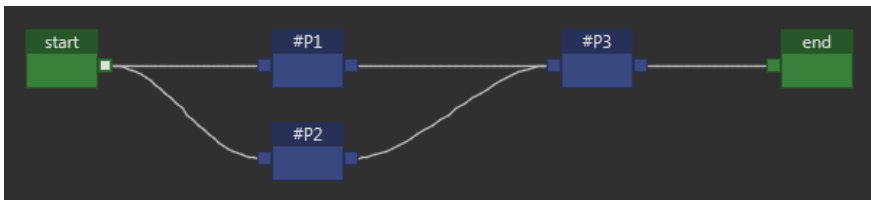
Grâce à cette astuce, vous pourrez définir autant de branches que vous le souhaitez. Il s'agit d'imbriquer les boîtes **or** pour ajouter des branches supplémentaires.



Ce n'est pas tout, vous avez sûrement remarqué la présence d'une troisième sortie qui en fait représente un troisième circuit sans entrée. Quelle que soit l'entrée choisie, le signal sortira par ce troisième circuit. Il ne s'agit pas d'un catch-all mais bien d'une sortie obligatoire si un câble est connecté.



Pour résumer le fonctionnement des boîtes nodales, selon le schéma ci-dessus au lancement d'une nouvelle partie, c'est d'abord le script sortant **start** qui est exécuté, puis dans la même frame de l'application, c'est au tour du script entrant **puzzle** d'être exécuté. Comme la particularité des boîtes **puzzle** est de bloquer la progression du scénario, l'interpréteur attend la notification du créateur pour en sortir comme nous l'avons évoqué précédemment. C'est donc seulement au moment de la validation de l'énigme par le déclenchement d'une action provoquée par le joueur, que le script sortant **puzzle** est exécuté, suivi du script entrant **end**.



Hormis **or**, lorsqu'une boîte reçoit deux connexions entrantes, le comportement est légèrement différent. Le script entrant de **#P3** n'est exécuté qu'après la validation des boîtes **#P1** et **#P2**, c'est-à-dire après que les scripts sortants sont exécutés. Pour mieux le comprendre, imaginez que le signal sortant de **#P1** se retrouve en attente devant l'entrée de **#P3** tant que les autres signaux ne sont pas tous arrivés.

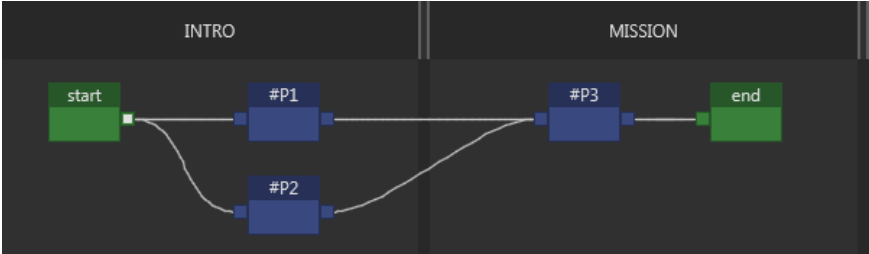
Puisque le scénario représente une vision globale de votre projet, ces scripts sont supposés changer exclusivement l'état général de la partie. Il est déconseillé de coder des actions qui ne feraient pas avancer la narration. Cette approche vous permettra de simuler une partie en cochant les boîtes résolues afin de tester plus facilement votre jeu à un moment précis de l'histoire.

Pour cela, créez d'abord une énigme **#take_key** (boîte bleue) et ajoutez l'instruction suivante dans le script sortant :

```
take P_HERO O_KEY
```

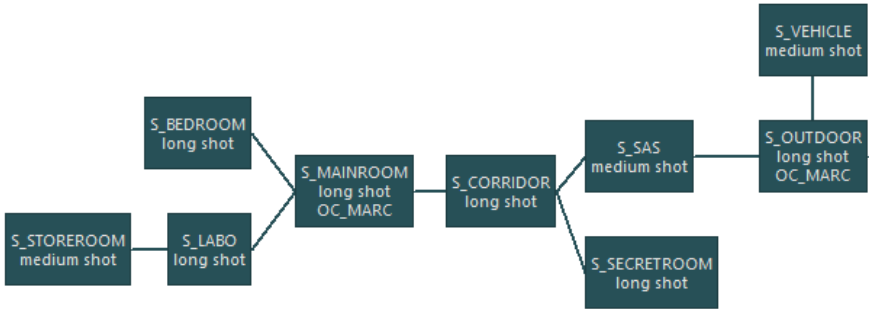
Il suffira de valider l'énigme ailleurs dans votre application, soit par le code soit par les éditeurs, pour placer l'objet dans l'inventaire. Ainsi, en cas de simulation de partie, l'objet sera déjà présent dans l'inventaire car les scripts du scénario seront exécutés au lancement sans interaction du joueur.

Section



Les sections sont optionnelles et permettent de mieux structurer le scénario chronologiquement. Elles sont représentées sous forme de colonnes pouvant contenir différents types de données. Pour ajouter une nouvelle section, il suffit de faire un clic droit sur le fond ou un titre de section et de sélectionner **Add Section**. Un double clic sur le titre permet d’éditer la section. Maintenez la touche **Shift** enfoncée pour éviter de déplacer automatiquement les boîtes lorsque vous redimensionnez une section.

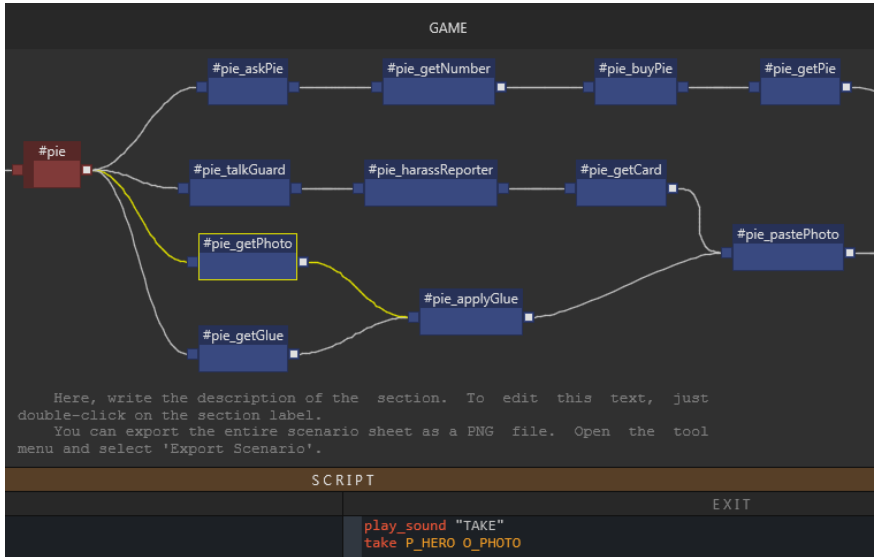
Un texte descriptif peut être affiché en bas de la section. Ce texte peut contenir plusieurs paragraphes et n’est pas limité en nombre de caractères. Les noms d’asset sont reconnus dans la description et mis à jour automatiquement en cas de changement de nom.



Pour documenter le plan des lieux à explorer par le joueur, un mini outil de diagramme permet d’ajouter des boîtes facilement et de les connecter entre elles pour désigner les accès aux **scènes**. Cliquez sur le coin en bas à droite de la boîte pour ajouter une connexion. Cet outil peut bien sûr être utilisé pour présenter d’autres formes de données si besoin. Gardez en tête qu’il ne s’agit que d’un moyen d’exposer et de partager des notes, en aucun cas ces relations seront appliquées par le logiciel.

Script

Les boîtes nodales du scénario possèdent au moins un script entrant ou un script sortant à l'exception des **puzzles** qui possèdent les deux. Vous pouvez éditer ces scripts en sélectionnant une boîte pour faire apparaître l'éditeur de code juste en dessous du graphe.



Vous n'avez pas à vous soucier d'appeler les scripts, l'interpréteur s'en charge pour vous. Vous n'avez besoin que de valider les boîtes **puzzle** avec par exemple avec la fonction **success** en écrivant cette instruction :

```
success #1_intro_takeKey
```

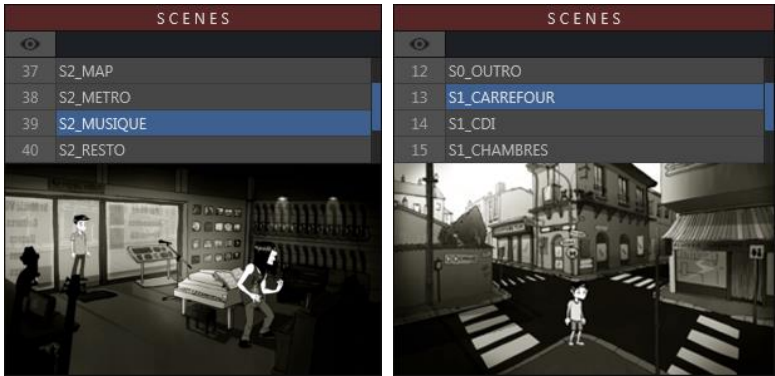
Exportation

Si plusieurs personnes travaillent sur le même projet, il peut s'avérer pratique de partager les informations relatives au scénario au sein de l'équipe. Le but de **seccia.dev** est d'encourager l'écriture du scénario directement depuis le logiciel sans passer par un outil intermédiaire, tel qu'un traitement de texte car cela permet de limiter les étapes de production. Il est donc possible d'exporter le scénario complet en un simple fichier **PNG** via le menu **SCENARIO/Export scenario**. Lorsque les boîtes nodales et les sections ont bien été définies, la capture devrait donner toutes les indications relatives au

scénario. Les commentaires propres aux boîtes sont exportés dans un fichier **HTML**.

Les bonnes pratiques

Étant donné le nombre important d'énigmes à prévoir dans un jeu et par conséquent le nombre de boîtes à mettre en place dans le scénario du projet, il est préférable de préfixer leur nom en se référant au numéro du chapitre. Si le jeu ne comporte qu'un seul chapitre, il serait alors judicieux de scinder la narration en actes pour le développement. Ce n'est pas une règle obligatoire, à vous de trancher selon votre histoire afin d'adopter la méthode la plus adaptée.

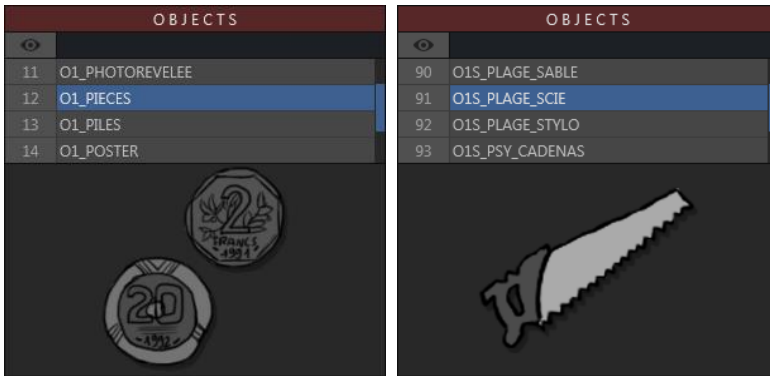


Concernant les assets, la nomenclature est très simple à une exception près pour les **objets**. Le numéro de chapitre peut aussi être repris dans les noms d'asset.

S_HOME S_LIVINGROOM P_HERO C_INTRO E_NIGHT

La première lettre, comme nous l'avons vu précédemment, est imposée et permet de définir son type. Un mot court et précis suffit pour décrire l'asset. Si vous optez pour la numérotation des chapitres :

S1_HOME S1_LIVINGROOM P1_HERO C1_INTRO E1_NIGHT



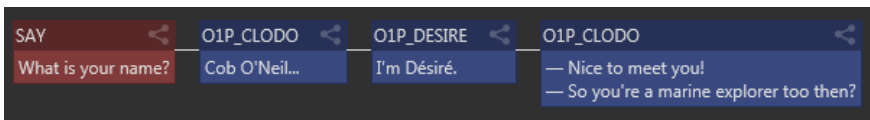
Concernant l'exception, une lettre supplémentaire permet de distinguer les différents types d'**objets**.

OC_HERO OI_KEY OS_DOOR

Ou bien alors :

O1C_HERO O1I_KEY O1S_DOOR

La lettre **C** indique qu'il s'agit d'un personnage, alors que la lettre **I** indique qu'il s'agit d'un **objet** à placer dans l'inventaire et aura donc besoin d'une ou plusieurs icônes. La lettre **S** est réservée aux **objets** présents dans les **scènes** qui devront interagir avec le joueur. Les **objets** statiques d'une **scène** n'ayant aucune interaction ou n'ayant qu'une fonction décorative pour meubler le lieu ou gérer la profondeur (perspective) seront à implémenter parmi les **stills**. Encore une fois, il n'est question que de bonnes pratiques.



Il y a généralement deux types de dialogues : les conversations et les répliques contextuelles. Pour mieux les différencier il est préférable de reprendre le nom du personnage ou du lieu.

D_OLDMAN D_HOUSE D_HERO

Pour les **rôles**, le logiciel a besoin d'associer une **scène** avec un **rôle** pour éditer les liens et les synchroniser durant la partie. Sur le même principe, optez pour un nom identique à la **scène** ou au **player**.

R_HOUSE R_GARDEN R_HERO

Côté énigmes, le titre des boîtes est capital pour s'y retrouver rapidement et optimiser par la suite le travail de débogage. Je vous conseille vivement d'utiliser la syntaxe suivante pour les boîtes bleues :

chapter_sequence_label

Le chapitre permet de découper le scénario en plusieurs parties.

La séquence est un mot court unique représentant une situation, par exemple : **intro**, **evasion**, **traque**, **combat**, **negociation**...

Le label permet de préciser le contexte.

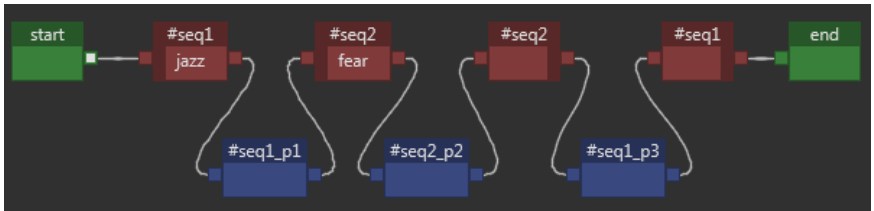
Si nécessaire, ajoutez une note à votre boîte pour enrichir le contexte. Pour une utilisation d'**objets**, il s'agirait du nom des **objets** concernés :

O_KEY + O_DOOR ou **O_WATER + O_BOTTLE = O_FULLBOTTLE**

Il est important d'appliquer des plages pour connaître facilement et à tout moment la progression du joueur via les fonctions liées aux séquences comme **started** ou **ended**. Pensez à reprendre le nom de la séquence dans chaque énigme. Un outil via le menu popup permet de renommer rapidement plusieurs boîtes après les avoir sélectionnées.

Le texte du corps de la boîte séquence entrante pourrait indiquer l'ambiance musicale de la séquence. À vous de définir au préalable une liste d'ambiance propre au jeu (*love, joy, sadness, fear, anger, shame, intrigue*...). Une information très utile pour les artistes et particulièrement pour le compositeur qui seront amenés à explorer l'univers du jeu.

Les séquences peuvent s'imbriquer et c'est d'ailleurs un des principaux intérêts des plages. Couplé avec l'indication de l'ambiance musicale, il devient commode d'enchaîner des musiques avec fondu d'une séquence à une autre. Dans l'exemple ci-dessous, la musique de la séquence 2 s'enchaîne sur la musique de la séquence 1.



Pour cela, il suffit d'interagir avec un rôle aux endroits indiqués ci-dessous :

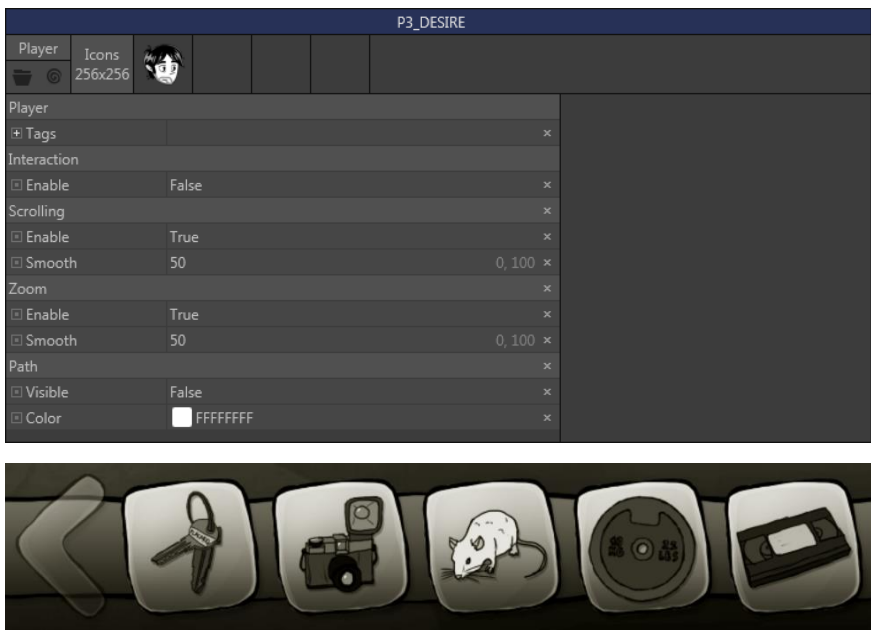


- 1) jouer le morceau "SONG1"
- 2) jouer le morceau "SONG2" en fondu enchaîné
- 3) jouer le morceau precedent en fondu enchaîné

PLAYER

L'éditeur de **player** permet à la fois de contrôler le personnage incarné par le joueur et de gérer son inventaire. Il est tout à fait possible d'avoir plusieurs inventaires en changeant de personnage via le code ou l'interface du jeu.

La caméra est associée au **player** courant et peut être paramétrée directement depuis l'éditeur.



Icône

Un **player** possède jusqu'à quatre icônes nommées et numérotées de **0.png** à **3.png**. L'icône par défaut est la première de la liste et elle est requise pour pouvoir afficher la tête du personnage dans la barre d'inventaire. La fonction **set_player_icon** permet de spécifier l'index de l'icône courante. Cela dit, si le jeu ne permet pas d'incarner plusieurs personnages, alors aucune icône n'est finalement nécessaire.

Les icônes sont des fichiers **PNG** et doivent être placés dans le dossier **ICON** de l'asset. Vous pouvez également cliquer sur l'aperçu de l'icône pour remplacer le fichier par un autre.

La taille optimale de l'image est de **256x256** pixels en 32 bits. La texture sera chargée en mémoire sur la carte graphique uniquement si l'icône est présente dans la barre d'inventaire.

Défilement & Zoom

Il est possible d'activer ou désactiver le défilement automatique de la **scène** par rapport à la position horizontale du personnage en cliquant sur la case à cocher **Scrolling** ou en appelant les fonctions **enable_scrolling** et **disable_scrolling**.

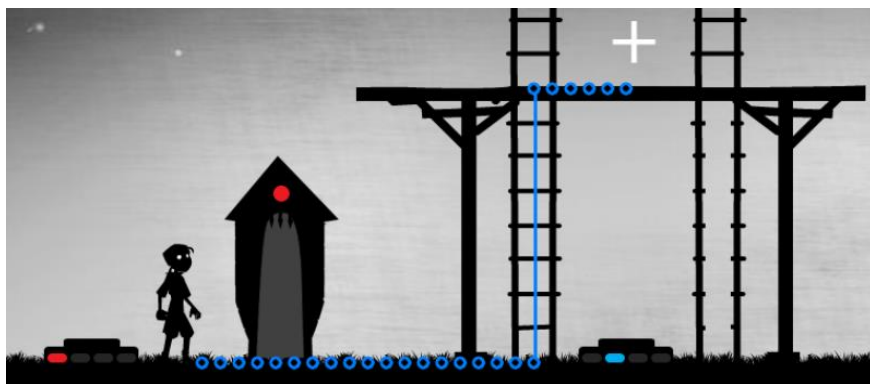
Le défilement peut être affiné par un effet **smooth**, permettant d'appliquer une décélération progressive au lieu d'un arrêt net et brusque.

Il va de même pour le zoom de la caméra via la case à cocher **Zoom** et les fonctions **enable_zoom** et **disable_zoom**.

Il existe d'autres fonctions pour contrôler la caméra telles que **set_shake** et **set_wave**. Ces fonctions sont documentées dans la page d'aide du logiciel.

Trajectoire

Par défaut, la trajectoire empruntée par le personnage lors d'un déplacement ne s'affiche jamais à l'écran. Cette option permet alors de la rendre visible et de choisir sa couleur.



Notez que l'image de la pastille utilisée pour dessiner la trajectoire se trouve à l'emplacement `IMAGE\WAY.png`. Rappelez-vous que vous pouvez vous rendre à la page **Project** pour visualiser la liste exhaustive des images de l'interface du jeu.

Intégration par le code

L'utilisation des **players** par le code est un jeu d'enfant. La première étape consiste à associer un **player** à un personnage (**objet**) via la fonction **control**. Un **player** ne peut contrôler qu'un seul **objet** à la fois. Quel que soit l'**objet** lié, l'inventaire est propre au **player**.

```
control P_HERO OC_HERO
```

La seconde étape consiste à sélectionner le **player** courant, c'est-à-dire celui qui est actif, grâce à la fonction **switch**.

```
switch P_HERO
```

C'est tout.

En cas d'inventaire multiple, il faut d'abord définir la liste des **players** à intégrer à la barre d'inventaire. Le **player** en cours n'est jamais affiché.

```
set_player_list P_HERO P_FRIEND
```

Il ne faut pas oublier de définir la **scène** initiale de chaque **player**.

```
set_player_scene P_HERO S_HOUSE
```

Pour changer l'icône d'un **player** :

```
set_player_icon P_HERO 1
```


OBJET

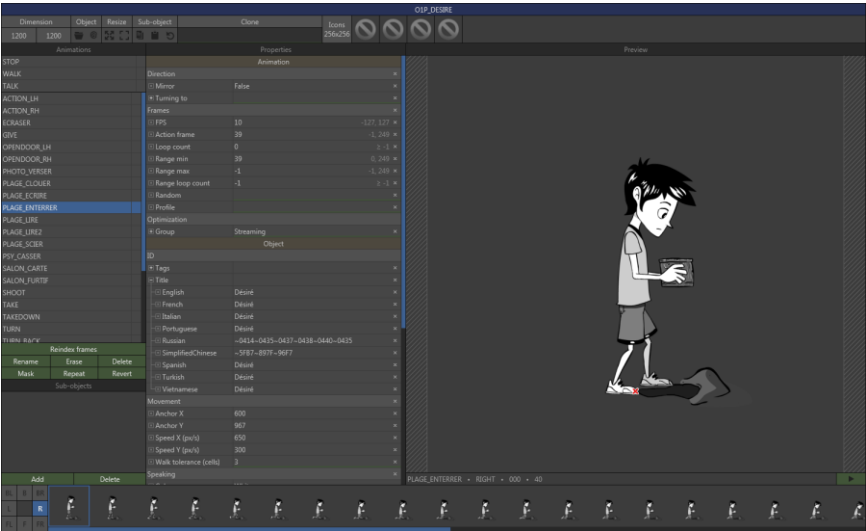
L'éditeur d'**objets** permet de créer des objets au sens large du terme et de les paramétrer pour ensuite les manier dans les **scènes**. Le terme générique **objet** est donc employé pour mentionner à la fois les **objets** actifs et les **objets** statiques de la **scène**, les items de l'inventaire et les personnages du jeu.

La première colonne de l'interface liste les trois animations par défaut **STOP**, **WALK** et **TALK** même si elles sont vides, et juste en dessous le nom de vos animations. Chaque animation possède ses propres paramètres.

La seconde colonne expose par catégorie les propriétés de l'**objet**, des animations et des **sous-objets**.

La troisième colonne est réservée à la visualisation de la **frame** sélectionnée.

Les **frames** de l'animation sélectionnée sont affichées en bas de l'éditeur.



Objet

Dans la seconde colonne réservée aux propriétés, vous trouverez la liste exhaustive des paramètres relatifs à l'**objet** détaillés dans le tableau suivant :

ID	
Title	Le titre est affiché à l'écran lorsque le joueur interagit avec l' objet . Par exemple lorsque le curseur de la souris survole la zone d'interaction de l' objet . Si le champ est vide, l'interaction sera limitée aux sélections. Le séparateur <i>pipe</i> permet de définir plusieurs titres pour le même objet . Par défaut seul le premier titre de la liste sera actif mais grâce à la fonction set_title , vous pourrez choisir un autre titre en spécifiant son index commençant par 0.
Movement	
Anchor X/Y	Ce point, marqué par une croix rouge, est utilisé pour gérer la profondeur de la scène , c'est-à-dire l'ordre d'affichage des objets . Car compte tenu de l'environnement de la scène en deux dimensions seulement, la coordonnée Y est interprétée comme une coordonnée Z : plus cette valeur est petite et plus l' objet sera éloigné de la caméra. Ce point représente la base de l' objet en contact avec le sol ou un support. Par exemple pour un personnage ce seront les pieds alors que pour une statue, ce sera le socle. Si l' objet est posé sur une table, une autre propriété est prévue dans la scène pour définir la hauteur de l' objet par rapport au sol. Ce cas de figure est pertinent seulement si le personnage doit se déplacer à la fois derrière et devant la statue. Les valeurs sont en pixels et relatives à l'image PNG . Vous pouvez soit entrer les valeurs soit double-cliquer sur la prévisualisation. Enfin si votre objet est censé changer de direction, la position X doit impérativement se trouver au centre de l'image pour éviter un décalage de transition. Je vous conseille donc de centrer vos personnages au niveau de la colonne vertébrale et d'ajuster si besoin en effectuant des tests. La fonction set_anchor permet de changer les valeurs par le code. Les rotations se basent également sur ce point pour définir le centre.
Speed X/Y	Il s'agit de la vitesse de la marche en pixels par seconde. Notez que ces valeurs peuvent être surdéfinies dans l'éditeur de scène si vos perspectives diffèrent entre deux scènes .
Walk tolerance	Lorsque le joueur clique sur une partie de votre décor, le personnage se déplace automatiquement vers ce

	point en parcourant le plus court trajet avec le moins de changements possible de direction. Si la destination est trop proche de la position actuelle du personnage, le déplacement sera perçu comme un saut rapide où l'animation de la marche sera sèchement interrompue. Pour éviter cet effet graphique désagréable, il suffit de définir la distance minimale autorisée en nombre de cells pour déclencher un déplacement. Cette contrainte est ignorée lorsque le joueur interagit avec un élément.
Speaking	
Color	Si votre objet est capable de parler, vous pouvez personnaliser la couleur des répliques via une liste de 16 couleurs prédéfinies. La valeur RGB de chaque couleur est modifiable depuis la page du projet si elle ne vous convient pas.
Movie style	Si l'option est activée, les répliques s'affichent en bas de l'écran comme des sous-titres de film. Si elle est désactivée, les répliques s'affichent au-dessus du personnage.
Avatar	Pour afficher la photographie ou l'avatar du personnage qui prend la parole, vous devez sélectionner un objet parmi la liste. L'animation TALK est jouée quand le personnage garde la parole et l'animation STOP lorsque le joueur est amené à choisir une réponse. L'animation STOP est utilisée à la place de l'animation TALK si cette dernière est absente. Si l'animation STOP ne contient pas d'images, aucun avatar n'est affiché au moment du choix. Les paramètres du groupe Avatar seront à renseigner dans l'objet en question.
Avatar	
Image layout	L'emplacement de l'image relatif à la résolution du jeu en pixels sous la forme « x, y, largeur, hauteur ». X et Y identifient le coin en haut à gauche du cadre.
Text layout	L'emplacement du texte relatif à la résolution du jeu en pixels sous la forme « x, y, largeur, hauteur ». X et Y identifient le coin en haut à gauche du cadre.
Text align	L'alignement du texte dans le cadre. Par défaut le texte est centré verticalement et horizontalement.

Darkness	Permet s'assombrir la scène entière pour faire ressortir l'avatar et son texte. La valeur est comprise entre 0 et 100 . Cette option est ignorée si l'animation STOP ne contient aucune image au moment de la sélection d'une réplique.
Interaction	
Enabled	Cette option permet de désactiver les interactions possibles avec l' objet . Vous pouvez modifier cette valeur depuis les scripts avec les fonctions enable et disable . Si votre objet n'a pas de titre, il est préférable de désactiver les interactions pour éviter d'obtenir des constructions de phrases incomplètes.
Bounding box	Par défaut la zone active de l' objet est définie par un rectangle, aligné sur les axes X/Y et englobant tous les pixels de la frame en cours. Ce rectangle peut donc être différent de la taille de l'image. En désactivant l'option, vous obtiendrez une précision plus accrue basée sur une détection de pixels de vos images. Bien entendu, cette précision a un coût non négligeable pour les hautes résolutions et les objets animés. Je vous conseille de l'activer seulement en cas de nécessité justifiée.
Sub-objects	Cette option permet de désactiver les interactions possibles avec les sous-objets . Vous pouvez modifier cette valeur depuis les scripts avec les fonctions enable_subs et disable_subs . Si votre sous-objet n'a pas de titre, il est préférable de désactiver les interactions.
Graphics	
Adjustment	L'ajustement est appliqué au moment de la génération du jeu.
Animated FX	Vous pouvez associer un shader interne Flame pour simuler un effet de flamme. Le shader interne utilisera la composante alpha de vos images PNG comme valeur d'opacité, et la couleur RGB affectera la teinte des flammes. Deux autres paramètres vous permettront d'ajuster la vitesse d'animation et l'échelle de rendu.
Monochrome tint	En mode monochrome, le format des textures est converti en appliquant une palette de deux couleurs : votre couleur personnalisée de la propriété au format

	RGBA et la transparence. Les images sont alors codées en deux bits seulement, ainsi une texture 32 bits permet de stocker jusqu'à 32 textures dans ce mode idéal pour compresser des silhouettes. Si votre couleur est à 0,0,0,0 alors le mode monochrome est désactivé.
Optimization	
Pack size	seccia.dev génère des atlas de textures pour limiter le nombre de textures à charger au runtime par la carte graphique. Un atlas de textures est un fichier PNG contenant plusieurs images écartées de deux pixels transparents afin d'optimiser le temps de chargement. De plus, les tailles exprimées en pixels sont variables pour économiser de la mémoire graphique. Il vaut mieux avoir trois textures de 512x512 qu'une seule texture de 1024x1024 pour économiser 262144 pixels. La taille maximale de l'atlas de textures est modifiable. Par défaut la valeur est 2048 (sous-entendu 2048x2048). Si vos images sont plus grandes que la taille maximale fixée, elles seront alors redimensionnées au détriment de leur qualité.
Low quality	Cette option permet de diviser par deux la taille de toutes les textures de l' objet afin de gagner en performance : soit quatre fois moins gourmand ! Mais des artéfacts pourraient dégrader le rendu.
Trim frames	Par défaut les images sont découpées pour éviter de laisser des portions vides sans pixels. Il est parfois utile de désactiver cette option, notamment pour afficher des objets dans le générique.

Animation

L'éditeur se base sur l'arborescence des dossiers pour établir la liste des animations. Un dossier représente une animation contenant les **frames** de toutes les directions au format **PNG**.

Les quatre directions primaires sont **RIGHT**, **LEFT**, **FRONT** et **BACK**. Les quatre directions secondaires sont **FL** (*front left*), **FR** (*front right*), **BL** (*back left*) et **BR** (*back right*). Les directions secondaires sont utiles si vous désirez réaliser un déplacement en huit directions. Lorsque l'**objet** n'a qu'une seule direction, il est préférable d'utiliser **RIGHT**. Aucune direction n'est obligatoire.

Les directions d’une animation peuvent contenir un nombre différent de **frames**. L’index d’une **frame** doit être compris entre **0** et **249** (soit au maximum 250 images par direction). L’index est formaté sur trois caractères.

O_MYOBJ\ANIM\STOP\BACK_000.png
O_MYOBJ\ANIM\STOP\FRONT_000.png
O_MYOBJ\ANIM\STOP\RIGHT_000.png
O_MYOBJ\ANIM\WALK\BACK_000.png
O_MYOBJ\ANIM\WALK\BACK_001.png
O_MYOBJ\ANIM\WALK\FRONT_000.png
O_MYOBJ\ANIM\WALK\FRONT_001.png
O_MYOBJ\ANIM\WALK\RIGHT_000.png
O_MYOBJ\ANIM\WALK\RIGHT_001.png

Direction	
Mirror	Les directions LEFT , BL et FL peuvent être déterminées par les directions RIGHT , BR et FR en appliquant un effet miroir si l’option est activée. L’avantage premier est de limiter la taille et le nombre de textures.
Turning to	Par défaut, il n’y a aucune transition lorsque le personnage change de direction. Pour en rajouter, il faut indiquer l’animation cible dans ce champ pour chaque transition. Par exemple si vous souhaitez créer une transition pour passer de la direction LEFT à la direction RIGHT . Vous devez d’abord créer une nouvelle animation TURN_LEFT (nom au choix) avec des frames RIGHT_*.png où le personnage se tournera de la droite vers la gauche, et ensuite vous choisirez l’option LEFT . Dans l’autre sens, c’est le même principe en ajoutant des frames LEFT_*.png à l’animation TURN_RIGHT .
Frames	
FPS	Le nombre de frames par seconde.
Action frame	Par défaut, un événement est appelé à la fin d’une animation, c’est-à-dire soit à l’index de la dernière frame soit à l’index -1 . Dans certaines situations, par exemple pour une animation TAKE , le déclenchement de l’événement serait plus judicieux au moment de la prise de l’ objet .
Loop count	L’animation peut être jouée une ou plusieurs fois en boucle. Spécifiez -1 pour une boucle infinie. Pour les

	trois principales animations, la boucle est forcément infinie.
Range min/max	Vous pouvez créer une sous boucle en définissant la frame initiale et la frame finale. Spécifiez -1 pour utiliser la dernière frame de l'animation.
Range loop count	Le nombre de boucles de la plage définie précédemment.
Profile	Le profil permet de changer les caractéristiques de la parole. Pour plus d'information, référez-vous à la section Parole de ce chapitre.
Optimization	
Group	<p>Il est très important de grouper les atlas de textures en fonction de l'usage des animations. Si vous avez une animation jouée qu'une seule fois dans le jeu, il n'est pas forcément nécessaire de la conserver en mémoire en permanence. Il existe différentes manières de procéder selon le mode d'usage.</p> <p>Loaded with object : Par défaut l'animation est chargée avec l'objet quand il est présent dans une scène. C'est le mode sans optimisation.</p> <p>Loaded only for scene : L'animation est chargée en mémoire uniquement à l'ouverture de la scène spécifiée. Aucun intérêt si l'objet se trouve dans une unique scène.</p> <p>Loaded on the fly : L'animation est chargée entièrement juste avant son apparition à l'écran, puis déchargée à la fin de la lecture. Une légère attente peut se faire ressentir si l'animation est volumineuse et d'autant plus si la carte graphique est peu performante.</p> <p>Streaming : L'animation est chargée image par image comme pour une vidéo. Ce mode est à tester sur des petites configurations pour être sûr que le FPS de l'animation puisse être garanti à vos joueurs.</p> <p>Les versions low permettent de diviser par deux la taille des textures.</p>

Sous-objet

Les **sous-objets** sont des sous parties rectangulaires et interactives de l’image. Dans le même **objet**, si vous souhaitez en effet avoir plusieurs zones cliquables avec des noms différents, vous pouvez ajouter des **sous-objets**. Il ne faut pas confondre **sous-objet** et portion rectangulaire. Les **sous-objets** permettent seulement de déclarer les éléments héritant de votre **objet**, par exemple le chapeau ou la canne d’un personnage. Il faut ensuite définir le rectangle cliquable sur toutes les **frames** nécessaires. La détection de collision au pixel de l’**objet** est également prise en compte pour les **sous-objets**.

Après avoir créé et sélectionné un **sous-objet**, de nouvelles propriétés s’affichent dans deux nouvelles rubriques : **Frame** et **Sub-object**.

Concernant d’abord l’identité du **sous-objet** sélectionné :

ID	
Name	Le nom du sous-objet . Une couleur lui sera automatiquement attribuée dans l’éditeur pour différencier les rectangles.
Title	Il s’agit du titre du sous-objet . Si le champ est vide, l’interaction sera limitée aux sélections.

Concernant ensuite les propriétés de la **frame** sélectionnée :

Interaction	
Rectangle	Les coordonnées du rectangle. Une autre manière de définir le rectangle consiste à sélectionner la portion de l’image en restant appuyé sur le bouton droit de la souris. Vous pouvez utiliser la barre d’outils pour copier en mémoire les coordonnées courantes afin de les appliquer rapidement sur d’autres frames .

Inventaire

L’asset **Object** propose une liste de quatre icônes personnalisables. Une seule icône peut être utilisée à la fois et s’afficher dans la barre d’inventaire ou à la place du curseur de la souris.

Les icônes sont des fichiers **PNG** de **256x256** pixels de préférence. Elles peuvent être inférieures ou supérieures mais doivent être carrées. Pour changer les icônes, vous pouvez cliquer sur l'aperçu pour parcourir vos dossiers et choisir un nouveau fichier. L'image sera redimensionnée si besoin. Vous pouvez aussi passer par l'explorateur de **Windows** comme pour les animations.

Durant la partie, pour changer l'icône courante, spécifiez le nouvel index avec la fonction **set_icon**.

```
set_icon 0_BOTTLE 1
set_title 0_BOTTLE 1
```

Un des intérêts de cette fonctionnalité est de pouvoir changer l'apparence de l'**objet** après une résolution d'énigme sans créer de nouveaux **objets**. Par exemple : le remplissage d'une bouteille, l'ouverture d'une boîte de conserve ou plus généralement le changement d'état d'un **objet**. Le changement d'icône implique souvent un changement de titre.

Clone

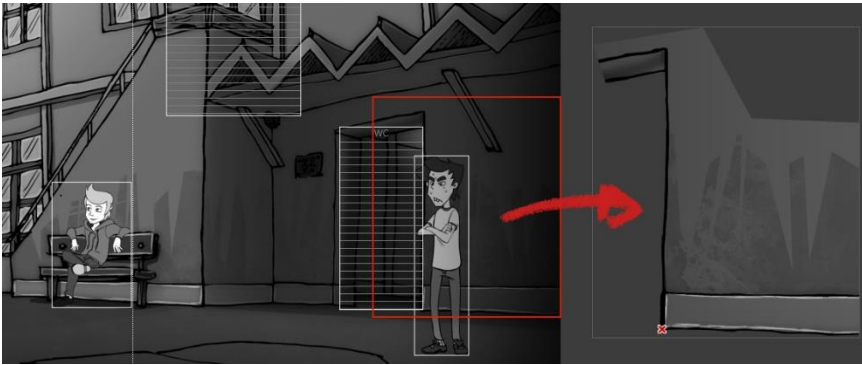
Parfois les **scènes** ont besoin de dupliquer les **objets** à l'écran, or il n'existe pas de notion d'instance dans **seccia.dev**. C'est un choix arbitraire car j'ai voulu simplifier le développement des jeux d'aventure pour faciliter la tâche aux créateurs. Et il est de toute façon assez rare d'instancier des **objets** dans un **Point & Click** ou un **Visual Novel**, même plus largement dans un jeu d'aventure narratif contrairement à un **Shoot'em up**. Cela dit, devoir dupliquer les ressources d'un **objet** dans **seccia.dev** n'est pas pour autant une option raisonnable, d'où cette notion de clonage.

Un clone est un **objet** à part entière qui partagent les ressources graphiques d'un autre **objet**. Ce n'est pas simplement une référence d'un autre **objet**. Pour que cela fonctionne, l'**objet** original doit se trouver dans la même **scène** que ses clones. Pour être plus précis, ce sont les ressources graphiques qui doivent être chargées en mémoire pour que les clones puissent y accéder.

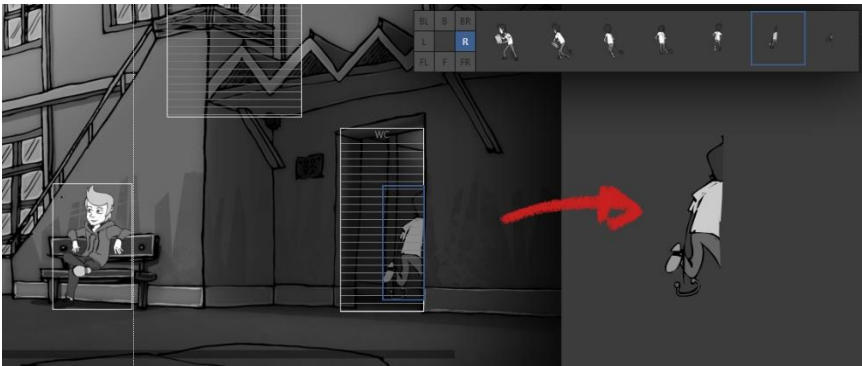
Après avoir sélectionné un **objet** à cloner, de nombreuses propriétés s'éclipseront pour laisser place uniquement aux propriétés propres au clone.

Masque

Vous pouvez facilement retirer une portion d'image sur une **frame** en appliquant un masque d'exclusion. J'ai utilisé cette technique dans le jeu **Désiré** au moment où David part aux toilettes à toute vitesse.



La partie droite du mur est un **objet** positionné par-dessus le décor. Il sert à gérer l'entrée et la sortie de Désiré. Quand il se rapproche de la porte, le mur s'affiche par-dessus et permet de donner l'illusion que le personnage entre dans la salle. Or le déplacement de David est déjà inclus dans l'animation, c'est-à-dire que l'**objet** ne change pas de position XY et ne permet pas d'inverser l'ordre d'affichage des éléments. Afin d'obtenir la même illusion, il faudrait découper le personnage sur les trois dernières **frames** de l'animation manuellement avec précision à l'aide d'un logiciel graphique. Fastidieux ?

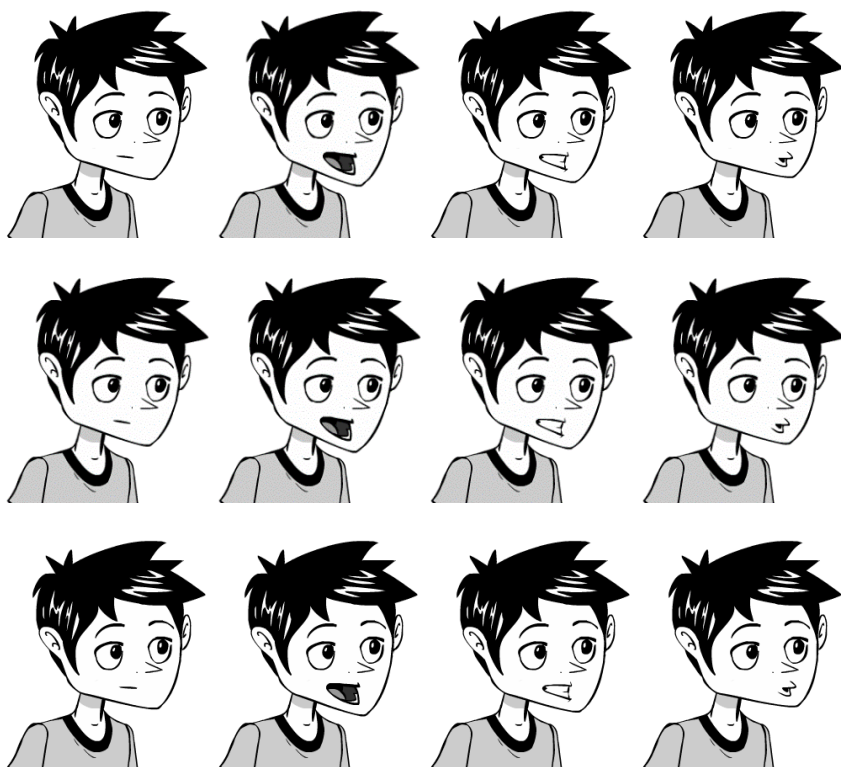


Vous pouvez le faire plus facilement en sélectionnant une **frame** de la séquence pour ensuite cliquer sur le bouton **Mask** et modifier le **PNG**. L'éditeur vous demandera de choisir la **scène** en question et l'**objet** correctement placé servant de masque : ici il s'agit du mur. Répétez la manip si nécessaire sur les autres **frames** en cliquant sur le bouton **Repeat**. En cas d'erreur, ne vous inquiétez pas, les fichiers d'origine sont conservés et peuvent être restaurés en cliquant sur le bouton **Revert**.

Parole

Vous devez fournir trois inclinaisons de tête pour que **seccia.dev** puisse piocher les **frames** aléatoirement, en appliquant des contraintes selon le profil sélectionné. Bon à savoir : l'algorithme ne tirera jamais au sort deux fois la même image.

Les inclinaisons doivent contenir des poses faciales identiques. Si votre animation possède quatre poses, alors il vous faudra douze **frames** en tout (soit 4 poses x 3 inclinaisons) comme le montre l'exemple ci-dessous.



Les images sont numérotées de 0 à 11 de gauche à droite et de haut en bas. La première ligne correspond à l'inclinaison normale de la tête. La seconde à l'inclinaison vers le bas et la troisième à l'inclinaison vers le haut. Notez que les inclinaisons doivent être à peine perceptibles pour obtenir un rendu naturel.

Intégration par le code

Il y a une subtilité importante à connaître dès à présent sur les **objets**. Il faut bien faire la distinction entre les trois modes d'usage existants lorsque vous manipulez les **objets**. C'est capital pour comprendre le sens des fonctions et pourquoi sont-elles réparties dans trois catégories dans la documentation.

Nous avons vu dans ce chapitre, la manière de créer et de paramétrer un **objet** depuis l'éditeur mais ce n'est pas la seule méthode, certaines propriétés sont également accessibles par l'intermédiaire des scripts. Toutes les fonctions permettant de lire ou de modifier ces propriétés sont présentées sous la catégorie **Object** de la page d'aide. Vous trouverez même des fonctionnalités supplémentaires telles que **kill** et **revive**, non présentes dans l'éditeur.

Lorsqu'un **objet** est placé dans une **scène**, il prend une autre forme d'usage et devient un **Scene object** (**objet de scène**) car ses caractéristiques sont dépendantes de la **scène**. Il faut donc différencier la notion d'**objet** et la notion d'**objet de scène** parce qu'il peut y avoir autant d'**objets de scène** issus du même **objet** que de **scènes**. Le fait de changer le titre d'un **objet**, aura un impact sur tous les **objets de scène** puisque cette propriété est dépendante de l'**objet**. Au contraire, le fait de changer l'animation **TALK** par défaut avec la fonction **set_default_talk** n'impactera que l'**objet de scène**. Les fonctions relatives aux **objets de scène** sont documentées dans **Scene Object** de la page d'aide.

La troisième différence est encore un peu plus subtile car les **objets de scène** ont finalement deux modes d'usage. Le second mode concerne les attributs de l'**objet** qui ne persistent pas au-delà de la **scène**. Si vous démarrez un mouvement avec **walk** ou **start_path** pour ne citer que deux fonctions, et que vous changez de **scène** durant le déplacement, la position ne sera pas enregistrée. Au retour, l'**objet** reviendra à sa position d'origine. Ces fonctions sont rangées dans la section **Play** de la page d'aide.

Les bonnes pratiques

Il est important de bien nommer les animations en les classant par lieu ou par action pour préparer l'optimisation de la mémoire du jeu et limiter au maximum le refactoring qui aura un impact négatif sur vos dates butoirs.

Si une animation est propre à une **scène** et si l'**objet** apparaît à plusieurs endroits, vous pouvez opter pour l'un de ces trois modes d'optimisation.

Only for scene	<p><u>Avantage</u> : Pas de latence au moment de jouer l'animation.</p> <p><u>Inconvénient</u> : Chargement des textures à l'ouverture de la scène même si l'animation n'est pas jouée.</p>
On the fly	<p><u>Avantage</u> : Moins de textures à charger si l'animation n'est pas jouée.</p> <p><u>Inconvénient</u> : Latence importante avant de jouer l'animation due au chargement de toutes les frames.</p>
Streaming	<p><u>Avantage</u> : Moins de textures à charger (frame par frame) et très peu de latence avant de jouer l'animation.</p> <p><u>Inconvénient</u> : Sur certaines configurations matérielles peu performantes, l'animation risque de ralentir et de ne pas être jouée à la bonne vitesse, surtout si la vitesse de l'animation est élevée.</p>

En règle générale, il vaut mieux *streamer* les animations volumineuses et non récurrentes. L'option supplémentaire **Low** vous permettra de réduire considérablement le poids des animations ne nécessitant pas une définition élevée, par exemple un nuage de fumée et de débris occupant l'espace entier de la **scène**.

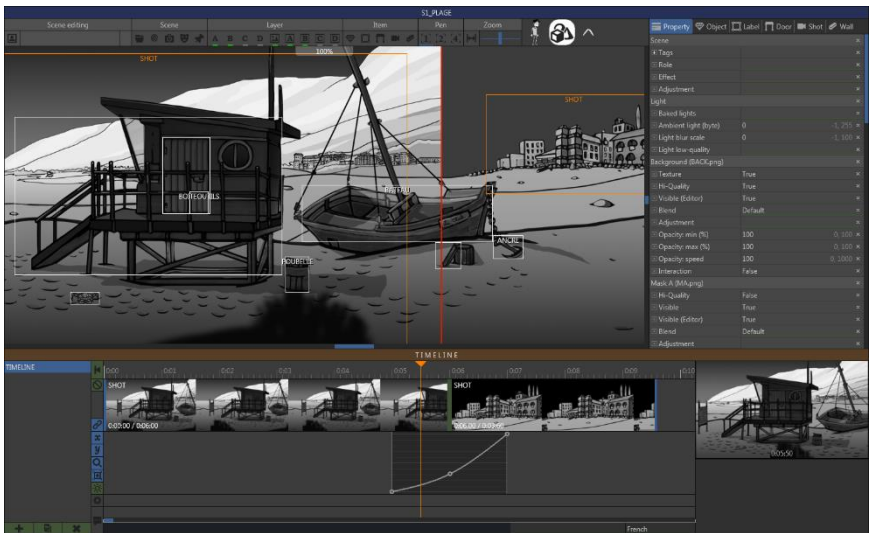
seccia.dev n'est pas si bête, il est capable de ne pas décharger un **objet** si ce dernier est également présent dans la nouvelle **scène** à charger. Si votre personnage principal se trouve donc dans toutes les **scènes**, ses ressources ne seront jamais déchargées durant la partie.

SCÈNE

L'éditeur de **scène** permet de composer les niveaux du jeu en plaçant les **objets** et en ajoutant les interactions nécessaires. La majeure partie de votre temps sera consacrée à cet éditeur.

Une des forces de **seccia.dev** réside dans sa simplicité de définir la zone de marche du personnage et d'associer une action à un comportement. Dans la majorité des cas lorsque le joueur décide de récupérer un **objet** présent dans le décor en cliquant dessus, le personnage doit d'abord se déplacer vers cet **objet** et éventuellement jouer l'animation adéquate pour ramasser l'**objet**. Cet enchaînement d'étapes est réalisable sans écrire une seule ligne de code grâce aux **grids** et **cells** que nous détaillerons ultérieurement.

Vous pourrez également créer des **cinématiques** in-game grâce aux **timelines** en plaçant des **shots** dans votre **scène**.



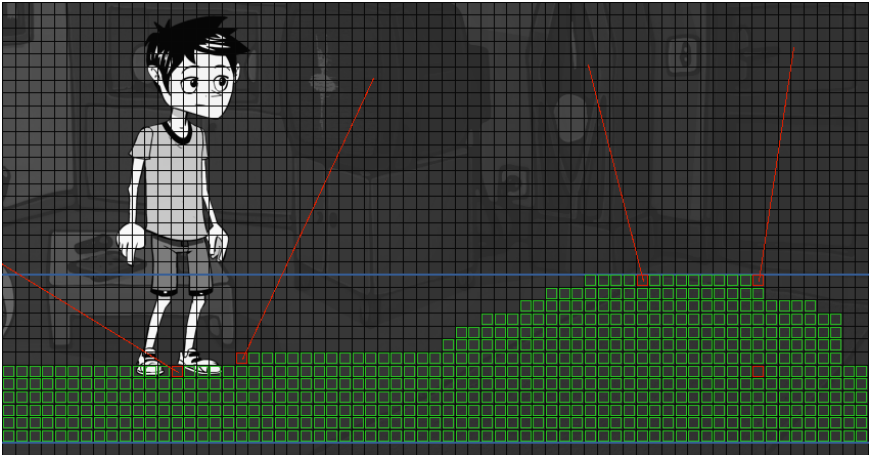
Scène

Une **scène** est composée d'un décor principal, de calques en avant ou arrière-plan, de **stills**, d'**objets** et de zones interactives.

Après avoir créé un nouvel asset **Scène**, la première étape consiste à importer une image au format **PNG** pour remplacer le décor principal par défaut. Depuis la barre d’outils **Scène** de l’éditeur, cliquez avec le bouton droit de la souris sur l’icône **Back** et importez un fichier **PNG**. Sinon ouvrez le dossier de l’asset pour transférer directement le fichier **PNG** via l’explorateur de **Windows**.



Le décor principal doit porter le nom **BACK.png** et être situé dans le dossier **LAYER**. Les dimensions de l’image définissent la taille de la **scène** et par conséquent le nombre de **cells** de la **grid**. Si la taille de la **scène** diffère de la résolution du jeu, des bandes noires seront incrustées ou un défilement horizontal sera appliqué en fonction des paramètres du projet. La taille d’une **cell** est fixée à **16x16** pixels quelle que soit la résolution du jeu.



Il existe deux types de calques : les décors lointains (**FAR**) et les masques. Vous pouvez ajouter au maximum quatre décors lointains et quatre masques par **scène**. L’ordre d’affichage des neuf images, en incluant le décor principal, se présente ainsi, du premier plan à l’arrière-plan.

MA.png	Le masque A affiché au premier plan (le plus proche de la caméra).
MB.png	Le masque B caché par le masque A.
MC.png	Le masque C caché par le masque B.
MD.png	Le masque D caché par le masque C.
BACK.png	Le décor principal caché par les masques qui définit la taille de la scène .

FA.png	Le premier décor lointain.
FB.png	Le second décor lointain.
FC.png	Le troisième décor lointain.
FD.png	Le décor le plus éloigné de la caméra.

Si le décor principal est entièrement opaque, les décors lointains ne seront pas visibles. Les calques peuvent être activés ou désactivés par le code en ajoutant des effets de fondu enchaîné.

LES PROPRIÉTÉS DE LA SCÈNE

Lorsqu’aucun élément n’est sélectionné, vous pouvez accéder aux propriétés de la **scène** dans la liste située en haut à droite de l’éditeur.

Scene	
Tags	Vous pouvez définir jusqu’à quatre tags par scène . Ils vous permettent d’identifier les scènes autrement que par leur nom afin de les regrouper par catégorie. Un cinquième tag est modifiable par la fonction set_asset_tag .
Role	Il est important d’associer un rôle à une scène pour faciliter les interactions entre les deux éditeurs. De plus, cela permet de lancer automatiquement un rôle à l’ouverture d’une scène et de l’arrêter à sa fermeture. Il est donc conseillé de créer toujours un rôle par scène pour implémenter les actions, conditions et événements propres à la scène .
Effect	Un effet peut être appliqué à la scène entière. Si la propriété est vide, la scène utilisera le champ Effect du projet.
Adjustment	L’ajustement est appliqué au moment de la génération du jeu.
Light	
Baked lights	Permet de surdéfinir la propriété définie dans les propriétés du projet.
Ambient light	Permet de surdéfinir la propriété définie dans les propriétés du projet. -1 utilise la valeur du projet.

Light blur scale	Permet de surdéfinir la propriété définie dans les propriétés du projet. -1 utilise la valeur du projet.
Light low-quality	Permet de surdéfinir la propriété définie dans les propriétés du projet.
Bokeh	
Shape width/height	Permet de surdéfinir la propriété définie dans les propriétés du projet. 0 utilise la valeur du projet.
Size	Change la taille de la forme du bokeh en spécifiant une valeur entre 0 et 100 . Cette valeur est prise en compte avec le zoom de la caméra pour ajuster la taille en temps réel.
Max zoom	Change la valeur maximale du zoom en spécifiant une valeur entre 100 et 400 . Si le paramètre est à 100 , le zoom de la caméra n'aura pas d'impact sur la taille du bokeh.
Background (BACK)	
Texture	Permet d'exclure l'enregistrement de la texture dans le build.
Hi-quality	Si l'option est activée, le décor est divisé en plusieurs textures de 1024x1024 . Si l'option est désactivée, le décor est redimensionné en une seule texture de 1024x1024 .
Visible (editor)	Change la visibilité du décor dans l'éditeur.
Blend	<p>Change le mode de fusion du calque.</p> <p>Default : La simulation de transparence utilise des valeurs RGB prémultipliées par la valeur alpha.</p> <p>Addition : Les couleurs sont additionnées pour produire le rendu final. Les pixels noirs deviendront transparents.</p>
Adjustment	Applique un ajustement uniquement au décor.
Opacity : min/max	Définit en pourcentage l'opacité utilisée pour produire un effet de transition entre deux valeurs.
Opacity : speed	Définit le multiplicateur de vitesse de la transition. À 100% , le facteur est égal à 1 .

Interaction	Intercepte le clic du joueur si le pixel est opaque.
Mask (MA, MB, MC, MD)	
Hi-quality	Si l'option est activée, le masque est divisé en plusieurs textures de 1024x1024 . Si l'option est désactivée, le masque est redimensionné en une seule texture de 1024x1024 .
Visible	Change la visibilité du décor dans le jeu.
Visible (editor)	Change la visibilité du décor dans l'éditeur.
Blend	Change le mode de fusion du calque. Default : La simulation de transparence utilise des valeurs RGB prémultipliées par la valeur alpha. Addition : Les couleurs sont additionnées pour produire le rendu final. Les pixels noirs deviendront transparents.
Adjustment	Applique un ajustement uniquement au décor.
Tile X/Y	Applique horizontalement et/ou verticalement une répétition de l'image pour remplir l'espace vide.
Offset X/Y	Décale horizontalement et/ou verticalement l'image en pixels.
Scroll speed X/Y	Change la vitesse de défilement horizontal de la scène . À 100% le masque défile aussi vite que le décor principal, à 50% il défile deux fois moins vite, à 200% il défile de fois plus vite et à 0% la vitesse est ajustée en fonction de la taille du décor principal et du masque.
Parallax	Il s'agit de la parallaxe de profondeur. Pour créer une illusion de profondeur lorsque le personnage s'éloigne de la caméra, il est nécessaire d'accentuer l'agrandissement des calques au premier plan par rapport à l'arrière-plan. Par défaut, tous les calques ont le même facteur d'agrandissement.
Opacity : min/max	Définit en pourcentage l'opacité utilisée pour produire un effet de transition entre deux valeurs.
Opacity : speed	Définit le multiplicateur de vitesse de la transition. À 100% , le facteur est égal à 1 .

Interaction	Intercepte le clic du joueur si le pixel est opaque.
Far (FA, FB, FC, FD)	
Hi-quality	Si l'option est activée, le décor est divisé en plusieurs textures de 1024x1024 . Si l'option est désactivée, le décor est redimensionné en une seule texture de 1024x1024 .
Visible	Change la visibilité du décor dans le jeu.
Visible (editor)	Change la visibilité du décor dans l'éditeur.
Blend	Change le mode de fusion du calque. Default : La simulation de transparence utilise des valeurs RGB prémultipliées par la valeur alpha. Addition : Les couleurs sont additionnées pour produire le rendu final. Les pixels noirs deviendront transparents.
Adjustment	Applique un ajustement uniquement au décor.
Tile X/Y	Applique horizontalement et/ou verticalement une répétition de l'image pour remplir l'espace vide.
Offset X/Y	Décale horizontalement et/ou verticalement l'image en pixels.
Scroll speed X/Y	Change la vitesse de défilement horizontal de la scène . À 100% le décor lointain défile aussi vite que le décor principal, à 50% il défile deux fois moins vite, à 200% il défile de fois plus vite et à 0% la vitesse est ajustée en fonction de la taille du décor principal et du décor lointain.
Parallax	Si l'option est activée, le décor conservera sa taille quel que soit le zoom appliqué à la scène . Cette option est utile pour les décors très lointains (montagnes, ciel, horizon...) qui ne doivent pas être altérés par la distance de la caméra.
Opacity : min/max	Définit en pourcentage l'opacité utilisée pour produire un effet de transition entre deux valeurs.
Opacity : speed	Définit le multiplicateur de vitesse de la transition. À 100% , le facteur est égal à 1.

Interaction	Intercepte le clic du joueur si le pixel est opaque.
--------------------	--

LES ACTIONS DE LA SCÈNE

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **scènes** que nous listons brièvement ci-dessous :

camera	Permet d'effectuer un mouvement de caméra.
examine	Permet d'afficher en grand un objet .
jump	Permet de changer de scène .
popup	Permet d'afficher une fenêtre popup.
timeline	Permet de lancer une timeline .
wait	Permet de faire une pause avant de passer à la boîte suivante.

LES CONDITIONS DE LA SCÈNE

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux **scènes** que nous listons brièvement ci-dessous :

scene is	Permet de connaître la scène en cours ou la scène précédente.
scene has	Permet de connaître la visibilité d'un élément de la scène .

LES ÉVÉNEMENTS DE LA SCÈNE

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux **scènes** que nous listons brièvement ci-dessous :

on click	Lorsque le joueur clique à un endroit de la scène .
on enter scene	Lorsque le joueur change de lieu et entre dans une scène .
on exit scene	Lorsque le joueur change de lieu et sort de la scène en cours.

on input	Lorsque le joueur effectue une action avec la souris ou l'écran tactile.
on select layout	Lorsque le joueur clique sur un bouton personnalisable de l'interface du jeu.

Still

Les **stills** permettent de réduire considérablement le nombre d'**objets** à créer dans votre projet. Ce sont des objets immobiles qui n'auront aucune interaction avec le joueur mais qui seront capables de gérer la profondeur de la **scène**. Les **stills** ne sont pas partagés avec d'autres **scènes** et ne créent aucune dépendance. Ainsi une librairie de **scène** contiendra également les **stills**. Tous les images sont stockées dans un atlas de textures comme pour les **objets**.

Pour ajouter un **still**, faites un clic droit sur la **scène** puis **Add stills** pour sélectionner un ou plusieurs fichiers **PNG**. Sans passer par le menu popup, vous pouvez glisser et déposer vos fichiers depuis l'explorateur de **Windows** à condition d'avoir ouvert le mode **still**. Si l'image est de même taille que le décor principal, elle sera rognée et placée au bon endroit de la **scène**. Grâce à cette astuce, l'artiste pourra exporter les items dans des calques séparés et les **PNG** seront ainsi facilement importables sans les positionner à la main au pixel près. Il ne restera plus qu'à paramétrer le **still**.

LES PROPRIÉTÉS DU STILL

Properties	
Tags	Vous pouvez définir jusqu'à quatre tags par still . Ils vous permettent de regrouper les stills par catégorie. Un cinquième tag est modifiable par la fonction set_still_tag .
ID	L'identifiant unique en lecture seule généré automatiquement à sa création.
Width/Height	La taille de l'image récupérée depuis le fichier.
Name	Le nom du still .
Visible	La visibilité du still par défaut. Les fonctions show_still et hide_still permettent de changer la visibilité dynamiquement par le code. Cela peut permettre de

	changer l'état des items en les superposant (laminaire éteint ou allumé).
X/Y	La position en pixels du still dans la scène . La position correspond au point en haut à gauche du rectangle.
Elevator	Permet de changer la hauteur du still comme pour les objets . La base est toujours située en bas du rectangle et centrée horizontalement. Cette hauteur peut être négative. Les stills sont placés sur le même calque que les objets dessinés après le décor BACK .

Objet de scène

Nous avons déjà développé la notion d'**objet de scène** dans le chapitre précédent. Pour résumer, il s'agit d'une version propre à la **scène** qui possède leurs spécificités.

LES PROPRIÉTÉS DE L'OBJET DE SCÈNE

Properties	
UID	La référence de l' objet .
Tags	Vous pouvez définir jusqu'à quatre tags par objet . Ils vous permettent d'identifier les objets autrement que par leur nom afin de les regrouper par catégorie. Un cinquième tag est modifiable par la fonction set_tag .
Visible in this scene	Force le changement de visibilité de l' objet au lancement de la scène . Par défaut, l' objet garde sa visibilité à travers les scènes en sachant qu'il ne peut être visible à deux endroits. Si vous souhaitez forcer son apparition à chaque lancement de la scène , vous devez changer la valeur en True . Passez par le code si vous avez besoin d'appliquer des conditions.
Parent	Permet de gérer des positions relatives par rapport à d'autres objets , labels ou au curseur de la souris. Vous pouvez vous en servir par exemple pour obtenir un effet de lampe torche dans une caverne.
X/Y	La position de l' objet en pixels. La position tient compte du point pivot défini dans l'éditeur d' objets .

Elevator	Permet de changer la hauteur de l' objet (position dans l'espace et non la taille de l' objet). Si vous avez une table et un vase posé dessus en vue de perspective (et que vous avez défini le point d'ancrage au niveau du socle du vase et au niveau du pied de la table le plus proche de la caméra) vous aurez un souci de profondeur. Rappelez-vous que la coordonnée Y de vos objets est également utilisée comme coordonnée Z pour gérer la profondeur de la scène . Étant donné que la table et le vase n'ont pas la même position Y, et que la table se trouve plus proche de la caméra (Y table > Y vase) vous verrez la table dessinée par-dessus le vase. Pour remédier à ce problème, vous pourriez descendre le point d'ancrage du vase mais à moins d'avoir une raison valable je vous le déconseille. Le point d'ancrage doit être relatif à l' objet et non à la scène dans la majorité des cas. La meilleure solution consiste à compenser cette différence (Y table et Y vase) en ajoutant une hauteur virtuelle au vase afin d'aligner les deux objets sur le même plan : c'est-à-dire le sol. Une ligne verticale s'affichera dans l'éditeur pour indiquer cette hauteur. La fonction set_elevator vous permettra de modifier la hauteur dans vos scripts.
Elevator scaling	Par défaut, le changement de taille ne tient pas compte du paramètre Elevator .
Elevator rotation	Par défaut, la rotation ne tient pas compte du paramètre Elevator .
Parallax	Il s'agit de la parallaxe de profondeur en pourcentage. Pour créer une illusion de profondeur lorsque le personnage s'éloigne de la caméra, il est nécessaire d'accentuer l'agrandissement des objets au premier plan par rapport à l'arrière-plan. Par défaut avec la valeur 100 , tous les objets ont le même facteur d'agrandissement. Si la valeur est égale à 0 , l' objet sera considéré comme un objet très lointain et ne changera jamais de taille.
Speed X/Y	Il est possible de surdéfinir la vitesse de déplacement de l' objet . Si le champ est vide, seccia.dev récupère la vitesse spécifiée dans l'éditeur d' objet .
Placement	Permet de changer l'ordre d'affichage d'un objet . Par défaut les objets sont dessinés juste après le décor

	principal (BACK). La profondeur de la scène sur l'axe Z lorsque les objets se déplacent est dépendante du placement. Cette fonctionnalité s'avère très utile pour ajouter des effets visuels avec une plus grande flexibilité.
Blend	<p>Change le mode de fusion de l'objet.</p> <p>Default : La simulation de transparence utilise des valeurs RGB prémultipliées par la valeur alpha.</p> <p>Addition : Les couleurs sont additionnées pour produire le rendu final. Les pixels noirs deviendront transparents.</p> <p>Soft Light : Similaire au <i>soft light</i> de Photoshop. Les couleurs n'ont aucun effet sur le noir.</p>
HUD coords	Si l'option est activée, les coordonnées de l' objet seront relatives à la caméra .
Drag	<p>Source : l'objet peut être glissé vers un objet ou label.</p> <p>Target : l'objet peut recevoir un objet glisser-déposer.</p> <p>Both : l'objet peut être soit l'objet glissé soit l'objet receveur.</p>
Cheat	Lorsque le joueur reste appuyé sur le bouton du milieu de la souris, une icône définie par l'image CHEAT_OBJECT est affichée à l'emplacement de l' objet .
Light Properties	
Enabled	Permet d'activer ou de désactiver la lumière.
Ambient	Valeur comprise entre 0 et 255 pour changer la lumière ambiante d'une zone. Si la scène a déjà un éclairage ambiant à 255, la lumière n'aura aucun effet. La valeur s'additionne à la valeur d'éclairage ambiant de la scène . Si la scène est assombrie à 50, qu'une première lumière de 20 et qu'une seconde lumière de 10 éclairent la même zone, cette dernière recevra une valeur ambiante de 70.
Diffuse	La couleur est appliquée à la zone éclairée avec ou sans transparence.

Angle	Valeur comprise entre 0 et 360 . La valeur 360 degrés permet d’avoir une lumière omnidirectionnelle.
Direction	Valeur comprise entre 0 et 359 pour définir l’angle de direction. Si la lumière est omnidirectionnelle, la direction est ignorée.
Distance	La distance d’éclairage en pixels, -1 pour l’infini.
Atténuation	Valeur comprise entre 0 et 10 . L’intensité de la lumière est altérée par l’atténuation en fonction de la distance. La distance entre chaque pixel et le point d’origine de la lumière est calculée et convertie en une valeur comprise entre 0 et 1 . Cette valeur est ensuite élevée à la puissance de l’atténuation : pow(dist01, attn) . Si la distance est infinie ou si l’atténuation est à 0 , aucun effet n’est appliqué.
Editor Properties	
Locked	Change le verrouillage de l’ objet dans l’éditeur. Lorsqu’un objet est verrouillé, il ne peut plus être sélectionné via la souris ou les raccourcis sans passer par la liste de l’onglet Object .
Visible	Change la visibilité de l’ objet dans l’éditeur.

LES ACTIONS DE L’OBJET DE SCÈNE

Dans l’éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **objets de scène** que nous listons brièvement ci-dessous :

animate	Permet de jouer l’animation d’un objet .
examine	Permet d’afficher en grand un objet .
light	Permet d’activer ou de désactiver la lumière d’un objet .
path	Permet de démarrer le déplacement d’un objet en suivant un chemin prédéfini.
select	Permet de reproduire un clic sur un objet sans interaction du joueur.
show	Permet de changer la visibilité d’un objet .

stop	Permet d'arrêter le mouvement d'un objet .
walk	Permet de démarrer le déplacement d'un objet via le pathfinding.

LES CONDITIONS DE L'OBJET DE SCÈNE

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux **objets de scène** que nous listons brièvement ci-dessous :

object has	Permet de connaître l'état des éléments d'un objet .
scene has	Permet de connaître la visibilité d'un objet .

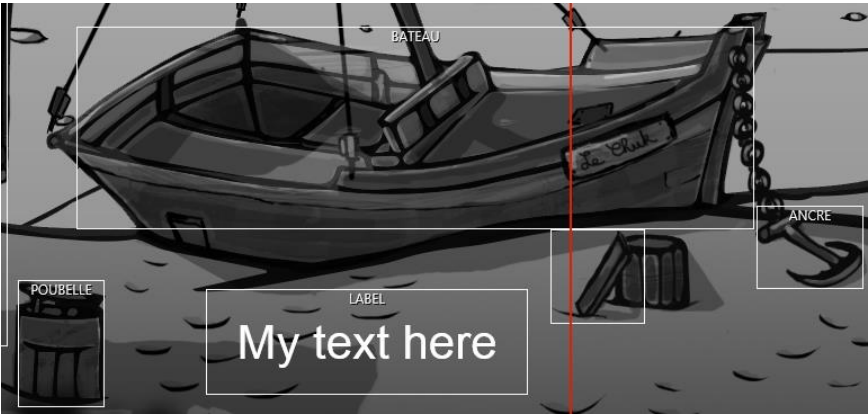
LES ÉVÉNEMENTS DE L'OBJET DE SCÈNE

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux **objets de scène** que nous listons brièvement ci-dessous :

on drag object	Permet de recevoir une demande de glisser-déposer et de pouvoir refuser si nécessaire.
on drop object	Permet de recevoir une notification lorsqu'un objet a été déposé sur aucune cible.
on select object	Permet de recevoir une notification lorsque le joueur clique sur un objet présent dans la scène .
on use label	Permet de recevoir une notification lorsque un objet et un label sont combinés.
on use object	Permet de recevoir une notification lorsque deux objets sont combinés.
on walk	Permet de recevoir une notification lorsque le pathfinding d'un objet est lancé ou arrêté.

Label

Un **label** permet soit de définir une zone rectangulaire interactive sans ajouter d'assets graphiques supplémentaires à la **scène**, soit d'intégrer du texte à l'écran.



LES PROPRIÉTÉS DU LABEL

Properties	
Tags	Vous pouvez définir jusqu'à quatre tags par label . Ils vous permettent d'identifier les labels autrement que par leur nom afin de les regrouper par catégorie. Un cinquième tag est modifiable par la fonction set_label_tag .
Name	Le nom du label sans caractères spéciaux.
Parent	Permet de gérer des positions relatives par rapport à d'autres objets , labels ou au curseur de la souris.
X/Y	La position du label en pixels.
Width/Height	La dimension du label en pixels.
Visible	Change la visibilité du label .
Enabled	Permet de désactiver l'interaction avec le joueur. Par le code il faut utiliser les fonctions enable_label et disable_label . Si votre label n'a pas de titre, il est préférable de désactiver les interactions.
Title	Le titre utilisé pour interagir avec le label . Si le titre est vide, les interactions seront limitées aux sélections. Le séparateur <i>pipe</i> permet d'avoir plusieurs titres et grâce à la fonction set_label_title de pouvoir le sélectionner par son index.
Text	Le texte affiché à l'écran à l'intérieur du cadre.

HUD	Si l'option est activée, le label sera affiché par-dessus les objets et sa position sera relative à l'écran.
Forward	Permet de changer la priorité d'interaction. Par défaut les objets sont prioritaires, c'est-à-dire que si un objet et un label sont superposés, le joueur ne pourra atteindre le label en cliquant sur l' objet . Si l'option est activée, le label devient prioritaire.
Size	La taille du texte en pixels, 0 pour la taille par défaut.
Color	La couleur du texte.
Align	L'alignement du texte dans le cadre. Par défaut le texte est centré verticalement et horizontalement.
Drag	Autorise le label à recevoir un glisser-déposer.
Cheat	Lorsque le joueur reste appuyé sur le bouton du milieu de la souris, une icône définie par l'image CHEAT_LABEL est affichée à l'emplacement du label .
Editor Properties	
Locked	Change le verrouillage du label dans l'éditeur. Lorsqu'un label est verrouillé, il ne peut plus être sélectionné via la souris ou les raccourcis sans passer par la liste de l'onglet Label .
Visible	Change la visibilité du label dans l'éditeur.

LES ACTIONS DU LABEL

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **labels** que nous listons brièvement ci-dessous :

show	Permet de changer la visibilité d'un label .
-------------	---

LES CONDITIONS DU LABEL

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux **labels** que nous listons brièvement ci-dessous :

scene has	Permet de connaître la visibilité d'un label .
------------------	---

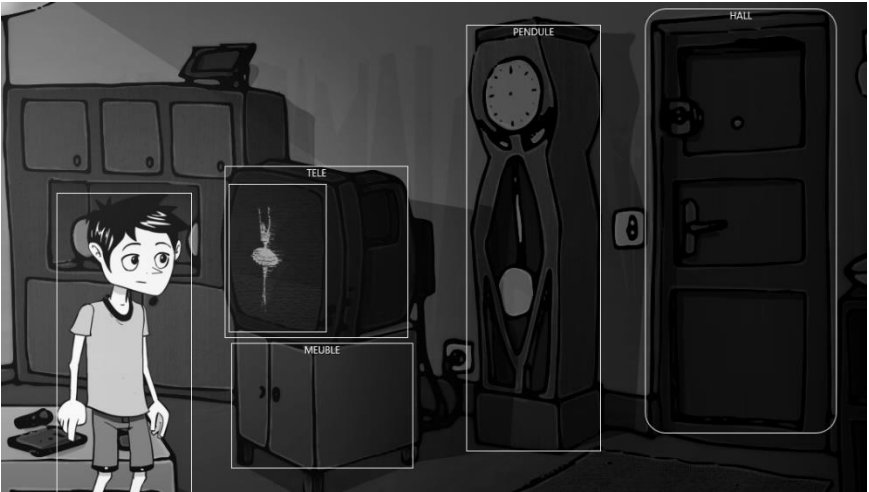
LES ÉVÉNEMENTS DU LABEL

Dans l’éditeur de rôle, certains événements sont spécifiques aux labels que nous listons brièvement ci-dessous :

on select label	Permet de recevoir une notification lorsque le joueur clique sur un label.
on use label	Permet de recevoir une notification lorsqu’un objet et un label sont combinés.

Door

Les doors permettent d’indiquer au joueur un changement de lieu ou un passage menant vers une autre salle. La couleur du texte et le curseur peuvent être différents. Cela dit, il est tout à fait possible de changer de lieu par l’intermédiaire d’un objet ou d’un label. L’avantage des doors est de pouvoir activer l’option SKIP pour écourter le déplacement du personnage lorsque le joueur clique une seconde fois sur la door.



LES PROPRIÉTÉS DE LA DOOR

Properties	
Name	Le nom de la door sans caractères spéciaux.
X/Y	La position de la door en pixels.

Width/Height	La dimension de la door en pixels.
Visible	Change la visibilité du label .
Visible	Permet de changer la visibilité de la door . Par le code il faut utiliser les fonctions show_door et hide_door .
Title	Le titre utilisé pour interagir avec la door . Le séparateur <i>pipe</i> permet d'avoir plusieurs titres et grâce à la fonction set_door_title de pouvoir le sélectionner par son index.
Cheat	Lorsque le joueur reste appuyé sur le bouton de la molette, une icône définie par l'image CHEAT_DOOR est affichée à l'emplacement de la door .
Editor Properties	
Locked	Change le verrouillage de la door dans l'éditeur. Lorsqu'une door est verrouillée, elle ne peut plus être sélectionnée via la souris ou les raccourcis sans passer par la liste de l'onglet Door .
Visible	Change la visibilité de la door dans l'éditeur.

LES ACTIONS DE LA DOOR

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **doors** que nous listons brièvement ci-dessous :

show	Permet de changer la visibilité de la door .
-------------	---

LES CONDITIONS DE LA DOOR

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux **doors** que nous listons brièvement ci-dessous :

scene has	Permet de connaître la visibilité de la door .
------------------	---

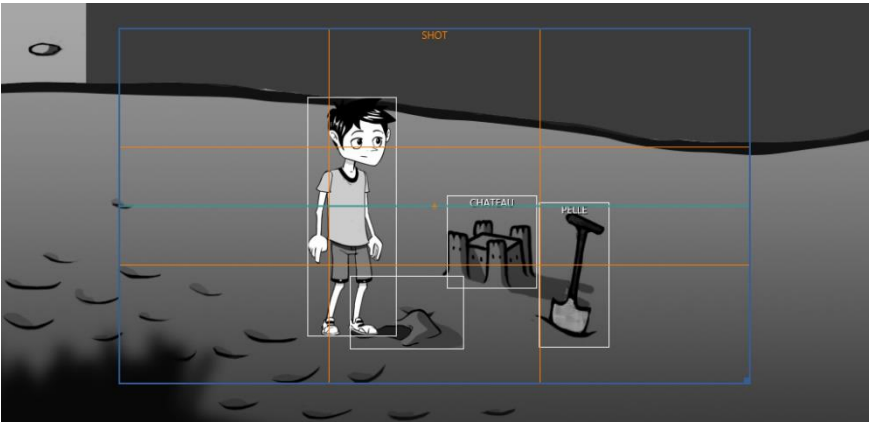
LES ÉVÉNEMENTS DE LA DOOR

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux **doors** que nous listons brièvement ci-dessous :

on select door	Permet de recevoir une notification lorsque le joueur clique sur une door .
-----------------------	--

Shot

Les **shots** ou plans de caméra permettent de définir l’emplacement de la caméra afin d’ajouter des transitions par le code ou par les **timelines**. Lorsqu’un **shot** est sélectionné, une grille s’affiche pour vous aider à composer votre image selon la règle des tiers. En sélectionnant le **shot** puis en maintenant la touche **ALT** enfoncée, tous les guides seront cachés sauf la grille sélectionnée. Il est possible de modifier sa position et sa largeur. La hauteur est automatiquement ajustée en fonction du ratio de votre jeu défini dans les propriétés du projet.



Par le code, vous pouvez activer un **shot** ou faire une transition entre deux **shots** en appelant la fonction **shot** avec des paramètres pour personnaliser l’effet. Cependant, une manière plus intuitive, plus pratique et plus élégante consiste à recourir aux **timelines** documentées un peu plus loin dans ce chapitre.

LES PROPRIÉTÉS DU SHOT

Propriétés	
Name	Le nom du plan sans caractères spéciaux.
X/Y	La position du plan en pixels.
Width	La largeur du plan en pixels.

Editor Properties	
Locked	Change le verrouillage du plan dans l'éditeur. Lorsqu'un plan est verrouillé, il ne peut plus être sélectionné via la souris ou les raccourcis sans passer par la liste de l'onglet Shot .
Visible	Change la visibilité du plan dans l'éditeur.

LES ACTIONS DU SHOT

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **shots** que nous listons brièvement ci-dessous :

shot	Permet de lancer une transition de caméra entre deux shots ou entre la position actuelle de la caméra et un shot .
-------------	--

Wall

Les **walls** permettent de définir des zones d'ombre en utilisant les lumières dynamiques. Ils n'existent pas visuellement et doivent épouser les lignes d'un décor. Il n'est pas possible de changer leur position par le code.

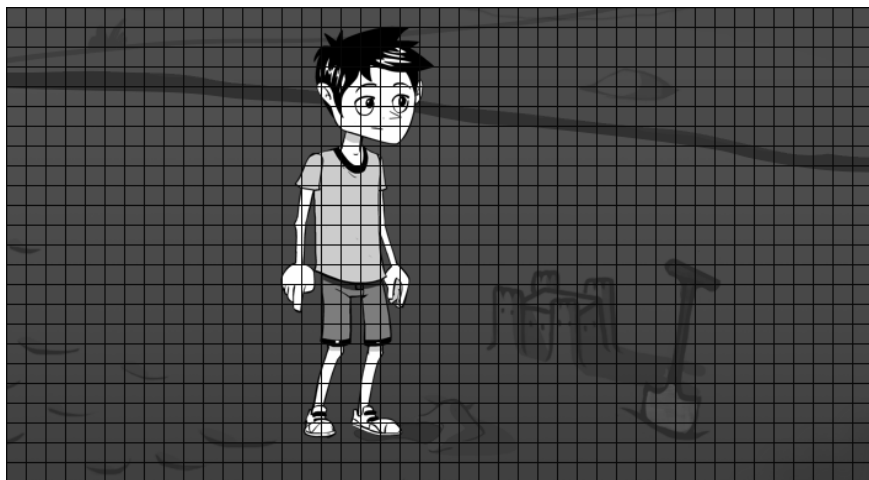
LES PROPRIÉTÉS DU WALL

Properties	
Name	Le nom du wall sans caractères spéciaux.
X/Y	La position en pixels du premier point du segment.
X2/Y2	La position en pixels du second point du segment.
Editor Properties	
Locked	Change le verrouillage du wall dans l'éditeur. Lorsqu'un wall est verrouillé, il ne peut plus être sélectionné via la souris ou les raccourcis sans passer par la liste de l'onglet Wall .
Visible	Change la visibilité du wall dans l'éditeur.

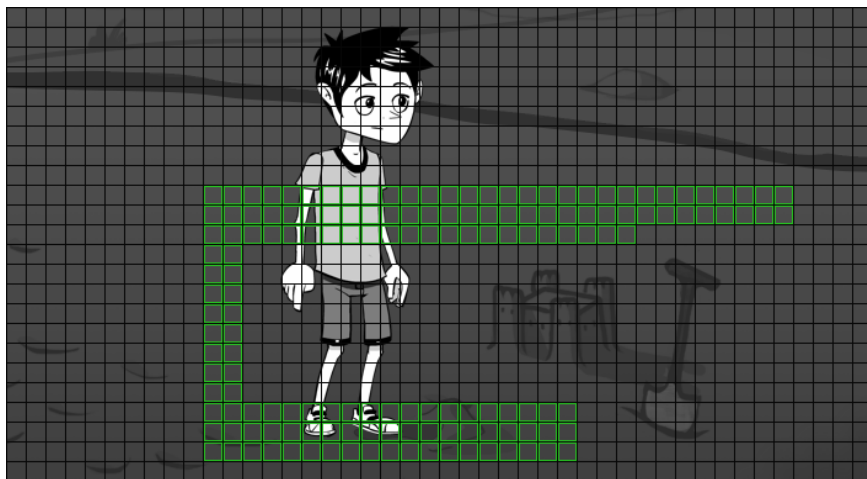
Grid & Cell

La **grid** permet de définir les zones de déplacement et les interactions avec les éléments de la **scène** (**objets**, **labels**, **doors**...). Seuls les **objets** peuvent contenir des **grids** au maximum de huit. Une **grid** est composée de **cells** paramétrables. Pour éditer une **grid**, il suffit de double-cliquer sur un **objet**. En appuyant sur les touches du pavé numérique (0 à 7) vous pourrez changer la sélection de la **grid** en mode édition. Pour changer de **grid** par le code, vous devez utiliser la fonction **set_grid**.

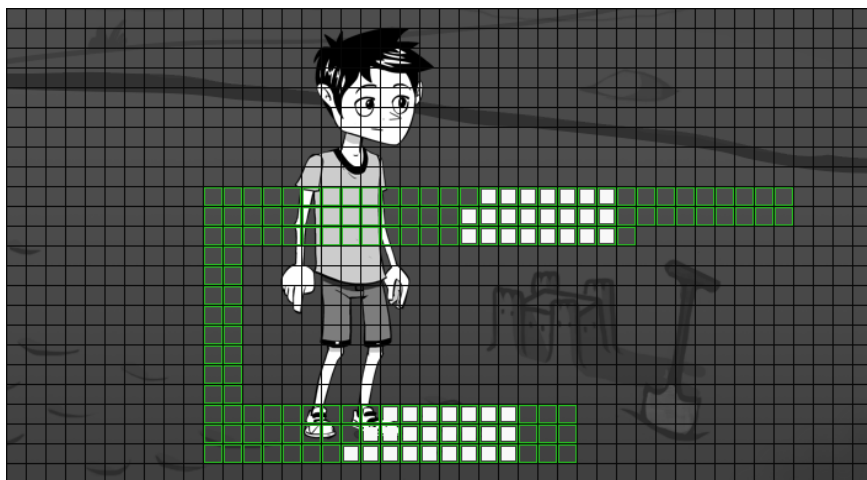
En mode édition de la **grid**, la **scène** entière s'affiche en noir et blanc avec un contraste moins prononcé (excepté l'**objet** sélectionné) et des lignes apparaissent pour délimiter les **cells**.



Pour ajouter une **cell** active sur la **grid**, restez appuyé sur la touche **SHIFT** et cliquez sur la **cell** souhaitée avec le bouton gauche de la souris. Comme avec un pinceau dans un logiciel de dessin, vous pouvez maintenir le bouton gauche enfoncé pour remplir plusieurs **cells** rapidement et même choisir une taille de pinceau plus grande depuis la barre d'outils.



Pour sélectionner les **cells** définies, choisissez le pinceau qui vous convient et cliquez simplement sur les **cells** qui deviendront blanches comme ceci.



Vous pourrez ainsi les supprimer avec la touche **DEL** ou les copier/couper dans le presse-papier avec les raccourcis **CTRL+C/CTRL+X**. La suppression de **cells** peut aussi s'effectuer avec les touches **CTRL** et **SHIFT** en cliquant sur le bouton gauche. Enfin les touches fléchées permettent de déplacer la sélection. Lorsqu'une ou plusieurs **cells** sont sélectionnées, une liste de propriétés relatives aux **cells** s'affiche en haut à droite de l'éditeur que nous allons détailler à présent.

LES PROPRIÉTÉS DE LA CELL

Cell	
Index	La position de la cell sur la grid en nombre de lignes et de colonnes.
Name	Le nom de la cell sans caractères spéciaux.
Event	Si l'option est activée, la cell pourra appeler l'événement on reach cell .
Magnet	Permet de placer l' objet au centre de la cell à la fin d'une marche. Lorsque la cell est liée à une action, cette option est automatiquement activée.
Walkable	Permet de rendre une cell non accessible à la marche. Cette option peut être utile lorsque vous utilisez des ponts et que vous souhaitez personnaliser des cells sur la trajectoire du personnage.
Flag	Attribue une valeur entre 0 et 9 à la cell . Avec la fonction cur_flag , vous pourrez savoir à un instant précis si un objet se trouve sur une cell portant un de ces numéros.
Speed factor X/Y	Multiplicateur de vitesse lorsque l' objet se trouve sur cette cell .
Scale factor	Multiplicateur d'échelle lorsque l' objet se trouve sur cette cell .
Walk animation	Forcer une animation lorsque l' objet se déplace sur la cell .
Walk directions	Liste des directions valides séparées par des espaces, lorsque l' objet se trouve sur la cell . Utile par exemple pour limiter l' objet à une ou deux directions sur une portion de la trajectoire.
ENTER	
Position	Il faut définir une cell FROM et une cell TO afin que le personnage puisse se déplacer de la cell FROM à la cell TO .
Previous scenes	Permet de définir une liste de scènes . Si la scène précédente (provenance du personnage) correspond à une scène de la liste, le déplacement sera alors effectué. Si votre scène n'a qu'un passage d'arrivée, vous pouvez ignorer ce paramètre.

Animation	Permet de jouer une animation à la fin du déplacement. Ne fonctionne qu'avec les cells TO .
Direction	Permet de forcer la direction du personnage à la fin du déplacement. Ne fonctionne qu'avec les cells TO .
SELECT Object	
Linked object	Permet de lier la cell à un objet . Ainsi lorsque le joueur cliquera sur l' objet , le personnage rejoindra automatique la cell avant que l'événement soit appelé.
IF resolved	<p>Par défaut les paramètres suivants sont pris en compte quelle que soit la progression du scénario. Il est possible d'ajouter une condition si vous souhaitez vérifier l'état d'une énigme. Dans ce cas si l'énigme spécifiée n'a pas été résolue, les paramètres seront ignorés.</p> <p>Tolérance : Il s'agit du nombre de cells minimum pour considérer que la destination est atteinte. Si la valeur est à 0, il n'y a aucune tolérance. Si la valeur est à 1, les cells contigües seront admises.</p> <p>Animation : Permet de jouer une animation à l'arrivée.</p> <p>Direction : Permet de force la direction du personnage à l'arrivée. La direction est appliquée à l'animation spécifiée.</p>
SELECT Label	
Linked label	Permet de lier la cell à un label .
IF resolved	cf. SELECT Object
SELECT Door	
Linked door	Permet de lier la cell à une door .
IF resolved	cf. SELECT Object
USE 2 Objects	
Linked object A	Permet de lier la cell lorsque deux objets sont combinés. L' objet A est toujours issu de l'inventaire.
Linked object B	Permet de lier la cell lorsque deux objets sont combinés. L' objet B est toujours l' objet de destination (player ou scène).

IF resolved	cf. SELECT Object
USE Object with Label	
Linked object	Permet de lier la cell lorsqu'un objet est combiné avec un label . L' objet est toujours issu de l'inventaire.
Linked label	Permet de lier la cell lorsqu'un objet est combiné avec un label .
IF resolved	cf. SELECT Object
DETACH Object	
Linked object	Permet de lier la cell lorsqu'un objet contenant deux éléments sont séparés. Pour utiliser cette fonctionnalité, il est nécessaire d'ajouter une image ITEM_DETACH au projet.
IF resolved	cf. SELECT Object

LES ACTIONS DE LA CELL

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **cells** que nous listons brièvement ci-dessous :

walk	Permet de démarrer le déplacement d'un objet via le pathfinding.
------	---

LES CONDITIONS DE LA CELL

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux **cells** que nous listons brièvement ci-dessous :

object has	Permet de savoir si l' objet se trouve sur une cell .
------------	---

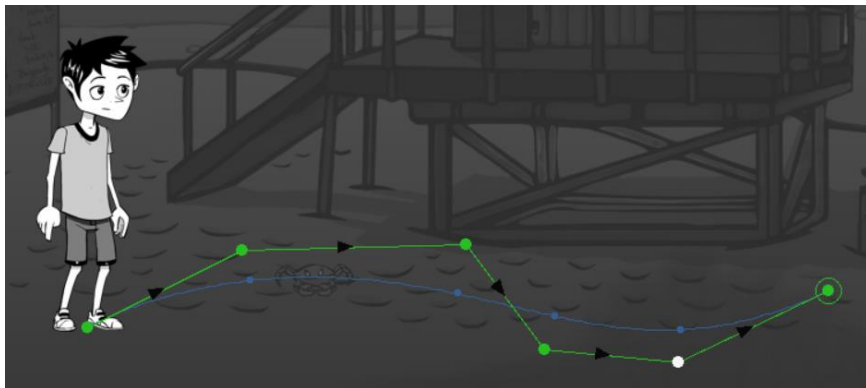
LES ÉVÉNEMENTS DE LA CELL

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux **cells** que nous listons brièvement ci-dessous :

on reach cell	Permet de recevoir une notification lorsqu'un objet atteint une cell .
---------------	--

Path & Spot

Il existe une autre façon de déplacer des **objets** en suivant un **path** prédéfini en courbe ou en ligne droite. Comme pour les **grids**, **seccia.dev** vous propose huit **paths** par **objet** activables avec la fonction **set_path**. Pour éditer les **paths**, sélectionnez l'**objet** puis **Edit paths** via le menu popup.



En maintenant la touche **SHIFT** enfoncée, vous pouvez cliquer sur le bouton gauche de la souris pour ajouter des **spots** dans votre **scène** et ainsi tracer un **path** linéaire. Les **paths** possèdent des propriétés et comme pour les **cells**, chaque **spot** possède ses propres propriétés.

Pour sélectionner un **spot**, il suffit de cliquer sur son point avec le bouton gauche de la souris. Si vous cliquez sur un **spot** en maintenant la touche **CTRL** enfoncée, vous sélectionnerez ou désélectionnerez le **spot** selon son état.

Pour une sélection plus rapide et pratique de plusieurs éléments à la fois, il suffit de maintenir le bouton gauche de la souris enfoncé pour faire apparaître un cercle blanc à l'écran. Tous les **spots** entrant en contact avec ce cercle seront automatiquement sélectionnés. Pour ne pas perdre la sélection actuelle, appuyez sur **CTRL** avant l'affichage du cercle puis relâchez la touche. Pour inverser l'état d'un **spot**, maintenez la touche **CTRL** enfoncée pendant le balayage.

Utilisez les touches fléchées du clavier pour déplacer les **spots** sélectionnés d'un pixel dans la direction souhaitée.

Utilisez les touches **PageUp** et **PageDown** pour passer au **spot** précédent ou au suivant. La touche **Home** placera la sélection au début et la touche **End** à la fin du **path**.

Il est possible d'appliquer une courbe à votre **path** afin de casser la linéarité du déplacement et de le rendre plus naturel grâce à la propriété **Spline** du **path**.

Dans ce cas, la position des **spots** dans le jeu sera définie par la position des points de la courbe.

LES PROPRIÉTÉS DU PATH

Path	
Spline	Il s'agit de la forme de la courbe. Plus la valeur est grande et plus la courbe s'éloignera des lignes droites. Le temps de calcul sera également plus long. Par défaut, la valeur 0 désactive l'utilisation de la courbe.
Speed	La vitesse de déplacement de l' objet en pixels par seconde. Il s'agit de la vitesse initiale qui peut être modifiée par les spots .
Zoom	Permet de tenir compte des valeurs Zoom renseignées dans les spots .
Loop count	Définit le nombre de boucles à effectuer. Si la valeur est -1 , le trajet bouclera à l'infini. La valeur 0 par défaut permet d'effectuer le trajet qu'une seule fois.

LES PROPRIÉTÉS DU SPOT

Spot	
Index	La position du spot dans la liste.
Name	Le nom du spot sans caractères spéciaux.
X/Y	La position du spot en pixels dans l'éditeur.
Speed	Permet de changer la vitesse de déplacement de l' objet . La vitesse est interpolée entre deux spots . Si la valeur est à 0 , la vitesse du spot précédent est alors conservée.
Zoom	Permet de changer le zoom de la caméra. La valeur est interpolée entre deux spots . Si le zoom est à 0 , la valeur du zoom du spot précédent est alors conservée. Cette propriété est ignorée si le Zoom du path est désactivé.
Pause	Si la valeur est supérieure à 0 , l' objet s'arrêtera au spot et patientera pendant le délai indiqué en

	secondes. Si la valeur est -1 , l' objet s'arrêtera sans poursuivre son path . La fonction continue_path permettra de reprendre le path .
--	--

LES ACTIONS DU SPOT

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **spots** que nous listons brièvement ci-dessous :

path	Permet de démarrer le déplacement d'un objet en suivant un chemin prédéfini dans l'éditeur.
-------------	--

LES CONDITIONS DU SPOT

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux **spots** que nous listons brièvement ci-dessous :

object has	Permet de savoir l' objet se trouve sur un spot .
-------------------	---

LES ÉVÉNEMENTS DU SPOT

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux **spots** que nous listons brièvement ci-dessous :

on reach spot	Permet de recevoir une notification lorsque l' objet atteint un spot .
----------------------	--

Drag & Drop

Le drag & drop ou glisser-déposer en français a un réel intérêt dans la réalisation de mini casse-têtes pour accompagner la trame narrative de vos jeux. En cliquant sur un **objet** et en maintenant enfoncé le bouton gauche de la souris, vous pouvez glisser l'**objet** présent à l'écran vers un autre **objet** ou un **label** de la **scène**. La logique est exactement la même sur écran tactile en glissant l'**objet** avec le doigt.

Il faut en premier lieu déterminer un comportement à l'**objet** en choisissant **Source**, **Target** ou **Both** via la propriété **Drag** de l'éditeur de **scène**.

Source	La source est l' objet pouvant être contrôlé par un glisser-déposer. Au moment où le joueur décide de déplacer l' objet , l'événement on drag object est appelé pour autoriser ou rejeter la requête en retournant une valeur différente de 0. Elle est toujours autorisée implicitement.
Target	La cible est un objet ou un label pouvant recevoir les objets sources. Lorsqu'un objet source est déposé sur un objet cible, l'événement on use object est appelé pour notifier la validation du glisser-déposer. Le premier paramètre identifie l' objet source. Pour un label , c'est l'événement on use label qui est appelé.
Both	Dans ce cas, l' objet peut être la source ou la cible.

Lorsqu'un glisser-déposer n'atteint aucune cible, l'événement **on drop object** est appelé.

Gardez en tête qu'en activant le glisser-déposer sur vos **objets** et **labels**, les interactions avec l'inventaire ne seront plus disponibles.

LES ÉVÉNEMENTS DU GLISSER-DÉPOSER

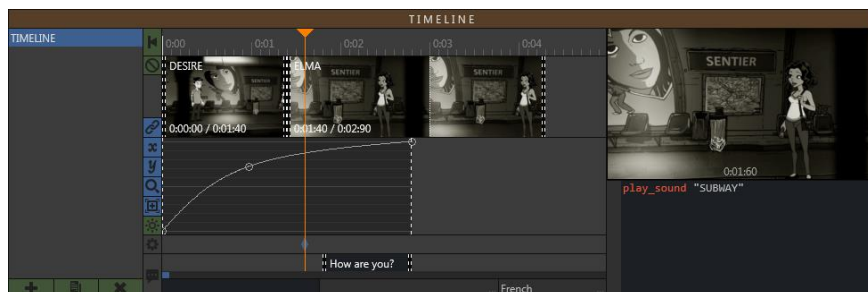
Dans l'éditeur de **rôle**, certains événements sont spécifiques aux glisser-déposer que nous listons brièvement ci-dessous :

on drag object	Permet de recevoir une demande de glisser-déposer et de pouvoir refuser si nécessaire.
on drop object	Permet de recevoir une notification lorsqu'un objet a été déposé sur aucune cible déclarée.
on use label	Permet de recevoir une notification lorsqu'un objet est déposé sur un label déclaré en tant que cible.
on use object	Permet de recevoir une notification lorsqu'un objet est déposé sur un autre objet déclaré en tant que cible.

Timeline

Les **timelines** ont deux principales utilités : créer des cutscenes in-game et des transitions en tirant profit des **shots**. Un des avantages bien pratiques est de pouvoir attacher une musique et ajouter des sous-titres en respectant la

synchronisation de tous les éléments présents sur la **timeline**, y compris l'exécution de scripts.



La première colonne de gauche vous permet d'ajouter et de gérer les **timelines**. Une **scène** peut contenir autant de **timelines** que vous le souhaitez. La partie de droite vous permet de prévisualiser le rendu en temps réel pendant l'édition et également d'écrire vos scripts. Enfin la partie centrale vous permet de gérer les éléments de la **timeline**.

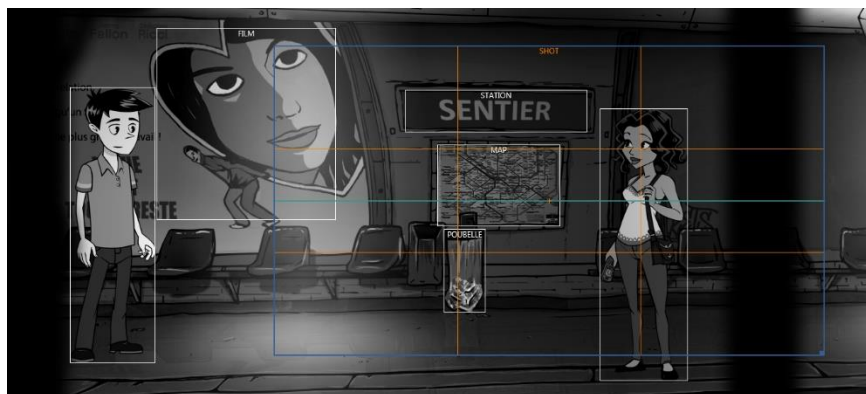


- ① Shots
- ② Transitions
- ③ Scripts
- ④ Sous-titres

Pour jouer une **timeline** pendant la partie, utilisez la boîte **timeline** et pour l'arrêter prématurément utilisez la boîte **stop**.

LES SHOTS

Vous devez d'abord ajouter et placer correctement vos **shots** dans la **scène** comme pour un **label** ou une **door**. Cependant, vous ne pourrez pas directement modifier la hauteur qui est déterminée par deux valeurs : la largeur du **shot** et le ratio de la résolution du jeu.



Quand un **shot** est sélectionné, une règle géométrique (appelée règle des tiers) s'affiche à l'intérieur du cadre pour vous aider à composer votre image. Sans entrer dans les détails, la règle consiste à placer le sujet ou un élément clé soit à une intersection soit sur une ligne. Insérez autant de **shots** que nécessaire mais il est important de les nommer avec si possible un nom court explicite. Par analogie avec un logiciel de montage vidéo, il s'agit pour l'instant d'importer vos rushes au sein de votre projet.

Vous devez ensuite cliquer sur un espace vide de la première rangée de la **timeline** pour ouvrir une boîte de dialogue avec la liste des **shots** disponibles et en ajouter à votre séquence.

LES TRANSITIONS

Lorsque deux **shots** sont côte à côte, une barre verte apparaît en dessous du raccord pour vous permettre d'appliquer une transition linéaire en cliquant dessus. Vous pouvez bien évidemment ajuster la durée de la transition en étirant le cadre par ses extrémités.

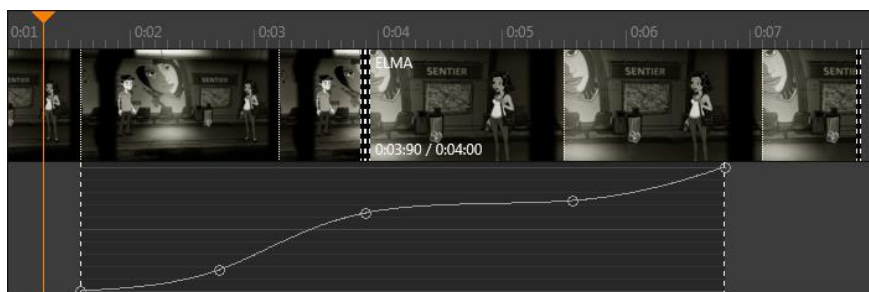


Une fois la transition appliquée, il n'est plus possible de détacher les deux **shots** sans la retirer explicitement par le menu popup.

L'INTERPOLATION SPLINE

Remarquez qu'il est très facile d'agir de manière indépendante sur la position X et Y de la caméra, ainsi que sur le zoom, le flou, la profondeur de champ, le focus et le fondu au noir grâce aux boutons mis à disposition sur le bord gauche. Le premier bouton symbolisé par une chaîne permet de contrôler un groupe de paramètres.

Par défaut l'interpolation est linéaire en utilisant deux points opposés qui peuvent être repositionnés sur l'axe des ordonnées. Pour bénéficier de l'interpolation **Spline** afin de créer des accélérations et décélérations plus fluides, vous devez ajouter des points intermédiaires. Pour cela, il suffit de cliquer avec le bouton gauche de la souris sur la courbe à l'endroit désiré. Un simple clic droit pour les retirer.



Le nombre de points intermédiaires est limité à huit, soit dix points au total par transition en incluant les extrémités.

LA MUSIQUE ET LES SOUS-TITRES

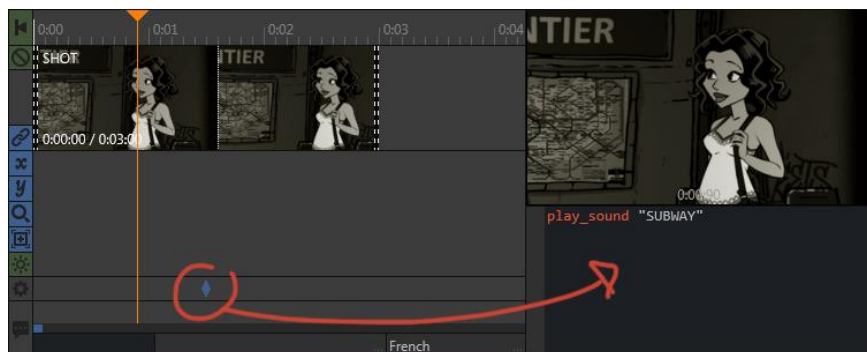
Pour synchroniser une musique, au lieu de passer par les scripts, ajoutez un fichier **OGG** dans le dossier **SONG** du projet et sélectionnez-la ensuite pour chaque langue directement depuis la liste déroulante en bas à droite de la **timeline**.

Les sous-titres de la **timeline** fonctionnent de la même façon que pour sous-titrer un film. Déplacez le curseur à l'endroit où vous souhaitez commencer le sous-titre et entrez le texte dans le champ en dessous. Comme pour les **shots** et les transitions, vous pouvez étirer leur cadre pour ajuster la durée. Les sous-titres ne sont pas prévisualisables dans l'éditeur.

Contrairement aux **cinématiques**, les sous-titres sont plus facilement traduisibles puisqu'ils sont intégrés aux fichiers **CSV** et entrent dans la boucle de production.

LES SCRIPTS

Un autre avantage des **timelines** est de pouvoir insérer un script à n'importe quel moment de la séquence, notamment pour lancer des actions, jouer des animations ou des sons de manière scénarisée.



Cliquez à l'endroit où vous souhaitez placer un nouveau script sur la **timeline** et ajoutez du code dans le champ texte en dessous de la prévisualisation.

Intégration par le code

Tout d'abord vous devez ouvrir une **scène** par l'intermédiaire de la fonction **jump** ou **jump_back**. Pour **jump** le premier paramètre est le nom de la **scène** et le second paramètre est le délai de transition en millisecondes. La fonction **jump_back** permet de rouvrir la **scène** précédente sans spécifier son nom.

```
jump S_GARDEN 500
```

Pour identifier la **scène** en cours ou la **scène** précédente, vous pouvez vous servir des fonctions **scene** et **old_scene**.

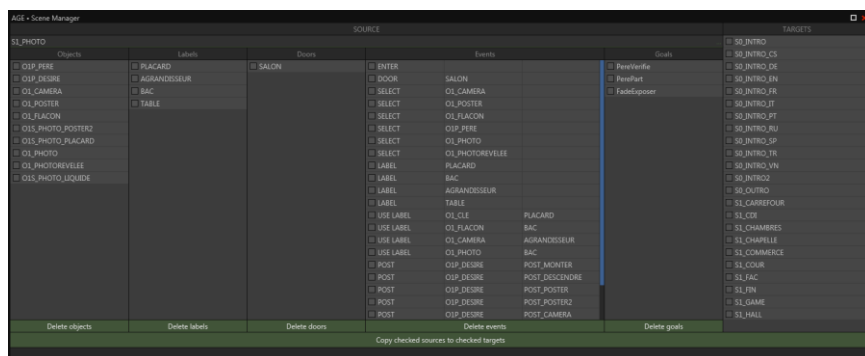
```
if scene S_GARDEN S_HOUSE
end
```

Dans le code ci-dessus, la condition est vraie si la **scène** en cours est soit **S_GARDEN** soit **S_HOUSE**.

Gestionnaire de scènes

Le gestionnaire de **scènes** est accessible depuis le menu principal du logiciel. Le but de ce gestionnaire est de pouvoir dupliquer les éléments d'une **scène** vers

d'autres **scènes** en quelques clics. Vous pouvez aussi supprimer des éléments de la **scène** sélectionnée. Les éléments sont les **stills**, les **objets**, les **labels**, les **doors**, les **shots**, les **walls** et les **timelines**.



Pour dupliquer des informations, sélectionnez d'abord la **scène** et cochez les éléments souhaités. Ensuite dans la colonne de droite, cochez les **scènes** à modifier et cliquez sur le bouton pour lancer la copie. Si un élément existe déjà dans une **scène**, il sera remplacé sauf pour les **stills**. Pour ce dernier, de nouveaux éléments seront créés.

Les bonnes pratiques

Voici quelques conseils pratiques pour optimiser les performances du jeu :

- Désactiver **Hi-quality** de vos calques si vous n'avez pas besoin de conserver la haute-fidélité du fichier original (par exemple pour les calques d'éclairage et amorces en silhouette).
- Fusionner les calques en amont avec votre logiciel d'édition graphique préféré.
- Limiter le nombre d'**effets** à appliquer à la **scène**, surtout pour les appareils mobiles. Vous pouvez aussi utiliser les ajustements appliqués aux textures à la génération de l'application comme méthode alternative pour trouver le meilleur compromis entre qualité et performance.
- Limiter les **objets** statiques de grande taille en les fusionnant avec le décor et les calques si possible, ou utilisez les **stills** pour bénéficier de la profondeur. Plus il y aura d'**objets** à afficher dans la **scène** et plus le jeu ralentira. Je vous rappelle que les **stills** sont regroupés dans une texture commune.

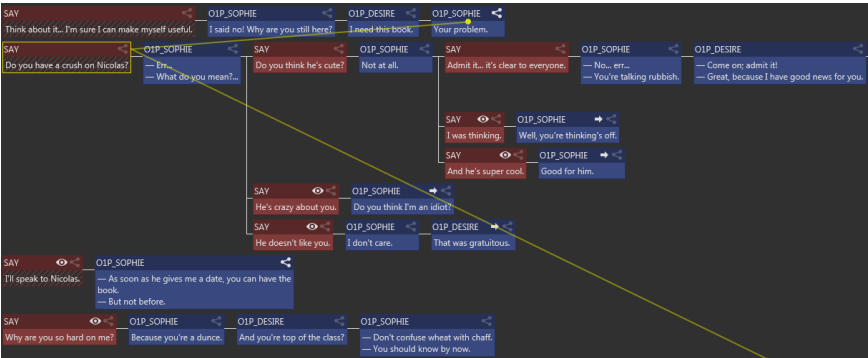
- Limiter le nombre d'images des animations ou utiliser le mode **Streaming** dans l'éditeur d'**objet**.
- Limiter le nombre de lumières et activer l'option **Baked lights** si applicable.
- Activer la compression **PNG** 8 bits avec **PngQuant** depuis les paramètres du projet. À moins d'utiliser des photographies, la différence sera à peine perceptible.

DIALOGUE

L'éditeur de **dialogue** permet d'écrire et de structurer vos conversations interactives. Pour un jeu d'aventure narratif, vous serez souvent amené à travailler en parallèle sur l'éditeur de **scène**, de **rôle** et de **dialogue** pour déclencher les changements d'état. Vous pouvez faire l'impasse sur cet éditeur si votre gameplay est essentiellement basé sur des énigmes d'**objets** sans texte narratif, comme ce fut le cas pour mon jeu **Vive le Roi**.

Pour des raisons pratiques, durant une conversation, le joueur n'a pas la possibilité de sauvegarder la partie. Il doit d'abord quitter la conversation lorsqu'une liste de choix lui est proposée.

Un accès rapide à la langue courante permet de traduire ou de vérifier les phrases sans exporter les fichiers **CSV**.



Dialogue

Un **dialogue** est constitué d'une arborescence de choix en bleu et de répliques en rouge.

LES PROPRIÉTÉS DU DIALOGUE

Dialog	
Tags	Vous pouvez définir jusqu'à quatre tags par dialogue . Ils vous permettent d'identifier les dialogues

	autrement que par leur nom afin de les regrouper par catégorie. Un cinquième tag est modifiable par la fonction set_asset_tag .
--	---

LES ACTIONS DU DIALOGUE

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux **dialogues** que nous listons brièvement ci-dessous :

stop	Permet de terminer la conversation en cours.
talk	Permet de lancer une conversation en ayant la possibilité de définir la phrase de début et la phrase de fin.

LES CONDITIONS DU DIALOGUE

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux **dialogues** que nous listons brièvement ci-dessous :

dialog is	Permet de connaître le dialogue en cours.
------------------	--

LES ÉVÉNEMENTS DU DIALOGUE

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux **dialogues** que nous listons brièvement ci-dessous :

on enter dialog	Permet de recevoir une notification lorsque le dialogue démarre.
on exit dialog	Permet de recevoir une notification lorsque le dialogue se termine.

Boîte rouge « Choice »

Les boîtes rouges permettent de proposer au joueur un choix de répliques à différents moments de la conversation. Ces répliques sont attribuées au **player** courant.



SAY	O1P_VIEUX		O1P_DESIRE
Why do you always watch the same videos?	<ul style="list-style-type: none">• The more you focus on something, the more you learn.• Sometimes, sequences I have watched many times before appear brand new to me.• There's always that golden nugget that escaped us...		But you're missing out on TV.
SAY	O1P_VIEUX	O1P_VIEUX	O1P_DESIRE
Ballet is boring.	Wrong, boy!... It's far more interesting than you think!...	<ul style="list-style-type: none">• These bodies in motion are so exquisite!... Such beautiful harmony!...• Believe me...	It's corny.
SAY	O1P_DESIRE	O1P_VIEUX	O1P_DESIRE
Why are you called the Old Man?	Don't you have a first name?	I am so old I cannot remember my name.	I don't believe you. Nob

LES PROPRIÉTÉS DE LA BOÎTE CHOICE

Choice	
ID	Identifiant unique de la boîte utilisé par les fonctions.
Tags	Vous pouvez définir jusqu'à quatre tags par choix. Ils vous permettent de les regrouper par catégorie. Un cinquième tag est modifiable par la fonction set_sentence_tag .
Visible	Permet de changer la visibilité du choix au lancement d'une nouvelle partie.
Do not say	Par défaut les choix sont prononcés par l'objet parlant.

Never hide	Par défaut les choix sont retirés après sélection.
Locked	Par défaut le joueur peut quitter une conversation au moment d'un choix. En activant cette option, le joueur sera obligé de choisir une réplique avant de pouvoir interrompre la conversation au prochain choix.
Randomly	Lorsque la boîte contient plusieurs choix séparés d'une ligne vide, l'option permet d'en choisir un aléatoirement.
Mutual show	Permet de rendre visible les boîtes indiquées par leur identifiant après la réplique. Le lien est alors représenté dans l'éditeur par une ligne lorsqu'une des boîtes ciblées est sélectionnée. Les identifiants sont séparés par un espace.
Hide branch	Permet de cacher la branche entière après la réplique.
Go to	Permet d'atteindre une autre boîte après la réplique.
Exit	Permet de quitter la conversation après la réplique.
Content	
Icon	Permet d'afficher l'icône courante d'un objet à la place d'un texte.
Object	
Direction	Permet de changer la direction de l' objet parlant durant la réplique.
Look at	Permet d'orienter l' objet parlant vers un autre objet durant la réplique.
Keep new direction	Permet de conserver la nouvelle direction après la réplique.
Animation	Permet de changer l'animation de l' objet parlant durant la réplique.
Keep new animation	Permet de conserver la nouvelle animation après la réplique.
Puzzle	
Success on say	Permet de résoudre l'énigme spécifiée après avoir lancé la réplique pour la première fois.

Success on show	Permet de résoudre l'énigme spécifiée lorsque le choix apparaît pour la première fois.
Show on success	Permet de rendre visible la réplique après la résolution de l'énigme spécifiée.

LES ACTIONS DE LA BOÎTE CHOICE

Dans l'éditeur de **rôle**, certaines actions synchronisées sont spécifiques aux choix que nous listons brièvement ci-dessous :

show	Permet de changer la visibilité d'un choix au lieu de passer par les fonctions show_choice et hide_choice .
-------------	---

LES CONDITIONS DE LA BOÎTE CHOICE

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux choix que nous listons brièvement ci-dessous :

dialog has	Permet de savoir si le choix est visible.
-------------------	---

LES ÉVÉNEMENTS DE LA BOÎTE CHOICE

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux choix que nous listons brièvement ci-dessous :

on say	Permet de recevoir une notification lorsque le choix est affiché. La propriété Said permet de distinguer le premier affichage. La propriété Sub-choice est ignorée pour les choix.
---------------	--

Boîte bleue « Sentence »

Les boîtes bleues permettent de donner la réplique à un personnage présent dans la **scène**.



LES PROPRIÉTÉS DE LA BOÎTE SENTENCE

Sentence	
ID	Identifiant unique de la boîte utilisé par les fonctions.
Tags	Vous pouvez définir jusqu'à quatre tags par réplique. Ils vous permettent de les regrouper par catégorie. Un cinquième tag est modifiable par la fonction set_sentence_tag .
Entry point	Permet de lancer la réplique si tous les choix à la racine sont épuisés au lancement de la conversation. S'il existe plusieurs points d'entrée alors une seule réplique sera tirée au sort.
Duration	Permet de forcer la durée de la réplique pour des raisons de synchronisation avec l'audio ou pour une action par exemple. La propriété est ignorée si la valeur est à 0.
Randomly	Lorsque la boîte contient plusieurs répliques séparées d'une ligne vide, l'option permet d'en choisir une aléatoirement.
Mutual show	Permet de rendre visibles les boîtes indiquées par leur identifiant après la réplique. Le lien est alors représenté dans l'éditeur par une ligne lorsqu'une des boîtes ciblées est sélectionnée. Les identifiants sont séparés par un espace.
Hide branch	Permet de cacher la branche entière après la réplique.
Go to	Permet d'atteindre une autre boîte après la réplique.

Exit	Permet de quitter la conversation après la réplique.
Object	
Speaker	L' objet parlant. VOICE-OVER permet d'afficher un dialogue sous la forme d'un sous-titre sans avoir besoin d'ajouter un objet dans la scène. La fonction set_voice_over_color servira à personnaliser la couleur du texte avant ou pendant la conversation.
Direction	Permet de changer la direction de l' objet parlant durant la réplique.
Look at	Permet d'orienter l' objet parlant vers un autre objet durant la réplique.
Keep new direction	Permet de conserver la nouvelle direction après la réplique.
Animation	Permet de changer l'animation de l' objet parlant durant la réplique.
Keep new animation	Permet de conserver la nouvelle animation après la réplique.
Puzzle	
Success on say	Permet de résoudre l'énigme spécifiée après avoir lancé la réplique pour la première fois.

LES CONDITIONS DE LA BOÎTE SENTENCE

Dans l'éditeur de **rôle**, certaines conditions sont spécifiques aux choix que nous listons brièvement ci-dessous :

dialog has	Permet de savoir si la réplique a déjà été affichée.
-------------------	--

LES ÉVÉNEMENTS DE LA BOÎTE SENTENCE

Dans l'éditeur de **rôle**, certains événements sont spécifiques aux choix que nous listons brièvement ci-dessous :

on say	Permet de recevoir une notification lorsque la réplique est affichée. La propriété Said permet de distinguer le
---------------	--

	premier affichage et la propriété Sub-choice de comparer l'index du paragraphe choisi par le joueur.
--	---

Interface

Vous pouvez déplacer une boîte avec ou sans ses enfants en la sélectionnant et en la glissant vers une autre boîte. Contrôlez l'emplacement grâce aux touches **CTRL** et **SHIFT**.



CTRL

SHIFT

CTRL + SHIFT

Le comportement par défaut positionne la branche sélectionnée avant une autre branche. D'autres comportements sont accessibles en vous servant du clavier. Avant de relâcher le bouton gauche de la souris lors du déplacement d'une branche ou d'une boîte, restez appuyé sur :

- la touche **CTRL** pour déplacer la branche sélectionnée à l'intérieur d'une autre branche.
- la touche **SHIFT** pour positionner la boîte sélectionnée avant une autre boîte.
- les touches **CTRL** et **SHIFT** pour déplacer la boîte sélectionnée à l'intérieur d'une autre branche.

Il va de même pour la copie vers le presse-papier via le menu popup. Les données au format **JSON** seront accessibles par tous les assets **Dialog**, y compris par les autres instances du logiciel. Notez que les identifiants seront modifiés.



BRANCH

CTRL + C



BOX

CTRL + SHIFT + C



BRANCH

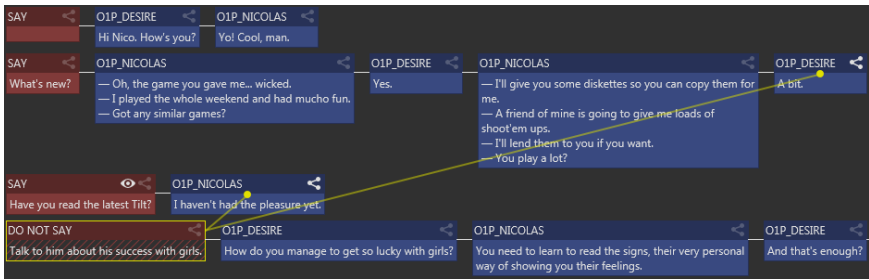
CTRL + X



BOX

CTRL + SHIFT + X

La propriété **Mutual show** possède un raccourci très pratique pour débloquer rapidement des choix après l'affichage d'une ou plusieurs répliques. Sélectionnez d'abord le choix à faire apparaître et cliquez sur l'icône en haut à droite de la boîte qui déblocuera le choix. Au moment du clic, la propriété **Visible** du choix (boîte sélectionnée) passera automatiquement à **False**.



Côté runtime, toutes les boîtes doivent être lues pour que le choix s'affiche à l'écran. Vous n'avez donc aucune ligne de code à écrire pour proposer de nouveaux choix au joueur au fur et à mesure de la conversation si vous procédez ainsi.

Style

seccia.dev introduit quatre modèles de mise en forme qui vous aideront à mieux définir le style de vos dialogues. Encore une fois, il faut bien faire la distinction entre les répliques des personnages et les choix proposés au joueur. Les répliques sont utiles pour faire avancer l'histoire, les choix pour renforcer le gameplay.

LES RÉPLIQUES

En ce qui concerne les répliques, quatre styles sont disponibles : le style **POINT & CLICK**, le style **COMICS**, le style **MOVIE** et le style **RPG**. Quel que soit le style employé, il est possible d'appliquer les effets **Typewriter** et **Vibration** au texte depuis les propriétés du projet. Ces deux effets agissent sur l'ensemble du jeu.



Le style **POINT & CLICK** proposé par **seccia.dev** est inspiré des jeux de **LucasArts** où les répliques sont toujours placées au-dessus du personnage qui prend la parole. La position et l'alignement du texte sont gérés automatiquement par le logiciel. Il s'agit du style par défaut. Il est important d'attribuer une animation **TALK** au personnage pour attirer l'attention du joueur. Pour mieux distinguer les prises de parole, vous pouvez aussi attribuer une couleur par personnage ou peut-être deux ou trois couleurs au maximum pour éviter l'effet *sapin de Noël*. Une couleur pour le héros, une seconde couleur pour les autres protagonistes et éventuellement une troisième couleur pour les rôles secondaires.



Le style **COMICS** dessine une bulle personnalisable à chaque réplique. Vous devez fournir une image **BALLOON** de **128x128** pixels avec ou sans transparence à placer dans le dossier **IMAGE**. La taille de la bulle s'ajustera par rapport au texte. Pour cela vous devez spécifier des coordonnées supplémentaires dans les propriétés du projet. **seccia.dev** propose un modèle gratuit avec ses marges prédéfinies à récupérer dans le dossier **Resources** du logiciel.



Le style **MOVIE** positionne le texte en bas de l'écran comme des sous-titres de film. Il suffit d'activer la propriété **Movie style** du personnage dans l'éditeur

d'**objet**. Ainsi vous pouvez combiner les deux styles lors d'une même discussion selon le personnage actif.



Enfin le style **RPG** ajoute un avatar durant la conversation. La position du texte dépendra de la propriété **Text layout** de l'avatar. Pour mettre en place ce style, le préalable nécessaire est de créer un nouvel **objet** contenant une animation **STOP** et éventuellement une animation **TALK**, pour pouvoir ensuite sélectionner cet **objet** dans la propriété **Avatar** du personnage via l'éditeur d'**objet**.

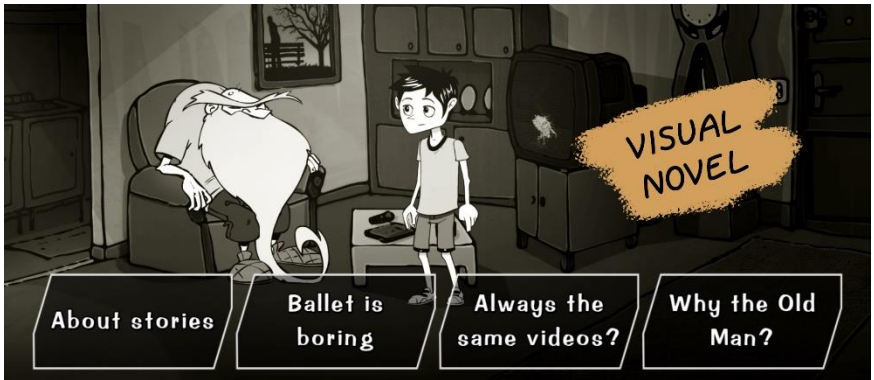
L'animation **TALK** est jouée quand le personnage conserve la parole et l'animation **STOP** lorsque le joueur est amené à choisir une réponse. L'animation **STOP** est utilisée à la place de l'animation **TALK** si cette dernière est absente. Si l'animation **STOP** ne contient pas d'images, aucun avatar ne sera affiché.

LES CHOIX

En ce qui concerne les choix de répliques, deux styles sont disponibles : le style **POINT & CLICK** et le style **VISUAL NOVEL**. Quel que soit le style employé, il est possible d'appliquer l'effet **Vibration** au texte depuis les propriétés du projet.



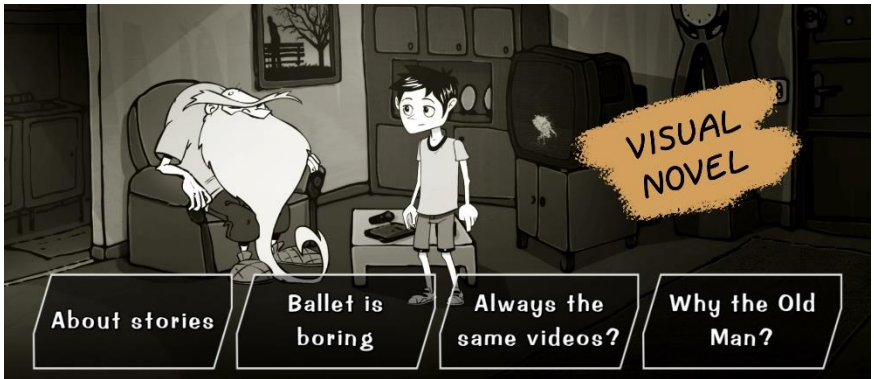
Le style **POINT & CLICK** affiche les choix par ligne en bas de l'écran. Il s'agit du style par défaut et reprend les codes des jeux **Point & Click** traditionnels. Si la phrase est trop longue, elle défilera horizontalement au passage du curseur de la souris. Ce mode est idéal pour développer son propos.



Le style **VISUAL NOVEL** présente les choix sous forme de cases personnalisables. Ce mode plus graphique est mieux adapté pour proposer des choix courts au joueur comme des sujets de discussion en quelques mots.

La première étape consiste à dessiner la case graphique dans un fichier **PNG** unique. La taille de l'image doit mesurer **256x256** pixels avec ou sans transparence. Si votre cadre est rectangulaire, il suffit de laisser un espace vide au-dessus sans redimensionner le fichier. Les images sont centrées horizontalement et disposées en bas de l'écran. Le fichier doit être nommé **CHOICE** et placé dans le dossier **IMAGE**.

La seconde étape consiste à paramétrer le style via les propriétés **Choice** du projet dans le groupe **Text**.



Les bulles peuvent contenir une icône à la place du texte en spécifiant le nom de l'**objet** dans la propriété **Icon** du choix.

Intégration par le code

Pour démarrer une conversation, il suffit d'ajouter une boîte **talk** en spécifiant l'asset **Dialog** et éventuellement l'identifiant de la réplique de début et la réplique de fin. L'ajout d'une boîte **stop** ou l'appel de la fonction **shutup** interrompra prématurément la conversation

```
shutup
```

Trois autres fonctions qui vous seront très utiles pour contrôler la visibilité des choix.

```
show_choice D_BOB 5
hide_choice D_BOB 8
if choice D_BOB 8
end
```

La fonction **said** permet de savoir si une réplique a déjà été dite comme son nom l'indique.

```
if said D_BOB 8
end
```

Si une boîte rouge contient plusieurs répliques séparées par une ligne vide et que vous souhaitez connaître le choix du joueur, vous pouvez utiliser la variable **\$_choice** dans le script de la boîte **on say** de cette façon :

```
if var _choice 0
  // first choice
else_if var _choice 1
  // second choice
end
```

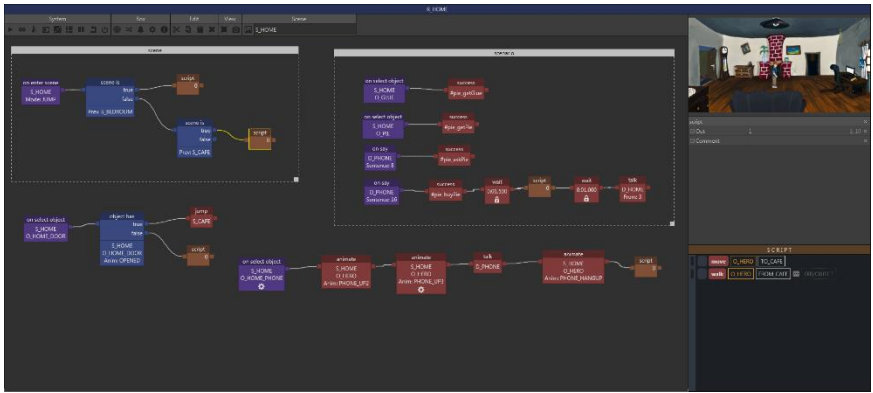
SAY	O2P_KEVIN
The team had a rash?	A bit vague.
They lost their balls at the last minute.	Were these wimps afraid of flying?
The rules forbade teams from playing barefoot.	They pulled rather than break with tradition.
They were only interested in qualifying.	Nope.

Cette variable sera accessible dans l'événement **on say** de la réplique.

RÔLE

L'éditeur de **rôle** permet de programmer visuellement les actions synchronisées, conditions et événements de votre jeu de façon plus intuitive que par les lignes de code.

Il est possible et conseillé d'associer un **rôle** à une **scène** pour interagir entre leur éditeur respectif. De plus, au runtime le **rôle** associé sera automatiquement démarré au chargement de la **scène**, et arrêté à sa fermeture.



Langage

L'éditeur de **rôle**, propose cinq types de boîte : les boîtes **system** en vert, les boîtes **action** en rouge, les boîtes **condition** en bleu, les boîte **event** en violet et les scripts en marron.

Les connexions ont les mêmes propriétés que dans l'éditeur de scénario. Si une boîte accepte plusieurs connexions en entrée, tous les signaux doivent être envoyés pour déclencher l'exécution de la boîte.

START

La boîte **start** est appelée à lancement du **rôle**. Il est possible de définir l'ordre d'appel des boîtes **start** en changeant la valeur du paramètre **Execution order**. Les boîtes seront triées par ordre croissant.

UPDATE

La boîte **update** est appelée une fois par frame de l'application (environ 60 fois par seconde pour une application tournant à 60 fps). Vous ne pouvez pas contrôler la fréquence de rafraîchissement. Veuillez utiliser soit la boîte **on repeat** pour créer des intervalles basés sur le temps soit la boîte **wait** pour insérer une pause en millisecondes. Il est possible de définir l'ordre d'appel des boîtes **update** en changeant la valeur du paramètre **Execution order**. Les boîtes seront triées par ordre croissant. Au lancement d'un **rôle**, elles seront toujours appelées après les boîtes **start**.

TASK

La boîte **task** est appelée par la fonction **task** et permet de faire le pont entre les scripts et les **rôles**. Vous pouvez vous en servir comme des routines pour réutiliser du code sans le dupliquer.

OR

La boîte **or** permet de mettre en place une condition logique. Si une entrée possède plusieurs connexions, le premier signal reçu suffira pour exécuter la boîte. La sortie n'a aucune particularité.

RANDOM

La boîte **random** permet d'émettre un seul signal au hasard parmi les connexions en sortie. L'entrée n'a aucune particularité.

SCHEDULE

La boîte **schedule** permet de planifier l'ordre des sorties. Le nombre de sorties est modifiable et ne peut être inférieur à 2. Rien ne vous empêche de connecter plusieurs câbles sur une même sortie pour combiner les deux méthodes. Il n'est pas obligatoire d'utiliser toutes les sorties.

SKIP

La boîte **skip** permet de sauter une frame du programme. Elle fonctionne comme une pause d'une durée d'une seule frame. Parfois il peut être utile d'exécuter des actions à la frame suivante.

RESTART

La boîte **restart** permet de relancer le **rôle** en réinitialisant tous les états.

END

La boîte **end** permet d'arrêter le **rôle**.

SCRIPT

La boîte **script** permet d'exécuter du code et de choisir une sortie à l'aide de la fonction **out**. Par défaut, la première sortie est prise en compte. La fonction **out** permet de spécifier plusieurs sorties si besoin. La boîte peut retourner une valeur différente de **0** afin de maintenir l'exécution et d'agir comme une boucle. Ainsi, le script sera relancé à la frame suivante sans transmettre de signaux.

Actions

Dans l'éditeur de **rôle**, l'exécution d'une action est synchronisée contrairement aux scripts. Cela signifie que le signal sort de la boîte lorsque l'action est considérée comme terminée. La notion « action terminée » dépend de l'action et de ses paramètres.

Une action ne peut avoir qu'une seule entrée et qu'une seule sortie. Elle possède un script permettant d'appeler d'autres fonctions par le code. Ce script est exécuté dans trois circonstances : lorsque le signal entre dans la boîte (enter), lorsque le signal persiste sur plusieurs frames (execute) et lorsque le signal sort (exit). Il s'agit du même script et pour pouvoir différencier ces trois cas possibles, vous devez utiliser les fonctions **enter**, **execute** et **exit** directement dans le code.

```
if enter
end
if execute
end
if exit
end
```

Il faut bien comprendre que le script n'est appelé qu'une seule fois par frame quelle que soit la situation. Si les trois fonctions retournent **true**, cela signifie que le signal entre et ressort immédiatement.

Il est possible de quitter prématurément une action en renvoyant une valeur différente de **0**. Si le signal est déjà supposé sortir de la boîte, la valeur retournée n'aura pas d'impact.

```
return 1
```

Les actions ont un paramètre en commun : **Lock game**. Ce paramètre permet de bloquer les interactions avec le joueur tant que le signal n’est pas sorti de la boîte. La fonction **lock** est indépendante et peut se combiner.

ANIMATE

La boîte **animate** permet de jouer l’animation d’un **objet**.

Scene	La scène où se trouve l’ objet à animer. Si le champ est vide, l’ objet doit exister dans la scène en cours.
Object	L’ objet à animer.
Animation	Le nom de l’animation à jouer.
Direction	Le nom de la direction. Si la direction n’est pas spécifiée et si elle existe pour cette animation, la direction en cours sera conservée.
Reverse	Permet de jouer l’animation à l’envers. Par défaut, la valeur définie dans l’éditeur d’ objet sera prise en compte.
Start frame	L’index de la frame à laquelle l’animation doit débiter. Par défaut, la valeur définie dans l’éditeur d’ objet sera prise en compte.
Action frame	L’index de la frame à laquelle une notification sera envoyée. Par défaut, la valeur définie dans l’éditeur d’ objet sera prise en compte.

CAMERA

La boîte **camera** permet de déplacer la caméra d’une position à une autre sans utiliser les shots et d’appliquer un zoom à la transition si nécessaire.

Scene	Le nom de la scène de la caméra. Si le champ est vide, les éléments indiqués doivent exister dans la scène en cours.
From shot	Le nom du shot faisant référence à la position de départ.
From object	Le nom de l’ objet faisant référence à la position de départ.

From cell	Le nom de la cell de l' objet faisant référence à la position de départ.
From zoom	La valeur du zoom de départ comprise entre 100 et 400 .
To shot	Le nom du shot faisant référence à la position de fin.
To object	Le nom de l' objet faisant référence à la position de fin.
To cell	Le nom de la cell de l' objet faisant référence à la position de fin.
To zoom	La valeur du zoom de fin comprise entre 100 et 400 .
To offset X/Y	Permet d'appliquer un décalage horizontalement et/ou verticalement en pixels sur la position de fin qui sera maintenue après la transition.
Axis	Permet d'inclure ou d'exclure les axes. Par défaut, les deux axes sont pris en compte.
Fade out	Permet d'appliquer un fondu de sortie au début de la transition.
Fade in	Permet d'appliquer un fondu d'entrée à la fin de la transition.
Duration	La durée en millisecondes de la transition.
Endless	Permet de maintenir l'état de la caméra avec les valeurs de fin de transition.

CINEMATIC

La boîte **cinematic** permet de lancer une vidéo par l'intermédiaire d'un asset **cinematic**. Le signal quitte la boîte lorsque la vidéo est arrêtée.

Cinematic	Nom de l'asset cinematic à jouer.
------------------	--

EXAMINE

La boîte **examine** permet d'afficher en grand un **objet** pour laisser la possibilité au joueur d'observer un détail important du jeu. Durant l'observation, la partie reste bloquée tant que le joueur ne quitte pas l'écran. Le signal quitte la boîte lorsque le joueur reprend la main.

Object	Nom de l' objet à examiner.
Animation	Nom de l'animation de l' objet à jouer en boucle.
Direction	Nom de la direction de l'animation.
Scale	La taille de l' objet en pourcentage par rapport à la taille de l'écran.
Opacity	L'opacité de l' objet en pourcentage. La valeur 0 rend l'objet invisible.

INTERPOLATE

La boîte **interpolate** permet d'interpoler un entier entre une valeur minimale et une valeur maximale, et de récupérer le résultat dans une variable.

Duration	La durée de l'interpolation en millisecondes.
Min/Max	La valeur minimale et la valeur maximale de l'interpolation.
Loop count	Le nombre de boucles à effectuer.
Loop pause	La pause en millisecondes entre deux boucles.
Reverse	Permet d'inverser l'interpolation à chaque boucle.
Ease IN/OUT	Ajouter une accélération au début et/ou une décélération à la fin.
Variable	Le nom de la variable qui contiendra le résultat de l'interpolation à chaque frame.

JUMP

La boîte **jump** permet de changer de **scène** en appliquant une transition de fondu au noir. Le signal quitte la boîte lorsque l'écran devient noir.

Scene	Le nom de la nouvelle scène à charger. Sélectionnez BACK pour recharger la scène précédente si elle existe (l'équivalent de la fonction jump_back).
Fade out duration	Durée en millisecondes de la transition du fondu de sortie.

Pause	Durée en millisecondes de la pause avant le changement de scène .
Fade in duration	Durée en millisecondes de la transition du fondu d'entrée.

LIGHT

La boîte **light** permet de changer les propriétés d'une lumière attachée à un **objet**.

Scene	Le nom de la scène de la lumière. Si le champ est vide, l' objet doit exister dans la scène en cours.
Object	Nom de l' objet où se trouve la lumière.
Visible	Change la visibilité de la lumière pour l'activer ou la désactiver.
Ambient	Change la valeur ambiante de la lumière entre 0 et 255 . Cette valeur s'additionne à la valeur ambiante actuelle de la scène. Pour obtenir un effet, il est donc nécessaire de réduire la valeur ambiante soit de la scène soit du jeu.
Diffuse	Change la couleur diffuse de la lumière au format RGBA . Pour ignorer ce champ, utilisez la valeur 0 pour la composante alpha.
Angle	Change l'amplitude de l'angle entre 0 et 360 degrés. Si l'angle est à 360 , la lumière est omnidirectionnelle.
Direction	Change la direction de la lumière en spécifiant un angle entre 0 et 359 degrés.
Distance	Change la distance en pixels de la lumière. Si la valeur est -1 , la lumière éclaire à l'infini.
Attenuation	Change la méthode d'atténuation de la lumière. La valeur est une puissance entre 0 et 10 . Si la valeur est 0 , aucune atténuation est appliquée. Si la valeur est 1 , une atténuation linéaire est appliquée. Si la valeur est 2 , une atténuation au carré est appliquée.

PATH

La boîte **path** permet de démarrer le **path** d'un **objet**.

Scene	Le nom de la scène de l' objet . Si le champ est vide, l' objet doit exister dans la scène en cours.
Object	Le nom de l'objet où se trouve le path .
Path	L'index du path à démarrer.
Spot	Le nom du spot où doit commencer le déplacement. Si le champ est vide, la premier spot est utilisé.
Loop count	Le nombre de boucles à effectuer. Si le champ est vide, la valeur définie dans les propriétés du path est prise en compte.

PLAYER

La boîte **player** permet modifier les attributs d'un **player**.

Player	Le nom du player à modifier.
Scene	Le nom de la scène utilisée pour le changement de personnage (l'équivalent de la fonction set_player_scene).
Object	Le nom de l' objet contrôlé par le player . (l'équivalent de la fonction control).
Switch	Permet de changer le player en cours si la valeur est true (l'équivalent de la fonction switch).
Empty	Permet d'enlever tous les objets de l'inventaire (l'équivalent de la fonction empty).
Transfer to	Permet de déplacer tous les objets de l'inventaire vers un autre player (l'équivalent de la fonction transfer).

POPUP

La boîte **popup** permet d'afficher un message à l'écran avec un choix de réponse. En fonction du type de la fenêtre, plusieurs sorties sont possibles.

Title	Le titre de la fenêtre.
--------------	-------------------------

Message	Le message de la fenêtre.
Type	Par défaut, la fenêtre affiche un bouton OK . Il est possible d'afficher deux boutons Yes/No en choisissant Question .

ROLE

La boîte **role** permet de démarrer un **rôle**. Le signal sera bloqué tant que le **rôle** lancé sera en cours d'exécution.

Role	Nom du rôle à démarrer.
-------------	--------------------------------

SELECT

La boîte **select** permet de reproduire le clic sur un **objet**.

Player	Le nom du player qui doit correspondre au player en cours.
Scene	Le nom de la scène qui doit correspondre à la scène en cours.
Object	Le nom de l'objet à sélectionner.
Sub-object	Le nom du sous-objet à sélectionner.

SHOW

La boîte **show** permet de changer la visibilité d'une ou plusieurs entités. Le signal sort immédiatement après avoir changé l'état des éléments.

Visible	Permet de rendre visible ou invisible les entités indiquées.
Duration	Durée de la transition du changement d'état si applicable.
Brightness	Permet de changer la luminosité de la scène entre 0 (noir) et 100 .
Scene	Le nom de la scène concernant les entités indiquées. Si le champ est vide, les entités doivent exister dans la scène en cours.

Layer	Le nom du layer dont la visibilité sera modifiée.
Still	Le nom du still dont la visibilité sera modifiée.
Object	Le nom de l' objet dont la visibilité sera modifiée.
Light	Le nom de la lumière via l' objet dont la visibilité sera modifiée.
Label	Le nom du label dont la visibilité sera modifiée.
Door	Le nom de la door dont la visibilité sera modifiée.
Cursor	Permet de changer la visibilité du curseur. Cette propriété ne tient pas compte de la propriété Visible .
Bag	Permet de changer la visibilité de l'inventaire.
Menu	Permet d'ouvrir ou fermer le menu du jeu. Cette propriété ne tient pas compte de la propriété Visible .
Credits	Permet d'afficher les crédits ou de retourner au jeu. Cette propriété ne tient pas compte de la propriété Visible .
Dialog	Le nom du dialogue où se trouve le choix à modifier.
Choice	L'identifiant du choix dont la visibilité sera modifiée.

SONG

La boîte **song** permet de jouer une musique sur un canal. Le signal quitte la boîte immédiatement même en cas de transition.

Song	Le nom du fichier de la musique sans l'extension.
Channel	L'index du canal entre 0 et 3 . La valeur -1 agit sur tous les canaux, si applicable.
Volume	Le volume de la musique entre 0 et 100 .
Loop	Lecture unique ou en boucle.
Crossfade	La durée en millisecondes de la transition entre la musique en cours de lecture et la nouvelle musique. Les deux musiques doivent utiliser le même canal.
Last song	Permet de rejouer la précédente musique.

STOP

La boîte **stop** permet d'arrêter les entités indiquées.

Object	Le nom de l' objet à arrêter (l'équivalent de la fonction stop).
Role	Le nom du rôle à interrompre (l'équivalent de la fonction stop_role).
Camera	Permet de réinitialiser les mouvements de caméra.
Dialog	Permet d'interrompre la conversion en cours.
Timeline	Permet d'interrompre la timeline .
Cinematic	Permet d'interrompre la cinématique en cours de lecture.
Song	Permet d'arrêter la musique en cours de lecture sur un canal. Si le canal est -1 , toutes les musiques seront interrompues.

SUCCESS

La boîte **success** permet de valider une énigme. C'est l'équivalent de la fonction **success** par le code.

Puzzle	Le nom de l'énigme à valider.
---------------	-------------------------------

TAKE

La boîte **take** permet de récupérer un objet pour le placer dans un inventaire ou dans la corbeille.

Player	Le nom du player qui recevra l' objet . Si le paramètre n'est pas renseigné, le player en cours sera utilisé.
Objet	Le nom de l' objet à récupérer.
Silent	Permet d'activer ou de désactiver l'effet visuel.
Replace	Le nom de l' objet à remplacer s'il ne s'agit pas d'un ajout.

Animation	Le nom de l'animation du player à jouer avant de déclencher l'action.
Direction	Le nom de la direction de l'animation.
Action frame	L'index de la frame de l'animation utilisée pour déclencher l'action. -1 correspond à la fin de l'animation. Si le paramètre n'est pas renseigné, l'index de l'éditeur d' objet sera utilisé.

TALK

La boîte **talk** permet de lancer une nouvelle conversation. Le signal quitte la boîte lorsque la conversation se termine.

Dialog	Le nom du dialogue à démarrer.
Sentence (start)	L'identifiant de la réplique du début de la conversation.
Sentence (end)	L'identifiant de la réplique qui interrompra la conversation. Il est possible que le joueur quitte la conversation avant d'atteindre cette réplique.

TIMELINE

La boîte **timeline** permet de lancer une **timeline**.

Scene	Le nom de la scène de la timeline . Si le champ est vide, la timeline doit exister dans la scène en cours.
Timeline	Le nom de la timeline à démarrer.

WAIT

La boîte **wait** permet d'imposer un délai d'attente en millisecondes avant de faire sortir le signal.

Duration	La durée en millisecondes de l'attente.
-----------------	---

WALK

La boîte **walk** permet de déplacer un **objet** de **scène** en calculer un chemin entre la position actuelle et la destination. C'est l'équivalent de la fonction **walk**. Le signal quitte la boîte lorsque l'**objet** a atteint sa cible.

Scene	Le nom de la scène de l' objet . Si le champ est vide, l' objet doit exister dans la scène en cours.
Object	Le nom de l' objet à déplacer.
From cell	Le nom de la cell de l' objet faisant référence à la position initiale. Ce paramètre ne peut pas se combiner avec les propriétés X et Y .
From x/y	La coordonnée de la position initiale. Ce paramètre ne peut pas se combiner avec la propriété Cell .
Cell	Le nom de la cell de l' objet faisant référence à la position de destination. Ce paramètre ne peut pas se combiner avec les propriétés X et Y .
To x/y	La coordonnée de la position à atteindre. Ce paramètre ne peut pas se combiner avec la propriété Cell .
Animation	Force la lecture d'une animation pendant le déplacement.
Direction	Force une direction pendant le déplacement.
Straight	Déplace l' objet en suivant une ligne droite de sa position initiale à la destination sans tenir compte de la vitesse de marche.
Straight duration	La durée du déplacement en ligne droite en millisecondes.
Straight ease IN/OUT	Permet d'appliquer une accélération au début et/ou une décélération vers la fin du déplacement en ligne droite.

Conditions

Les conditions permettent d'enrichir les séquences d'actions synchronisées sans passer par le code, tout en gardant la logique de la programmation visuelle. Elles

possèdent une entrée et plusieurs sorties spécifiques. Si la sortie déterminée par le résultat de la condition n'est pas câblée, le signal sera alors perdu et sa progression interrompue.

Une condition peut prendre la forme d'une boucle si l'option **Loop box** est activée. Cependant, la boucle cessera de tourner si la sortie appelée est connectée à une autre boîte. « Tant que la condition est fausse, la séquence doit être bloquée » nécessiterait de brancher un câble uniquement à la première sortie : le cas où la condition est vraie.

Les conditions les plus fréquentes sont accessibles via l'éditeur de **rôles** mais il est tout à fait possible d'appeler n'importe quelle fonction du langage par l'intermédiaire de la boîte **result is** qui retournera un booléen, c'est-à-dire le résultat de la fonction elle-même.

BOOLEAN IS

Cette condition permet de comparer la variable en tant que booléen. Le résultat ne peut donc être seulement vrai ou faux.

Variable	Le nom de la variable à tester.
-----------------	---------------------------------

DIALOG HAS

Cette condition permet de connaître les états d'un **dialogue**. La condition sera vraie si tous les paramètres fournis coïncident.

Dialog	Le nom du dialogue à tester.
Choice	L'identifiant du choix qui devra être visible.
Said	L'identifiant de la réplique qui devra être affichée au moins une fois.

DIALOG IS

Cette condition permet de connaître le **dialogue** et la réplique en cours d'exécution. La condition sera vraie si tous les paramètres fournis coïncident.

Dialog	Le nom du dialogue .
Sentence	L'identifiant de la réplique.

OBJECT HAS

Cette condition permet de connaître les états d'un **objet**. La condition sera vraie si tous les paramètres fournis coïncident.

Scene	Le nom de la scène en cours d'exécution.
Object	Le nom de l' objet se trouvant dans la scène .
Animation	Le nom de l'animation en cours de lecture.
Direction	La direction de l'animation.
Frame	L'index de la frame de l'animation.
Cell	Le nom de la cell où se trouve l' objet .
Flag	La valeur du flag de la cell où se trouve réellement l' objet . Ce paramètre ne tient pas compte de la propriété Cell .
Sticker	Le nom du sticker de l' objet .
Visible	La visibilité de l' objet .

PLAYER HAS

Cette condition permet de connaître les états d'un **player**. La condition sera vraie si tous les paramètres fournis coïncident.

Player	Le nom du player à tester.
Object	Le nom de l' objet se trouvant dans l'inventaire du player .

PLAYER IS

Cette condition permet de connaître le **player** en cours et l'**objet** contrôlé par le **player**. La condition sera vraie si tous les paramètres fournis coïncident.

Player	Le nom du player .
Object	Le nom de l' objet contrôlé par le player actuel. Ce paramètre ne tient pas compte de la propriété Player .

PUZZLE IS

Cette condition permet de connaître l'état d'une énigme et possède quatre sorties :

resolved : l'énigme a été résolue par le joueur.

resolving : l'énigme est en cours de résolution (le signal est entré dans la boîte mais n'est pas encore sorti). Cet état considère que l'énigme n'est pas encore résolue.

unresolved : l'énigme n'est pas résolue (le signal n'est jamais sorti de la boîte). Cet état peut être combiné avec l'état **resolving**.

catch-all : c'est le cas par défaut si le signal n'a pas pu sortir par les autres prises.

Lorsqu'une boîte **or** exclue un puzzle, ce dernier redevient exclusivement une énigme non résolue.

Puzzle	Le nom de l'énigme.
---------------	---------------------

RESULT IS

Cette condition permet d'appeler une fonction du langage et de comparer la valeur retournée de type booléen. Il ne faut pas utiliser le mot clé **return**.

SCENE HAS

Cette condition permet de connaître les états d'une **scène**. La condition sera vraie si tous les paramètres fournis coïncident.

Scene	Le nom de la scène à tester. Si le paramètre n'est pas renseigné, la scène en cours sera utilisée.
Still	Le nom du still se trouvant dans la scène et devant être visible.
Object	Le nom de l' objet se trouvant dans la scène et devant être visible.
Label	Le nom du label se trouvant dans la scène et devant être visible.
Door	Le nom de la door se trouvant dans la scène et devant être visible.

SCENE IS

Cette condition permet de connaître la **scène** en cours d'exécution et la précédente **scène**. La condition sera vraie si tous les paramètres fournis coïncident.

Scene	Le nom de la scène .
Previous scene	Le nom de la précédente scène .

SEQUENCE IS

Cette condition permet de connaître l'état d'une séquence et possèdent six sorties :

started : la séquence a démarré (le signal est déjà entré dans la première boîte de la séquence).

playing : la séquence est encore en cours d'exécution (le signal se trouve entre la première et seconde boîte délimitant la séquence).

ended : la séquence a démarré et s'est terminée (le signal est sorti de la seconde boîte de la séquence).

unstarted : la séquence n'a jamais démarré (le signal n'est pas entré dans la première boîte de la séquence).

catch-all : c'est le cas par défaut si le signal n'a pas pu sortir par les autres prises.

Sequence	Le nom de la séquence faisant référence aux deux boîtes rouges.
-----------------	---

VALUE IS

Cette condition permet de comparer une variable et dévier le signal en fonction de sa valeur. C'est en quelque sorte un **switch** à quatre **case** avec un **default**.

Value1	La valeur de la première sortie.
Value2	La valeur de la deuxième sortie.
Value3	La valeur de la troisième sortie.
Value4	La valeur de la quatrième sortie.
Variable	Le nom de la variable à tester.

Événements

Les événements permettent de recevoir des notifications lorsque le joueur interagit avec les éléments du jeu, et par conséquent de pouvoir réagir en effectuant des actions en retour. Les événements sont spécifiques aux éléments de jeu (*objets, labels, doors...*).

Par défaut lorsqu'un événement est déclenché, une notification est envoyée à toutes les boîtes faisant référence à ce dernier en parcourant les rôles en cours d'exécution. Ainsi, plusieurs boîtes issues du même type d'événement peuvent être appelée à la suite d'une seule et même action.

Afin d'apporter plus de flexibilité aux événements, l'option **Break** met un terme à la propagation du signal. Cette option est très pratique pour gérer des combinaisons différentes : par exemple si vous souhaitez intercepter la combinaison de l'objet A avec l'objet B et la combinaison de l'objet A avec n'importe quel autre objet. Dans ce cas, il vous suffira de créer deux boîtes distinctes du même événement avec des paramètres différents et d'activer l'option **Break** pour la combinaison A/B. Sans cette contrainte, les deux boîtes seront appelées pour la même action.

LES FILTRES

Les filtres permettent de sélectionner en amont les boîtes à appeler lors d'un événement. Certains filtres ne s'appliquent pas à toutes les boîtes. Par défaut, aucun filtre n'est appliqué.

Current player	Permet de filtrer en fonction du player en cours.
Current scene	Permet de filtrer en fonction de la scène en cours.
Resolved puzzle	Permet de filtrer en fonction de l'état d'une énigme qui doit être résolue.
Resolving puzzle	Permet de filtrer en fonction de l'état d'une énigme qui doit être en cours de résolution.
Unresolved puzzle	Permet de filtrer en fonction de l'état d'une énigme qui doit être non résolue.
Started sequence	Permet de filtrer en fonction de l'état d'une séquence qui doit être démarrée.
Playing sequence	Permet de filtrer en fonction de l'état d'une séquence qui doit être en cours d'exécution.

Ended sequence	Permet de filtrer en fonction de l'état d'une séquence qui doit être terminée.
Unstarted sequence	Permet de filtrer en fonction de l'état d'une séquence qui n'a pas encore été démarrée.
Variable	Permet de filtrer en fonction de la valeur d'une variable. Il est nécessaire d'indiquer le nom de la variable, l'opérateur de condition et la valeur à comparer.

ON CLICK

Cet événement permet de recevoir une notification lorsque le joueur clique sur la scène du jeu. La zone de clic est représentée par un rectangle. Par défaut, le rectangle correspond à la taille de la **scène**. En retournant une valeur différente de 0, la marche sera ignorée.

Scene	Nom de la scène .
X/Y	La position du point en haut à gauche du rectangle en pixels.
Width/Height	La taille du rectangle en pixels.

ON DETACH OBJECT

Cet événement permet de recevoir une notification lorsque le joueur décide de séparer deux éléments d'un **objet** de l'inventaire. Pour utiliser cette fonctionnalité, il est nécessaire d'ajouter une image **LAYOUT_DETACH** au projet.

Object	Nom de l' objet présent dans l'inventaire.
---------------	---

ON DRAG OBJECT

Cet événement permet de recevoir une notification lorsque le joueur décide de démarrer un glisser-déposer en sélectionnant un **objet** ayant la propriété **Source** ou **Both**. En retournant une valeur différente de 0, le glisser-déposer est interrompu.

Scene	Le nom de la scène .
--------------	-----------------------------

Object	Le nom de l' objet source qui a initié le glisser-déposer. Depuis le script de la boîte, il est accessible via la variable <code>\$_obj</code> .
Sub-object	Le nom du sous-objet présent dans l' objet . Depuis le script de la boîte, il est accessible via la variable <code>\$_sub</code> .

ON DROP OBJECT

Cet événement permet de recevoir une notification lorsque le joueur dépose l'**objet** sur une zone statique de la **scène**.

Scene	Le nom de la scène .
Object	Le nom de l' objet source qui a initié le glisser-déposer. Depuis le script de la boîte, il est accessible via la variable <code>\$_obj</code> .

ON END SONG

Cet événement permet de recevoir une notification lorsqu'une musique se termine avant de boucler.

Song	Le nom de la musique.
-------------	-----------------------

ON ENTER DIALOG

Cet événement permet de recevoir une notification lorsqu'une conversation démarre.

Dialog	Le nom du dialogue .
---------------	-----------------------------

ON ENTER SCENE

Cet événement permet de recevoir une notification lorsque le joueur entre dans une nouvelle **scène**. En cas de fondu au noir, la notification est reçue juste avant la transition d'ouverture quand l'écran est dans l'obscurité.

Scene	Le nom de la scène .
--------------	-----------------------------

Mode	<p>JUMP : l'événement est appelé en cas de changement de scène.</p> <p>WALK : l'événement est appelé lorsque le personnage atteint la cell TO après être entré dans la scène.</p> <p>SWITCH : l'événement est appelé après un changement de player.</p> <p>LOAD : l'événement est appelé lorsqu'une sauvegarde de partie est restaurée.</p>
Cell	WALK : le nom de la cell TO .

ON EXIT DIALOG

Cet événement permet de recevoir une notification lorsque la conversation se termine.

Dialog	Le nom du dialogue .
---------------	-----------------------------

ON EXIT SCENE

Cet événement permet de recevoir une notification lorsque le joueur quitte la **scène**. En cas de fondu, la notification est reçue juste après la transition.

Scene	Le nom de la scène .
--------------	-----------------------------

ON INPUT

Cet événement permet de recevoir une notification lorsque le joueur clique sur l'écran.

Button	<p>MAIN : le bouton principal (clic gauche pour la souris).</p> <p>ALT : le bouton alternatif (clic droit pour la souris).</p>
---------------	--

ON REACH CELL

Cet événement est appelé lorsqu'un **objet** entre ou sort d'une **cell** de type **Event**. Les **cells** consécutives portant le même nom sont considérées comme un groupe de **cells** ou une **cell** unique. Si l'**objet** entre dans une **cell**, vous pouvez retourner

une valeur différente de 0 pour annuler le dernier déplacement et empêcher l'entrée.

Scene	Le nom de la scène .
Object	Le nom de l' objet présent dans la scène .
Cell	Le nom de la cell présente dans la scène . Le nom par défaut CELL est ignoré.
Mode	IN : entre dans une cell . OUT : sort d'une cell .

ON REACH SPOT

Cet événement permet de recevoir une notification lorsqu'un **objet** atteint un **spot** placé sur un **path**.

Scene	Le nom de la scène .
Object	Le nom de l' objet présent dans la scène .
Spot	Le nom du spot présent dans la scène . Le nom par défaut SPOT est ignoré.

ON REPEAT

Cet événement permet de recevoir une notification périodiquement.

Duration	Durée de l'intervalle en millisecondes.
-----------------	---

ON SAY

Cet événement permet de recevoir une notification lorsque le joueur affiche un choix ou une réplique.

Dialog	Le nom du dialogue .
Sentence	L'identifiant du choix ou de la réplique.
Said	L'état précédent.

Sub-choice	L'index du paragraphe après un choix, uniquement valable pour les répliques.
-------------------	--

ON SELECT DOOR

Cet événement permet de recevoir une notification lorsque le joueur clique sur une **door**.

Scene	Le nom de la scène .
Door	Le nom de la door présente dans la scène .
Mode	<p>WALK : l'événement est appelé lorsque le joueur clique une première fois sur la door pour l'atteindre.</p> <p>SKIP : l'événement est appelé lorsque le joueur clique une seconde fois sur la door pendant le déplacement du personnage. Grâce à cette option vous pourrez donner la possibilité au joueur d'écourter son déplacement.</p>

ON SELECT LABEL

Cet événement permet de recevoir une notification lorsque le joueur clique sur un **label**.

Scene	Le nom de la scène .
Label	Le nom du label présent dans la scène .

ON SELECT LAYOUT

Cet événement permet de recevoir une notification lorsque le joueur clique sur un bouton personnalisable de l'interface.

Button	L'identifiant du bouton.
---------------	--------------------------

ON SELECT OBJECT

Cet événement permet de recevoir une notification lorsque le joueur clique sur un **objet** du décor ou en utilisant la fonction **select**.

Scene	Le nom de la scène .
Object	Le nom de l' objet présent dans la scène .
Sub-object	Le nom du sous-objet présent dans l' objet .
Mode	USER : déclenché après l'action du joueur. CALL : déclenché après l'appel de la fonction select .

ON SHOW CHOICE

Cet événement permet de recevoir une notification lorsque le joueur débloque un choix caché par l'intermédiaire des répliques définies dans l'éditeur de **dialogue**.

Dialog	Le nom du dialogue .
Choice	L'identifiant du choix.

ON SUCCESS

Cet événement permet de recevoir une notification lorsque le joueur résout une énigme.

Puzzle	Le nom de l'énigme.
---------------	---------------------

ON SWITCH PLAYER

Cet événement permet de recevoir une notification lorsque le joueur change le **player** courant.

Player	Le nom du nouveau player .
Previous player	Le nom du précédent player .

ON USE LABEL

Cet événement permet de recevoir une notification lorsque le joueur utilise un **objet** d'inventaire avec un **label**.

Scene	Le nom de la scène .
--------------	-----------------------------

Object	Le nom de l' objet présent dans l'inventaire.
Label	Le nom du label présent dans la scène .

ON USE OBJECT

Cet événement permet de recevoir une notification lorsque le joueur combine un **objet** de l'inventaire avec un **objet** du décor ou un autre **objet** de l'inventaire.

Object A	Le nom du premier objet présent dans l'inventaire.
Object B	Le nom du second objet présent dans la scène ou dans l'inventaire.
Sub-object B	Le nom du sous-objet présent dans le second objet .

ON WALK

Cet événement permet de recevoir une notification lorsque le personnage commence ou termine son déplacement.

Scene	Le nom de la scène .
Object	Le nom de l' objet présent dans la scène .
Mode	ENTER : pour indiquer le début de la marche. EXIT : pour indiquer la fin de la marche.

Intégration par le code

Les **rôles** peuvent être lancés et arrêtés par le code.

```
start_role R_HOUSE
stop_role R_GARDEN
```

Pour communiquer avec l'éditeur de **rôle**, il existe la fonction **task** permettant d'envoyer une tâche. Cette dernière fonctionne comme une routine. Spécifiez d'abord le nom du **rôle** où se trouve la tâche, puis le nom de la tâche et enfin l'argument comme ceci :

```
task R_HOUSE name "argument"
```

Pour rechercher une tâche parmi tous les rôles en cours d'exécution, utiliser le caractère `*`.

```
task * name $city
```

S'il existe plusieurs boîtes `task` portant le même nom, seule la première trouvée sera exécutée. Utiliser le même nom de tâche dans différents rôles peut avoir un intérêt de généralité puisqu'ils ne sont pas forcément tous en route en même temps.

Pour récupérer l'argument dans le script de la boîte `task`, vous devez simplement utiliser la variable `$_arg`. Par défaut, l'argument est un texte vide.

```
alert $_arg
```

Pour retourner une valeur booléenne à la fonction `task`, vous devez modifier la valeur de la variable `$_res` en indiquant `1` ou `true`. Par défaut, la fonction `task` retourne `false`.

```
set _res 1
set _res true
```

Concernant la boîte `task` de l'éditeur de rôle, vous pouvez choisir la sortie en appelant la fonction `out` et en spécifiant un index entre 0 et 9.

```
out 2
```

En retournant une valeur différente de 0, la boîte `task` n'enverra aucuns signaux de sortie.

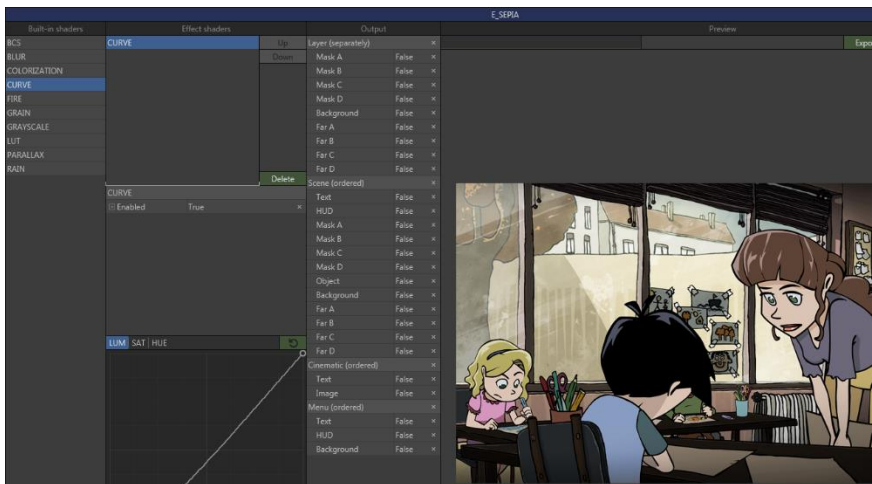
```
return 1
```

Le rôle doit être en cours d'exécution pour recevoir une tâche, sinon la fonction `task` retourne `false`.

EFFET

L'éditeur d'**effet** permet d'appliquer des effets visuels à l'écran en temps réel en sollicitant la carte graphique grâce aux **shaders**. Un **effet** peut être associé à une **scène** entière ou à un layer depuis l'éditeur de **scène** ou à tout le jeu depuis les paramètres du projet.

Un **effet** est constitué d'une liste de **shaders** parce qu'il est possible de les empiler pour créer une succession d'effets visuels. À utiliser avec parcimonie car cela peut diminuer considérablement les performances du jeu en cas d'exagération. Pour un jeu mobile il est recommandé de vérifier régulièrement les performances en testant l'application sur plusieurs modèles d'appareil.



Output

Certains éléments constituant la **scène** sont indépendants tels que le décor principal, les quatre décors éloignés et les quatre masques au premier plan. Ces éléments ne sont que des fichiers **PNG**. En activant ces calques, vous appliquerez l'**effet** uniquement sur les calques en question.

Pour les autres catégories, le fonctionnement est différent et l'**effet** est appliqué sur la totalité de l'écran. L'ordre des paramètres correspond à l'ordre d'affichage des éléments. Concernant la **scène**, l'ordre d'affichage est le suivant :

Far	Les calques D, B, C puis A.
Back	Le décor principal.
Object	Les objets et labels .
Mask	Les masques D, C, B puis A.
HUD	Les éléments d'interface.
Text	Les textes (sauf les labels).
Cursor	Le curseur de la souris.

Il est donc impossible d'appliquer l'**effet** par exemple aux masques sans l'appliquer aux **objets**, au décor principal ou aux décors éloignés.

Brightness, Contrast, Saturation

Cet **effet** permet de contrôler la luminosité, le contraste et la saturation de l'image.

La luminosité est comprise entre **-1** et **1**. Sa valeur par défaut est **0**.

Le contraste est une valeur positive. Sa valeur par défaut est **1**.

La saturation est une valeur positive. Lorsque sa valeur est à **0**, l'image est complètement désaturée. Sa valeur par défaut est **1**.

Blur

Cet **effet** permet d'appliquer un flou gaussien.

L'intensité initiale est comprise entre **0** et **1**. Le max zoom permet d'augmenter l'intensité en fonction du zoom de la **scène** pour simuler un effet de profondeur de champ. Si la valeur est égale à **1**, seule l'intensité initiale sera utilisée. Cette technique est plus adaptée pour les **Visual Novels** lorsque le décor se trouve plus éloigné des personnages.

Colorization

Cet **effet** permet d'appliquer deux teintes à une image en nuance de gris, en se basant sur les hautes et basses lumières. Si l'image est en couleurs, elle est d'abord convertie en noir et blanc par le **shader**.

La valeur **Highlight color** est appliquée aux hautes lumières et la valeur **Shadow color** aux basses lumières. Par défaut les valeurs sont fixées à FFFFFFFF (**RGB**), c'est-à-dire la valeur du blanc.

La valeur **Color smoothless** adoucit la transition entre les deux teintes. Pour désactiver la transition, il suffit de spécifier la valeur **0**.

Curve

Cet **effet** permet de modifier la luminosité, la saturation et la couleur de l'image à l'aide de trois courbes.

Fire

Cet **effet** permet de générer un rendu réaliste de flamme de feu sur toute la surface de l'image avec la possibilité d'ajuster la vitesse d'animation.

La valeur **Fusion** permet de contrôler le mode de fusion. Par défaut, le feu est rendu sur fond noir. En mode **Addition**, il est superposé à l'image de fond. Le mode **Mask** est légèrement plus complexe à mettre en place car le fond doit être une image en nuance de gris, où le noir représente la transparence.

Grain

Cet **effet** permet d'appliquer un grain monochrome dynamique à l'image en tenant compte de l'effet bokeh si besoin.

La valeur **Gain** permet de changer la taille du grain à l'image.

Grayscale

Cet **effet** permet de retirer les couleurs de l'image. Ce **shader** est plus rapide que le **shader** BCS si vous souhaitez seulement convertir l'image en noir et blanc.

Look Up Table

Cet **effet** permet de rajouter un look à votre image. Il vous suffit de choisir un style parmi la liste des filtres disponibles.

Les filtres peuvent également être appliqués directement aux images en amont au moment du build. Dans ce cas, la carte graphique n'a plus besoin d'effectuer un traitement au runtime. L'inconvénient majeur de cette méthode est la perte de qualité puisque chaque image transparente est traitée indépendamment, alors que les **effets** peuvent être appliqués à la **scène** entière. Il est parfois plus judicieux de traiter statiquement les images pour les versions mobiles afin de trouver un bon compromis entre la qualité et les performances de votre jeu.

Parallax

Cet **effet** permet d'appliquer un effet parallaxe en fournissant une version du décor (**DEPTH.png**) contenant la distance de chaque pixel par rapport à la caméra. Le blanc représente les éléments au plus proche de la caméra.

Rain

Cet **effet** permet de générer un rendu réaliste de pluie sur toute la surface de l'image avec la possibilité d'ajuster l'intensité et la teinte des gouttes.

La valeur **Fusion** permet de contrôler le mode de fusion. Par défaut, la pluie est rendue sur fond noir. En mode **Addition**, elle est superposée à l'image de fond. Le mode **Mask** est légèrement plus complexe à mettre en place car le fond doit être une image en nuance de gris, où le noir représente la transparence.

Intégration par les éditeurs

Après avoir indiqué avec quels éléments les **shaders** doivent interagir, vous devez associer votre asset pour qu'il soit visible.

Pour les **scènes**, utilisez la propriété **Effect** du groupe **Scene**. Idem pour les **cinématiques**.

Pour le menu, le réglage se trouve dans les propriétés du projet à la ligne **Effect** du groupe **Menu**.

Vous trouverez une autre ligne **Effect** dans le groupe **Graphics** qui vous permettra d'associer un **effet** à l'ensemble des **scènes** du jeu. Notez que cette propriété sera ignorée si le champ **Effect** d'une **scène** est renseigné. Il s'agit en fait de définir un **effet** global au jeu et d'appliquer ensuite un **effet** spécifique à certaines **scènes**. Cela vous évite de renseigner le champ **Effect** de toutes les **scènes**.

Intégration par le code

Le changement d'**effets** se fait par les fonctions **set_effect** et **set_scene_effect**. La première fonction change l'**effet** du jeu et la seconde fonction change l'**effet** d'une **scène**. Si le paramètre est vide, aucun **effet** ne sera utilisé.

```
set_effect E_NIGHT  
set_scene_effect S_GARDEN E_DAY  
set_effect
```


CINÉMATIQUE

Dans un jeu d'aventure, on a souvent recours à des séquences cinématographiques ou des cutscenes en plein écran pour présenter l'histoire, les personnages, l'environnement ou renforcer l'intrigue à des moments clés de la narration. Ce sont des passages qui apportent beaucoup d'informations supplémentaires au jeu et qui peuvent difficilement être intégrés au gameplay et aux énigmes. Ces séquences servent également à récompenser le joueur après une longue étape d'énigmes résolues pour lui donner un moment de répit, et lui donner envie de poursuivre l'aventure vers un autre objectif.

Il existe une méthode alternative pour réaliser des cutscenes in-game directement dans l'éditeur de **scène** à l'aide des **timelines**, qui ne sera pas traitée dans ce chapitre. Veuillez-vous référer au chapitre consacré à l'éditeur de **scène**.



Vidéo, Audio et Sous-titre

Une **cinématique** est constituée d'un fichier vidéo au format **MP4**, d'un fichier audio au format **OGG** et d'un fichier sous-titre au format **SRT**.

English		
<input type="checkbox"/> Video	FRENCH.agv	...
<input type="checkbox"/> Audio	FRENCH.ogg	...
<input type="checkbox"/> Sub-Title	ENGLISH.srt	...
French		
<input type="checkbox"/> Video	FRENCH.agv	...
<input type="checkbox"/> Audio	FRENCH.ogg	...
<input type="checkbox"/> Sub-Title	FRENCH.srt	...

Après avoir créé un nouvel asset **Cinematic** depuis la barre d’outils principale ou le raccourci **ALT+C**, un dossier est ajouté au projet pour placer la vidéo, l’audio et les sous-titres de toutes les langues. Vous avez la possibilité d’associer un fichier à chaque élément d’une langue et ainsi réutiliser le même fichier pour différentes langues. Il se peut que la vidéo soit unique et que les fichiers sonores soient spécifiques à chaque langue.

Encodage d’un fichier vidéo MP4

Pour des raisons de compatibilité optimale, il est préférable d’encoder les vidéos au format H.264 Baseline 3.1 en 1280x720 pixels et **24** images par seconde.

Intégration par le code

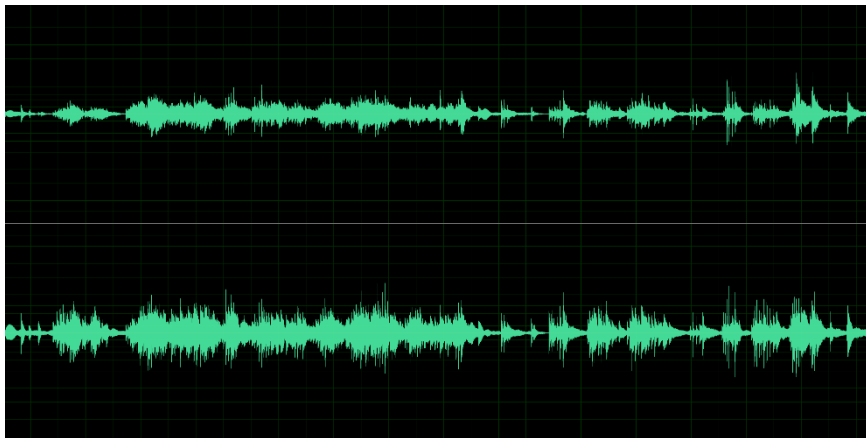
Afin de lancer une cinématique dans votre jeu, vous devez modifier un **rôle** en ajoutant une nouvelle boîte **Cinematic**. Modifiez le paramètre **Cinematic** pour spécifier l’asset de votre vidéo. Reliez-la aux autres boîtes selon la composition de votre narration. Le programme quittera la boîte à l’arrêt de la vidéo.



Pour forcer le joueur à visionner la vidéo en intégralité, ouvrez l’asset et changez l’option **Skip** en **None**.

AUDIO

L'audio a une place importante dans les jeux d'aventure : tout particulièrement la musique et les voix. C'est pourquoi **seccia.dev** propose des fonctionnalités simples mais essentielles à l'intégration des sons.



Musique et ambiance sonore

Les musiques et les ambiances sonores sont au format **OGG (Windows)** et **MP3 (iOS, Android, WebGL)** sont placées dans le dossier **SONG** à la racine du projet. Quatre pistes sont proposées, ce qui permet de jouer jusqu'à quatre fichiers simultanément afin de dissocier l'ambiance sonore de la musique. Vous pouvez enchaîner deux musiques ou ambiances sonores sur la même piste en utilisant la boîte **song**.

Bruitage

Les bruitages sont des fichiers audios courts au format **WAV** et situés dans le dossier **SOUND** à la racine du projet. Ces sons sont joués sur des pistes indépendantes.

```
play_sound "BOOM"
```

Voix

Les voix au format **OGG** (**Windows**) et **MP3** (**iOS**, **Android**, **WebGL**) sont placées dans le dossier **VOICE** des assets **Dialog**. Ce dossier contient un fichier par réplique. Pour nommer correctement vos fichiers, vous devez récupérer l'identifiant de la boîte dans l'éditeur et spécifiez le nom de la langue comme ceci :

```
D_MYDIALOG\VOICE\EN_16.ogg  
D_MYDIALOG\VOICE\FR_16.ogg
```

Si plusieurs paragraphes séparés par un trait sont présents dans la même boîte, vous devez ajouter son index après l'identifiant à l'exception du premier paragraphe :

```
D_MYDIALOG\VOICE\EN_16.ogg  
D_MYDIALOG\VOICE\EN_16_1.ogg  
D_MYDIALOG\VOICE\EN_16_2.ogg
```

Les identifiants sont accessibles depuis les propriétés et inscrits dans les fichiers **HTML** générés via le menu **TEXT/Export to HTML**.

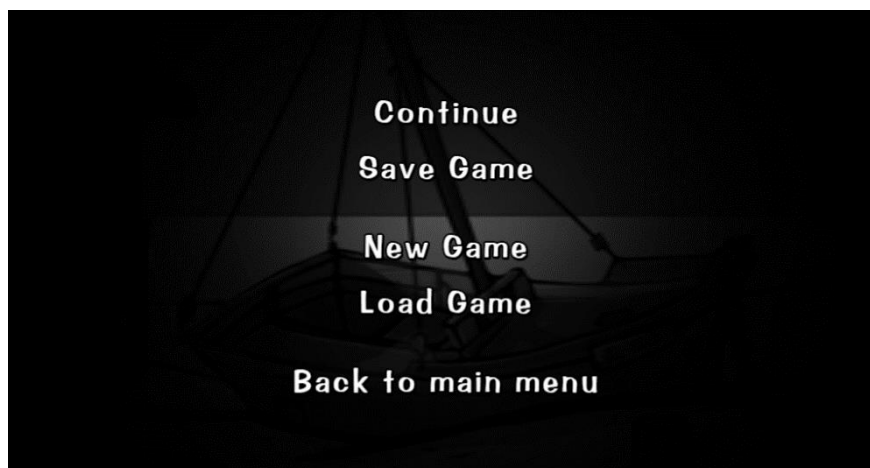
SAUVEGARDE

Les jeux créés avec **seccia.dev** peuvent contenir jusqu'à six emplacements de sauvegarde pour permettre aux joueurs de reprendre leur partie quand bon leur semble. Il faut différencier les sauvegardes locales et les sauvegardes en ligne plus complexes à mettre en place.

Pour faire apparaître le menu de sauvegarde dans le jeu, vous devez activer la propriété **Menu** dans les paramètres du projet à la section **Savegame**.

Savegame		
<input type="checkbox"/> Menu	True	...
<input type="checkbox"/> Server URL		...
<input type="checkbox"/> Server game (letter/digit)		...
<input type="checkbox"/> Version (0=discard)	1	...
<input type="checkbox"/> Autosave for all puzzles	False	...

Sinon seules les fonctions du langage permettront de gérer les sauvegardes.



Sauvegarde locale

L'emplacement **0** est réservé à la sauvegarde automatique enclenchée par la fonction **save**. C'est à vous d'appeler cette fonction lorsque vous souhaitez sauvegarder la progression du joueur. Il ne peut y avoir qu'une seule sauvegarde

automatique par jeu. En cas de nouvelles parties, l'ancienne sauvegarde sera écrasée par la nouvelle progression. Si vous voulez charger cette partie à la suite d'une action sans passer par le menu, vous pouvez le faire en appelant la fonction **load**.

Les emplacements 1 à 4 sont réservés aux sauvegardes manuelles. Si le menu est désactivé ou si vous utilisez un menu personnalisé, vous pouvez quand même accéder à ces emplacements en appelant les fonctions **load_game** et **save_game**.

Le dernier emplacement portant le numéro 5, est réservé aux sauvegardes en ligne.

Pour afficher une icône à l'écran pendant la sauvegarde pour informer le joueur, placez un fichier **LAYOUT_AUTOSAVE** dans le dossier **IMAGE** de votre projet.

Sauvegarde en ligne

La sauvegarde en ligne consiste à conserver la progression du jeu sur un serveur web pour poursuivre la partie sur une autre plateforme ou pour ne pas perdre les données en cas de réinstallation de l'application ou perte de l'appareil. Une seule sauvegarde par compte d'utilisateur est permise.

La mise en place du système de sauvegarde en ligne nécessite un site web avec au minimum la version 5.5 de **PHP** installée. Aucune base de données n'est nécessaire. **Android** requiert un certificat **SSL** pour établir une connexion sécurisée via le protocole **HTTPS**. Je vous conseille d'opter pour une solution sécurisée sur toutes les plateformes. Certains hébergeurs en fournissent une gratuitement.

Vous trouverez les explications relatives à l'installation des scripts sur un serveur web dans le dossier **Server** du logiciel.

Les scripts **PHP** vous permettront de stocker sur le même espace disque de votre site, les comptes utilisateurs de tous vos jeux créés avec **seccia.dev**.

Étant donné que les scripts collectent des informations personnelles de vos joueurs, vous devez respecter le règlement général sur la protection des données **RGPD** en renseignant, dans les paramètres du projet, le lien vers la page de votre politique de confidentialité. Veuillez-vous orienter vers les organismes officiels pour prendre connaissance du règlement afin de rédiger correctement votre document.

Chapitres

En ce qui concerne un jeu épisodique, vous pouvez être amené à débloquent des chapitres depuis un menu personnalisé. Afin de générer des fichiers de sauvegarde par chapitre, vous devez lancer une partie de votre jeu, sauvegarder la progression à l'endroit souhaité, récupérer le fichier SAV produit par l'application et le transférer dans le dossier **SCENARIO** en le nommant ainsi :

```
SCENARIO\CHAPTER1.sav
SCENARIO\CHAPTER2.sav
```

Ces fichiers en lecture seule seront intégrés à l'application lors du build et la fonction **load_chapter** vous permettra de les charger.

```
load_chapter 1
```

Faites attention à la compatibilité des sauvegardes entre deux versions de votre application. En revanche le format est universel et fonctionnera sur toutes les plateformes.

Données partagées

Il est tout à fait possible d'enregistrer des données supplémentaires qui seront partagées par toutes les parties de l'appareil. Ces données sont propres au jeu, comme le déblocage d'un chapitre ou d'un achievement. Elles ne peuvent pas être sauvegardées en ligne.

Voici quelques exemples de code des fonctions à utiliser :

```
write_game_value "CHAPTER2" 1
write_game_value "CHAPTER3" 0
write_game_value "SCORE" 10
read_game_value variable "SCORE"
alert $variable
```


LOCALISATION

Je vous ai expliqué au début de l'ouvrage qu'une des particularités du jeu d'aventure narratif est l'importance de l'histoire, des dialogues et de la mise en scène, qui jouent un rôle clé dans l'enrichissement du gameplay au même niveau que les énigmes. Par conséquent, si la narration est un des piliers de la structure de votre jeu et que les dialogues sont maladroits, l'expérience de jeu sera complètement gâchée et les joueurs passeront à côté de votre univers. Cela sous-entend que les textes doivent être bien écrits et bien traduits.

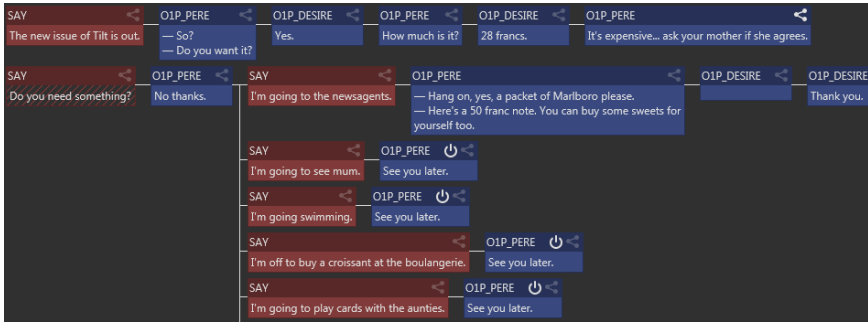
seccia.dev propose deux modes d'édition pour faciliter l'écriture des dialogues et la traduction en de nombreuses langues, sans taper une seule ligne de code.

Avant de commencer, vous devez d'abord activer les langues souhaitées dans les paramètres du projet. Pour plus d'informations à ce sujet je vous renvoie au chapitre **Projet**.

40	O1P_STANDARDISTE	0,0,0		Operator	Standardiste
41	O1P_SURVEILLANTE	0,0,0		Supervisor	Surveillante
42	O1P_VIEUX	0,0,0		The Old Man	Le Vieux
43	O1S_CARREFOUR_CHAT	0,0,0		Felix	Félix
44	O1S_CDI_PHOTOCOPIEUSE	0,0,0		Printer	Photocopieuse
45	O1S_CHAMBRES_PORTE	0,0,0		Door	Porte
46	O1S_CHAMBRES_TIROIR	0,0,0		Drawer	Tiroir
47	O1S_COMMERCE_CANETTE				
48		1,0,0	SUB	Can	Canette
49	O1S_COUR_BALLON	0,0,0		Ball	Ballon

Édition interne

La première étape consiste à rédiger le texte original en utilisant l'interface du logiciel. Les titres des éléments sont modifiables dans les éditeurs de **scène** et d'**objet**. L'éditeur de **dialogue** permet de construire une conversation complète entre un ou plusieurs personnages, tout en proposant des choix de réplique au joueur. Contrairement au scénario, les conversations ne sont pas des graphes mais des arborescences : une boîte ne peut avoir qu'un seul parent direct mais plusieurs enfants.



Quel que soit l'éditeur utilisé, vous pouvez éditer les textes de toutes les langues à tout moment sans sortir de l'environnement du logiciel. Cette méthode est pratique pour corriger rapidement des fautes ou effectuer des changements mineurs sur les traductions.

En revanche, je vous déconseille de procéder de la sorte pour traduire entièrement votre jeu. Optez plutôt pour l'édition externe.

Édition externe

L'édition externe a trois principaux avantages :

- garantir une efficacité de la traduction
- travailler avec des traducteurs externes
- bénéficier du format **UTF-8**

Cette méthode consiste à exporter l'intégralité du texte vers des fichiers **CSV** au format **UTF-8** éditables dans **Excel**, et à les réimporter après modifications. Il est important de conserver le format **UTF-8** à l'enregistrement et de conserver la virgule comme séparateur de colonnes.

Pourquoi est-ce plus efficace ? Tout simplement parce qu'il est plus pratique de traduire un unique fichier **CSV** sur deux colonnes que d'aller ouvrir tous les assets au sein du logiciel. Vous obtenez une meilleure visibilité du travail à fournir et vous pouvez vous servir de votre tableur habituel.

Le second avantage permet d'envoyer les textes aux traducteurs sans avoir besoin de transmettre les sources de votre jeu. De plus, **seccia.dev** génère un **CSV** par langue, ce qui facilite l'échange de fichiers lorsque plusieurs traducteurs travaillent sur le même projet.

Enfin le troisième avantage est de pouvoir traduire le texte en utilisant d'autres alphabets non reconnus par le logiciel grâce au format **UTF-8** des fichiers **CSV**.

Cette méthode peut également être adoptée pour la relecture du texte original avant la phase de localisation. L'export **HTML** est aussi une bonne alternative et le format est plus facilement convertissable en **PDF**.

Alphabet Latin, Cyrillique et autres

L'interface **seccia.dev** ne gère pas l'**Unicode**, seulement le format **ASCII**. C'est pourquoi il est nécessaire de passer par les fichiers **CSV** pour traduire votre texte en russe ou en chinois par exemple.

Les caractères **Unicode** sont codés en hexadécimal comme ceci : ~FFFF. Pour convertir un texte **Unicode** dans ce format, il suffit de copier le texte dans le presse-papier et de cliquer sur le menu **TEXT/Unicode to seccia.dev**. Le contenu du presse-papier sera alors modifié et vous pourrez coller le nouveau texte dans un champ du logiciel.

Pour les langues cyrilliques, n'oubliez pas d'inclure l'alphabet cyrillique dans la génération de votre police de caractères. Notez que les polices existantes ont été générées avec qu'un seul alphabet pour des raisons d'optimisation graphique.

En ce qui concerne les langues asiatiques, la manière de procéder est légèrement différente parce qu'il n'est pas possible de générer par avance tous les symboles de la langue. Au moment du build de votre jeu, **seccia.dev** ne récupère que les symboles présents dans vos textes afin de limiter le nombre de textures à générer. Des polices de caractères asiatiques sont utilisées par défaut mais vous pouvez en choisir d'autres depuis les propriétés du projet en écrivant leur nom exact.

Les lettres turques ne figurant pas dans le jeu de caractères **ASCII**, sont incluses dans l'alphabet **Latin** de **seccia.dev**.

!"#\$%&'()*+,-./012345	КЛМНОПРСТУФХЦЧШ
6789:;=?@ABCDEFGHI	ЩЪЫЬЭЮЯабвгдежзий
JKLMNOPQRSTUVWXYZ	клмнопрстуфхцшщъы
Z\[^_abcdefghijklmnopqrstuvwxyz	ьэюяё
qrsstuvwxyziz€¥©«»¿À	
ÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÑÒÓ	
ÔÕÖØÙÚÛÜßàáâãäåæçèé	
ëìíîïîñòóôõöøùúûüÿğİ	
ŒœŞş•□ËАБВГДЕЖЗИЙ	

SCRIPT

Nous voilà arrivés au sujet sensible. Même si vous n'êtes pas un développeur dans l'âme, l'apprentissage du code **seccia.dev** ne devrait pas vous freiner mais au contraire vous encourager grâce à son langage de programmation minimaliste.

Le langage est sensible à la casse uniquement pour les commandes et fonctions. Par exemple **if** ne peut pas s'écrire **If**, **iF** ou **IF**.

Les commentaires commencent par deux barres obliques (slash) et se terminent à la fin de la ligne d'instruction.

```
// if the current scene is S_PARK or S_SUBWAY
if scene S_PARK S_SUBWAY
    // change the current scene
    jump S_HOME
end

// test variable value
set pseudo "Sylvain"
if var pseudo "Sylvain" "Mike" "Bob"
    // here
end

// play animation
animate O_HERO FIGHT *
animate O_HERO FIGHT
```

L'astérisque permet de conserver la valeur par défaut du paramètre. Dans l'exemple, le paramètre de la direction est ignoré pour jouer une animation sans changer la direction de l'**objet**. Étant donné qu'il s'agit du dernier paramètre de la fonction, il est possible de l'exclure.

Syntaxe

Une seule instruction par ligne est autorisée. Une instruction est constituée de trois éléments : la commande, la fonction et les paramètres. La commande et les paramètres peuvent être optionnels. La commande **return** n'exige ni fonction ni paramètres. Les commandes **else**, **end**, **break** et **continue** n'acceptent ni fonction ni paramètres.

```
command function param1 param2 ...
```

Une commande permet d'appliquer des conditions à un bloc d'instructions, d'effectuer des boucles d'instructions et de retourner un entier. Pour définir un bloc, il suffit de placer les instructions en dessous de la commande. Les indentations sont insérées automatiquement par l'éditeur de code mais ne sont pas obligatoires. Voici quelques exemples :

```
if function param
    // instructions
else_if function param
    // instructions
else
    // instructions
end
```

Les commandes **if**, **if_not**, **while**, **while_not** sont utilisées pour ouvrir un nouveau bloc d'instructions. Pour fermer ce bloc, vous devez obligatoirement ajouter la commande **end**.

```
if function param
    // instructions
end
```

Les commandes **else**, **else_if**, **else_if_not** sont optionnelles et utilisées pour ajouter une condition à un bloc **if** ou **if_not** avant **end**.

```
if function param
    // instructions
else
    // instructions
end
```

Les commandes **while** et **while_not** doivent toujours être fermées avec **end**. Elles peuvent aussi contenir des commandes **break** et **continue**.

```
while function param
    // instructions
end
```

Quant à la commande **return**, elle n'a pas besoin d'un bloc d'instructions et s'utilise sur une seule ligne. L'interpréteur quittera alors immédiatement le script sans exécuter les instructions restantes. Dans le code ci-dessous, la première instruction empêchera d'exécuter les deux instructions suivantes.

```
return
return 1
return android
```

Voici un récapitulatif des commandes :

if	Permet d'exécuter le bloc d'instructions en dessous si l'expression est vraie.
if_not	Permet d'exécuter le bloc d'instructions en dessous si l'expression est fausse.
else	Permet d'exécuter le bloc d'instructions en dessous si les expressions précédentes sont fausses.
else_if	Permet d'exécuter le bloc d'instructions en dessous si l'expression est vraie et si les expressions précédentes sont fausses.
else_if_not	Permet d'exécuter le bloc d'instructions en dessous si l'expression est fausse et si les expressions précédentes sont fausses.
while	Permet d'exécuter le bloc d'instructions en dessous tant que l'expression est vraie.
while_not	Permet d'exécuter le bloc d'instructions en dessous tant que l'expression est fausse.
end	Permet de fermer un bloc if , if_not , while ou while_not .
break	Permet d'interrompre la boucle while ou while_not en cours et de continuer l'exécution du script après le end , c'est-à-dire après la fermeture du bloc.
continue	Permet de revenir immédiatement à l'instruction while ou while_not de la boucle en cours. Si la condition est fausse, la boucle se terminera normalement.
return	Permet de quitter le script immédiatement en retournant un entier. Par défaut les scripts retournent la valeur 0.

Variable

Le langage utilise qu'un seul type de variable : les chaînes de caractères, autrement dit les textes. Lorsque des fonctions attendent une valeur numérique, le texte est alors converti en nombre. Les guillemets deviennent optionnels pour les valeurs numériques et les textes sans espace.

Pour définir une nouvelle variable ou changer sa valeur :

```
set foo "monday"
set foo 10
```

Pour concaténer deux textes, deux possibilités :

```
// Method 1
set foo "the sky is " "blue"
// Method 2
set foo "the sky is "
cat foo "blue"
```

Pour manipuler des nombres :

```
set foo 10 // foo is 10
add foo 1 // foo is 11
sub foo 3 // foo is 8
mul foo 2 // foo is 16
div foo 4 // foo is 4
mod foo 2 // foo is 0
```

Pour comparer deux textes :

```
set foo "monday"
if var foo "monday"
    // if foo is monday
end
if var foo "monday" "tuesday"
    // if foo is monday OR tuesday
end
```

Pour comparer deux nombres :

```
set foo 10
if equal $foo 10
    // if foo is 10
end
if less $foo 10
    // if foo is less than 10
end
```

Si vous jetez un coup d'œil à la documentation de ces fonctions, vous remarquerez la différence entre le paramètre **name** et le paramètre **value**. Pour **name** il s'agit du nom de la variable sans guillemets et pour **value** il s'agit de la valeur de la variable. Mais est-il possible de spécifier la valeur d'une autre variable pour le paramètre **value** ? Oui bien sûr et de deux manières différentes.

La première méthode permet de récupérer la valeur d'une variable par son nom en utilisant le caractère **\$** comme en **PHP**.

```
set a "monday"
set b $a
```

Cette technique est évidemment valable pour toutes les fonctions et dans l'éditeur de **rôle**.

```
set foo "THEME"
play_sound $foo
```

La seconde méthode est légèrement plus complexe mais de toute façon elle vous sera moins utile. Lorsqu'une variable contient le nom d'une autre variable, il est possible de récupérer sa valeur avec le caractère **&**.

```
set a "monday"
set b "a"
set c &b
```

La première instruction change la valeur de la variable **a**. Il va de même pour la deuxième instruction qui change la valeur de la variable **b**. Ainsi nous avons *a="monday"* et *b="a"*. En utilisant le caractère **&**, la troisième instruction récupère en fait la valeur de la variable **a**. Le gros avantage de cette méthode est de pouvoir former des noms de variable à l'exécution du jeu pour modifier ou récupérer leur valeur. Voici un exemple qui affiche le texte **Italy** à l'écran.

```
set countries_0 "France"
set countries_1 "Italy"
set index 1
set variable "countries_" $index
alert &variable
```

Les fonctions préfixées par **list_** permettent de manipuler des textes dans une liste plus facilement. Une liste n'est rien d'autre qu'un texte contenant des valeurs séparées par une virgule. Voici un exemple qui affiche le texte **France** à l'écran.

```
set countries "Italy,France"
list_add countries "UK"
list_sort countries
```

```
list_get countries 0 first_country
alert $first_country
```

Le nom d'une énigme est préfixé par le caractère # et ne peut être utilisé que par les fonctions faisant références au scénario.

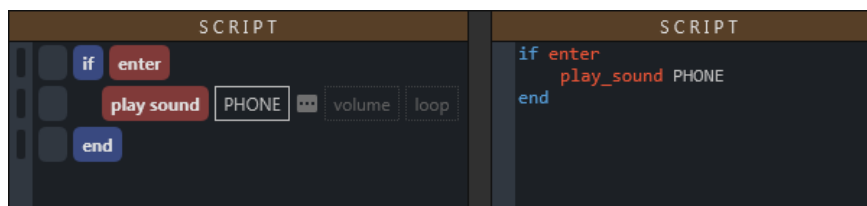
```
success #get_key
if resolved #open_door
end
```

Les tags doivent être préfixés par le caractère @.

```
if scene @prologue
end
```



Assisted Scripting

seccia.dev intègre un mode plus intuitif pour éditer les scripts : l'Assisted Scripting. Ce mode est activé par défaut à la première installation du logiciel et vise à faciliter l'édition des scripts sans passer par l'éditeur de code classique, c'est-à-dire en tapant les instructions au clavier. Il s'agit seulement d'une représentation graphique différente du script. Le code reste inchangé. Le passage du mode classique au mode graphique est possible à tout moment en cliquant sur l'icône **Assisted Scripting** de la barre d'outils principale. Vous pouvez comparer les deux modes sur l'image ci-dessous.



Même si vous êtes codeur, l'Assisted Scripting peut vous aider à développer plus efficacement votre jeu grâce à l'assistance des boîtes de dialogue.

ASSISTED SCRIPTING • INSTRUCTION



if

else

else if

while

end

break

continue

return

not


set default walk

SCENE.OBJECT

WALK

This function sets the default WALK animation. If the parameter is not specified, the function uses the default game animations.

SECCIA. D

	show	{SCENE}OBJECT
<input type="checkbox"/> All functions	hide	{SCENE}OBJECT
<input type="checkbox"/> Variable	visible	{SCENE}OBJECT
<input type="checkbox"/> Box	set parent	{SCENE}OBJECT {OBJECT/LABEL}
<input type="checkbox"/> Role	set default anims	{SCENE}OBJECT {STOP} {TALK} {WALK}
<input type="checkbox"/> Play	set default stop	{SCENE}OBJECT {STOP}
<input type="checkbox"/> Role	set default walk	{SCENE}OBJECT {WALK}
<input type="checkbox"/> Camera	set default talk	{SCENE}OBJECT {TALK}
<input type="checkbox"/> Object	set tag	{SCENE}OBJECT {"tag"}
<input type="checkbox"/> Position	get tag	{SCENE}OBJECT variable
<input type="checkbox"/> Animation	tag	{SCENE}OBJECT @tag
<input type="checkbox"/> Direction	set sticker	{SCENE}OBJECT {"value"}
<input type="checkbox"/> Path	get sticker	{SCENE}OBJECT variable
<input type="checkbox"/> Scene	sticker	{SCENE}OBJECT "value"
<input type="checkbox"/> Object	set elevator	{SCENE}OBJECT {height}
<input checked="" type="checkbox"/> Scene Object	set grid	{SCENE}OBJECT {index}
<input type="checkbox"/> Dialog	set path	{SCENE}OBJECT {index}
<input type="checkbox"/> Player	set opacity	{SCENE}OBJECT {opacity}
<input type="checkbox"/> Effect	set angle	{SCENE}OBJECT {degree}

This function changes the opacity of the specified object.
Opacity is a value between 0 and 100.
The default opacity value is 100.

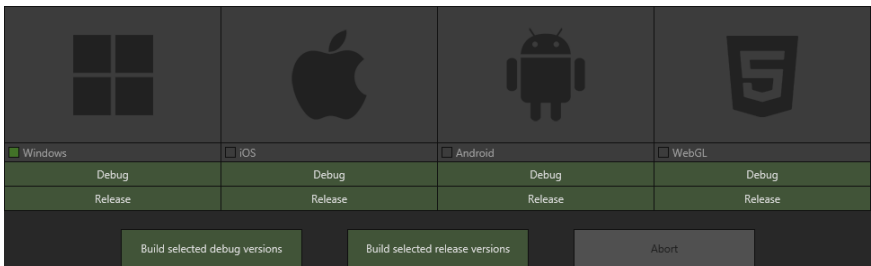
La boîte de dialogue principale vous permet de composer votre instruction en allant chercher les éléments de façon totalement intuitive. Contrairement à l’affichage classique du code, tous les paramètres disponibles d’une fonction sont visibles en un clin d’œil et sont évidemment interactifs. Les mots clés du langage sont définis en haut de la fenêtre et toutes les fonctions à gauche sont classées par catégorie. Après avoir validé une fonction, il vous suffit de remplir les paramètres en cliquant dessus et en suivant les indications fournies à l’écran. Les instructions sont directement éditables depuis les scripts sans rouvrir la boîte de dialogue principale.

— 155 —

BUILD

Si vous avez correctement suivi les étapes du premier chapitre, vous pourrez exporter votre jeu vers ces plateformes, sinon vous devrez vous contenter de la version **Windows**.

À part les assets graphiques situés dans le dossier **PLATFORM**, un jeu **seccia.dev** n'a besoin d'aucun portage pour tourner sur toutes les plateformes proposées.

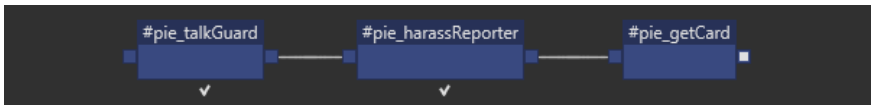


Live, Debug et Release

Pour chaque plateforme, vous avez la possibilité d'exporter votre jeu selon deux formats : **Debug** ou **Release**.

Release est réservé aux versions distribuables. Ces versions contiennent toutes les fonctionnalités et optimisations de votre jeu qui peuvent prendre beaucoup de temps au moment de la génération.

Debug est réservé exclusivement au développement. Ces builds vous permettent de tester le jeu en simulant l'état d'une partie. Si vous sélectionnez une boîte du scénario et que vous appuyez sur la barre d'espace, vous verrez apparaître un symbole en dessous de la boîte. Ce repère vous indique que l'énigme est supposée être résolue en mode **Debug**.



Live fonctionne comme **Debug** mais permet de lancer le jeu plus rapidement en évitant la compression des textures. Les images **PNG** sont directement lues par la carte graphique sans optimisation, ce qui peut engendrer une différence de

rendu et de performance. Cette configuration est très utile pour améliorer votre efficacité sur le projet en limitant la durée de génération des builds **Windows** et **WebGL** lors du développement. Ces builds sont enregistrés dans les dossiers respectifs **play_windows** et **play_web**.

Android

Avant de pouvoir générer le build **Android**, vous devez d'abord créer un fichier **Keystore** avec l'outil **keytool** du **JDK**. Ce fichier vous servira pour tous vos projets. Vous devez ensuite renseigner les champs du groupe **Signature** dans les propriétés du projet : le chemin du fichier, le nom et les mots de passe que vous avez utilisés avec l'outil **keytool**. Selon la configuration, vous obtiendrez soit un fichier **APK** soit un fichier **AAB**.

iOS

seccia.dev génère un dossier qu'il faudra ouvrir et compiler sur **macOS** avec l'application **Xcode** distribuée gratuitement par **Apple**. En revanche un compte développeur payant est nécessaire à la fois pour publier votre jeu sur l'**AppStore** et pour le tester sur tablettes et téléphones.

Depuis l'apparition des puces M, vous pouvez faire tourner une application **iOS** sur **macOS**. Je vous invite à vous renseigner auprès d'**Apple** pour suivre les démarches de publication d'un jeu **iOS** sur **macOS**.

En tant qu'indépendant, je vous déconseille de publier deux versions : une pour **iOS** et une autre pour **macOS**. Les mises en production sont chronophages.

WebGL

Deux distributions sont possibles :

public

La version par défaut à uploader sur votre site web.

local

La version locale fonctionne avec un serveur local accessible à l'adresse suivante : **http://localhost/**

Concernant la version locale, désormais par mesure de sécurité les navigateurs bloquent systématiquement la connexion à l'application **WebGL** lorsque vous utilisez le protocole **file** pour spécifier le chemin du dossier. Il est donc nécessaire

de passer par un serveur local sur le port **80.seccia.dev** s'en charge si vous testez votre jeu en cliquant sur **Play release game**. Le serveur sera automatiquement déconnecté à la fermeture du projet ou du logiciel.

Post-build

Il est possible d'exécuter un programme en ligne de commande pour chaque plateforme à la fin de la génération. Cette fonctionnalité ne s'applique qu'au build **Release**.

Pour changer l'icône de l'exécutable **Windows**, téléchargez l'application **Resource Hacker** à l'adresse suivante :

<http://www.angusj.com/resourcehacker/>

Puis ajoutez la ligne de commande suivante dans les propriétés de votre projet en remplaçant **"exe"** par le chemin complet de l'exécutable généré par **seccia.dev** et **"ico"** par le chemin complet du fichier icône au format **.ico** :

```
ResourceHacker.exe -open "exe" -save "exe" -action  
modify -res "ico" -mask ICONGROUP,103,
```

Ligne de commande

Vous pouvez générer vos builds **Release** en ligne de commande, en rajoutant simplement l'argument **-build** après le nom du fichier.

```
seccia.exe "d:\game\game.seccia" -build
```

Tous les plateformes sélectionnées et enregistrées dans le projet seront prises en compte.

L'exécutable **seccia.exe** vous permet d'éditer des fichiers **JSON** en modifiant des valeurs de type **string** en ligne de commande. Vous pouvez ainsi modifier les paramètres du projet pour générer des builds alternatifs via un fichier **.BAT**. Par exemple si votre projet inclut seulement la plateforme **Windows** et que vous souhaitez générer un build **Android** en ligne de commande, vous pouvez effectuer une copie temporaire comme ceci :

```
[BUILD.bat]
```

```
copy d:\game.seccia d:\temp.seccia
```

```
seccia.exe -json d:\temp.seccia "project build windows" "0"  
"project build android" "0"
```

BUILD

```
seccia.exe d:\temp.seccia -build
```

```
del d:\temp.seccia
```

L'outil ne pourra pas ajouter de nouvelles valeurs mais seulement les modifier.

PROJET UNITY

Le runtime des jeux conçus avec **seccia.dev** est entièrement développé avec **Unity**. Le projet **Unity** est donc fourni avec le logiciel pour que vous puissiez le compiler avec votre licence **Unity** et éventuellement rajouter de nouvelles fonctionnalités. Le seul inconvénient est de devoir reporter les modifications à chaque nouvelle mise à jour du logiciel. C'est pourquoi il vaut mieux rajouter des fichiers et ne modifier que **AgePlugin.cs** pour faciliter la maintenance des mises à jour.

```
18 references
public Scene FindScene(string uid)
{
    if ( uid.Length==0 )
        return null;

    Scene scene;
    if ( m_scenesByUID.TryGetValue(uid, out scene)==false )
        return null;

    return scene;
}
```

Fonctions exposées

Pour éviter de modifier le code source du projet et de casser quelque chose, j'ai mis en place un fichier source **AgePlugin.cs** en exposant une liste de fonctions et d'événements. Il vous suffit de changer le code de ces fonctions.

```
public static bool OnAppInit() {
    return true;
}

public static void OnAppQuit() {
    Application.Quit();
}

public static void OnAppUpdate() {
}

public static bool IsLoadEnabled() {
    return true;
}
```

```

public static bool IsSaveEnabled() {
    return true;
}

public static bool OnUserButton(int index, string url) {
    return true;
}

```

Callback

La fonction **Callback** du code **C#** est appelée lorsque vous utilisez la fonction **callback** dans votre jeu. Les deux paramètres sont optionnels et n'ont pas de signification particulière. Vous pouvez vous servir du premier paramètre pour désigner un nom d'événement et le second pour sa valeur. La valeur booléenne de retour est envoyée à votre jeu de manière synchronisée. Voici un exemple d'utilisation de la fonction **Callback**.

```

public static bool Callback(string param1, string param2)
{
    switch ( param1 )
    {
        case "unlock":
            if ( param2=="1" )
                // code here
            break;
    }
    return true;
}

```

Interlude

Vous pouvez lancer une action depuis le projet **Unity** dès que le joueur récupère la main sur le jeu. Contrairement à la fonction **Callback** qui est appelée immédiatement, la fonction **Interlude** est asynchronisée.

```

public static void Interlude(string param) {
    return true;
}

```

Si l'application est en attente pour appeler la fonction **Interlude**, il est possible d'annuler la requête en cours à l'aide de la fonction **cancel_interlude**.

Si vous souhaitez bloquer l'application pendant l'interlude, vous devez appeler les fonctions statiques **G.InterludeLock** et **G.InterludeUnlock** du projet **Unity**. Si vous spécifiez une durée à la première fonction, vous n'aurez pas besoin d'appeler la seconde fonction, à moins d'écourter le délai initial.

Lorsque l'application est bloquée, vous pouvez personnaliser l'écran du jeu grâce à la fonction **OnInterludeDraw**. Cette fonction doit retourner la valeur **true** pour pouvoir l'utiliser correctement. Le code suivant affichera un écran blanc.

```
public static bool OnInterludeDraw()
{
    G.FillScreen(new Color(1.0f, 1.0f, 1.0f, 1.0f))
    return true;
}
```

Enfin, si vous avez besoin de valider une énigme depuis le projet **Unity**, vous pouvez vous servir de la fonction **G.Success** en spécifiant l'identifiant de la boîte. Cet identifiant est à récupérer dans l'éditeur de **scénario**.