

Beste de savoir

Réalisez un jeu de plates-formes avec GDevelop

12 août 2019

Table des matières

1. Installation et préparation de GDevelop	3
1.1. Téléchargement de GDevelop	3
1.2. Création d'un nouveau jeu	3
1.3. Les images de notre jeu	4
2. Les objets et les comportements	6
2.1. Création du personnage	6
2.2. Création des plates-formes	7
2.3. Ajout des comportements aux objets	8
2.3.1. Ajout du comportement au « Personnage »	8
2.3.2. Ajout du comportement à l'objet « Plateforme »	9
2.4. Changement de la couleur de fond de la scène	10
2.5. Premier test du jeu	10
3. Les événements	11
3.1. Comment fonctionnent les événements ?	11
3.1.1. Conditions et actions, quésaco ?	11
3.1.2. Les sous-événements	12
3.2. Gestion des animations du personnage	13
3.2.1. Saut et chute du personnage	13
3.2.2. Animation de déplacement et arrêt	15
3.3. Changement de direction du personnage	16
3.4. Déplacement de la caméra	17
Contenu masqué	18
4. Les variables et les expressions	20
4.1. Ajout des pièces	20
4.1.1. Ajout de l'objet	20
4.1.2. Suppression des pièces quand le joueur les collecte	20
4.1.3. Un peu de son	21
4.2. Comment fonctionnent les variables ?	21
4.2.1. Une variable ?	22
4.2.2. Types de variables	22
4.2.3. Portées de variable	22
4.2.4. Comment utiliser les variables dans GDevelop ?	23
4.3. Comptage de score avec une variable	23
4.3.1. Définition de la variable	23
4.3.2. Augmentation du score lors de la collecte de pièces	24
4.4. Les expressions	25
4.4.1. Qu'est-ce qu'une expression ?	25
4.4.2. Quelle est la syntaxe des expressions ?	27

Table des matières

4.5. Affichage du score	28
4.5.1. Création du compteur	28
4.5.2. Création d'un calque	29
4.5.3. Mise à jour du texte de « Compteur »	30
4.6. Amélioration du compteur	31
Contenu masqué	33
4.7. Correction	33
5. Annexes	34
5.1. Exporter le jeu	34
5.1.1. La plateforme	34
5.1.2. Changer la plateforme du jeu	34
5.1.3. Génération du jeu final	35

GDevelop est un logiciel permettant de créer des jeux vidéo 2D sans coder, en utilisant un système de conditions et d'actions. Contrairement à la majeure partie des logiciels concurrents, il a la particularité d'être libre et multiplate-forme. Vous pouvez donc réaliser vos jeux vidéo sans contraintes. Le logiciel permet la création de jeux natifs et de jeux HTML5.

Dans ce tutoriel, nous allons réaliser pas à pas un jeu de plates-formes en 2D.

Ce tutoriel s'adresse à tous, pas besoin de connaissances en programmation pour utiliser le logiciel, car tout se fait de manière logique avec un système d'événements.

1. Installation et préparation de GDevelop

GDevelop est un logiciel libre et multiplate-forme permettant de réaliser des jeux vidéo en 2D sans utiliser de langage de programmation. Dans cette partie, nous allons installer GDevelop et débiter la création d'un jeu de plates-formes en configurant le projet et en y ajoutant les images.

1.1. Téléchargement de GDevelop

Si vous n'avez pas GDevelop, je vous invite à le télécharger sur [le site officiel](#) .

i

Le logiciel est disponible sur Windows, sur Ubuntu (via un ppa), sur Fedora/OpenSUSE/Arch Linux (via un dépôt sur OpenSUSE Build Service) et une version bêta est disponible sur OS X.

1.2. Création d'un nouveau jeu

Après avoir lancé GDevelop, la page d'accueil s'affiche.



FIGURE 1.1. – Page d'accueil de GDevelop.

Afin de débiter la création d'un nouveau jeu, cliquez sur « Nouveau » dans le ruban « Projets ». Une fenêtre s'ouvre et vous permet alors de choisir un modèle de jeu, le type du jeu (natif ou HTML5) et le dossier dans lequel créer le jeu.



FIGURE 1.2. – La fenêtre de création de projet.

1. Installation et préparation de GDevelop

Vous pouvez choisir de créer un jeu HTML5 ou natif (peu importe, car sa réalisation sera identique). Sélectionnez ensuite « Projet vierge » afin de créer un projet vide. Enfin, n'oubliez pas de préciser où vous souhaitez créer le projet (fichier d'extension `.gdg`).



Dans le chemin du projet, il faut bien saisir le chemin du fichier `.gdg` et non seulement le dossier où le fichier de projet doit être créé.



Un modèle de projet « *Platformer* » est disponible et fournit déjà un jeu de plates-formes complet. Évidemment, cela n'a aucun intérêt dans le cadre de ce tutoriel.

Après avoir validé votre choix, le projet est créé et sa première scène (appelée « Nouvelle scène ») est automatiquement ouverte pour être éditée. Une scène est une partie du jeu qui possède ses propres objets et ses événements (qui vont animer le jeu, nous y reviendrons plus tard). Par exemple, le jeu pourra contenir une scène représentant le menu, une représentant le 1^{er} niveau, etc.

Lorsqu'une scène est ouverte, l'interface de GDevelop ressemble à ceci.



FIGURE 1.3. – GDevelop avec l'éditeur de la scène ouvert.

- À gauche, le gestionnaire de projets affiche le(s) projet(s) ouvert(s) et permet de voir leur contenu : les images du jeu (« Images ») et les différentes scènes du jeu.
- À droite, l'éditeur d'objets permet de gérer les objets de la scène actuellement ouverte.
- En bas, un onglet permet de passer de l'édition de la scène (« Scène ») à l'éditeur d'événements (« Événements ») qui permettra de définir la logique du jeu.

1.3. Les images de notre jeu

Avant de débiter la création du jeu à proprement parler, je vous invite à télécharger [des images](#) qui seront utiles lors de la création du jeu.

Dès que le téléchargement est fini, il suffit d'extraire les images (ainsi que les sons et polices de caractères) dans le dossier du projet.

Maintenant, pour ajouter les images au jeu, il faut ouvrir la « banque d'images » en double-cliquant sur « Images » dans le gestionnaire de projets. Désormais, en cliquant sur « Ajouter une image » dans le ruban supérieur, ajoutez les différentes images fournies dans l'archive. Les images sont maintenant utilisables par le projet.

1. Installation et préparation de GDevelop



FIGURE 1.4. – Les images ajoutées dans la banque d’image.

Nous pouvons maintenant retourner dans la scène « Nouvelle scène » en utilisant les onglets juste en dessous du ruban supérieur.

i

GDevelop supporte les images aux formats BMP, JPEG (non progressifs), GIF (non animés) et PNG.

Des images trop grandes (au-delà de 8192 pixels de largeur et/ou hauteur) peuvent poser problème à cause de la limitation de taille des textures de certaines cartes graphiques.

!

Attention, les images au format **PNG enregistrées au format 8 bits (couleurs indexées) ne sont pas supportées** (c’est parfois le format de PNG utilisé par certains *sprites* en *pixel art*). Il faut alors les réenregistrer en PNG 32 bits (sous GIMP, cliquer sur « Image > Mode > RVB » à la place de « Couleurs indexées »).

Dans la prochaine partie, nous allons pouvoir réellement débiter la création du jeu en créant des objets qui vont s’animer à l’écran lors du déroulement du jeu.

2. Les objets et les comportements

Maintenant que le projet du jeu est configuré et les images ajoutées, nous allons ajouter les objets et leurs comportements afin de faire les bases de notre jeu.

2.1. Création du personnage

Nous allons maintenant créer le premier objet de votre jeu. Ce sera le personnage principal, qui sera en mesure de courir, sauter sur les plates-formes et collecter des pièces.

Pour créer un nouvel objet, faites un clic droit sur l'élément « Objets » dans l'éditeur d'objets et cliquez sur « Ajouter un objet ». À la place de cette manipulation, vous pouvez également cliquer sur l'éditeur d'objets pour faire apparaître le ruban supérieur qui lui est dédié et cliquer sur « Ajouter un objet ».

Une nouvelle fenêtre apparaît et vous permet de choisir le type d'objets à créer.



FIGURE 2.1. – La liste des types d'objets.

Choisissez « Sprite (image animée) ». L'objet se crée et vous pouvez saisir le nom de l'objet. Nous l'appellerons « Personnage ». Maintenant que l'objet est créé, nous allons l'éditer afin de définir ses différentes animations. Pour cela, double-cliquez dessus dans l'éditeur d'objets.



FIGURE 2.2. – L'éditeur de *sprites*.

La fenêtre se décompose en différents volets :

- à gauche, la liste des animations du *sprite* ;
- à droite, toutes les images disponibles dans la banque d'image du projet, ainsi qu'une petite zone d'aperçu pour voir l'image sélectionnée ;

2. Les objets et les comportements

- en bas, la liste des images / *frames* de l'animation sélectionnée ;
- au centre, un aperçu de la *frame* sélectionnée, avec la possibilité de changer son masque de collision et d'ajouter des points de repère.

Pour notre objet « Personnage », nous allons créer trois animations : une pour le personnage immobile, une autre pour le personnage en train de marcher et une dernière pour le personnage en train de sauter ou de chuter. Utilisez le bouton « + » du volet « Animations » pour créer deux animations supplémentaires.

Dans la première animation (« Animation 0 »), ajoutez l'image `p1_stand.png` en la glissant-déposant de la banque d'images dans le volet « Images ». Dans la seconde animation (« Animation 1 »), ajoutez `p1_walk3.png`, `p1_walk4.png`, `p1_walk5.png`, `p1_walk6.png` et `p1_walk7.png` (dans l'ordre, sinon l'animation sera un peu bizarre). Enfin, dans la dernière animation, ajoutez l'image `p1_jump.png`.

Pour les animations 0 et 2, qui ne sont composées que d'une seule image chacune et qui seront donc statiques, il n'y a pas besoin de configuration supplémentaire. Par contre, pour l'animation 1, il faut préciser l'intervalle de temps entre chaque *frame*, et qu'elle doit se répéter. Pour cela, faites un clic droit sur « Animation 1 » et cliquez sur « Temps entre chaque image » afin de définir l'intervalle de temps à 0.2 secondes (il faut bien utiliser un point et non une virgule). Enfin, répétez la même opération, mais cette fois-ci en cliquant sur « Boucler l'animation ».

Voilà, vous pouvez fermer l'éditeur de *sprites* car votre objet « Personnage » est prêt. Maintenant, il est possible de placer l'objet sur la scène en glissant-déposant l'objet depuis l'éditeur d'objets jusqu'à l'éditeur de scènes (au centre). On parle alors de **placer une instance de l'objet** « Personnage » sur la scène, car il est tout à fait possible d'en placer plusieurs sur la même scène (ce sera le cas pour les pièces, par exemple).



Sur certains environnements de bureau (en particulier Gnome 3 et tous ceux basés sur ce dernier), le bouton pour fermer l'éditeur de *sprites* peut être masqué. Dans ce cas, pour fermer la fenêtre, il faut faire un clic droit sur sa barre de titre et cliquer sur « Fermer ».



FIGURE 2.3. – Une instance de « Personnage » placée sur la scène.

2.2. Création des plates-formes

Nous allons maintenant créer un objet représentant une plate-forme. Cette fois-ci, ce ne sera pas un objet de type « Sprite (image animée) » mais un objet « Mosaïque ». Cela permet d'utiliser une texture répétée (et non étirée) lorsque l'objet est redimensionné, permettant de créer facilement de grandes plates-formes (sans placer des milliers d'instances de la plate-forme). Nous l'appellerons « Plateforme ».

2. Les objets et les comportements

Dans l'éditeur de l'objet « Mosaïque », sélectionnez l'image `grassHalfMid.png` et cliquez sur



le bouton afin de définir cette image comme image de l'objet « Mosaïque ». Vous pouvez également définir la taille par défaut de l'objet quand il sera placé sur la scène. Mettez par exemple « 48 » en hauteur.

Validez votre modification en cliquant sur « Ok ».

Votre objet « Plateforme » est désormais prêt et peut être instancié (placé) sur la scène. Vous pouvez également redimensionner chaque instance (dans notre cas, ce sera plus intéressant en largeur) afin de faire des plates-formes plus grandes.



FIGURE 2.4. – Exemple de plates-formes placées sur la scène.

2.3. Ajout des comportements aux objets

Il faut maintenant animer notre jeu pour en faire un jeu de plates-formes. Vu que ce genre de jeu est très commun, GDevelop fournit ce que l'on appelle des **comportements** pour définir très rapidement ce que vont faire les objets.

Dans notre cas, nous utiliserons deux comportements : « Objet se déplaçant sur une plateforme » pour l'objet « Personnage » et « Plateforme » pour l'objet « Plateforme ». Comme vous l'aurez compris, le premier comportement permet de gérer le déplacement d'un objet sur d'autres objets qui possèdent le deuxième comportement.

2.3.1. Ajout du comportement au « Personnage »

Pour ajouter le premier comportement à l'objet « Personnage », faites un clic droit sur ce dernier dans l'éditeur d'objets et cliquez sur « Autres propriétés ». Un panneau s'ouvre. Il contient les autres propriétés de l'objet (en plus de ce que l'on a défini dans l'éditeur de *sprites* précédemment).



FIGURE 2.5. – Les autres propriétés de « Personnage ».

2. Les objets et les comportements

Cliquez ensuite sur « Ajouter... » à côté de « Ajouter un comportement ». De la même manière que pour les types d'objet, une fenêtre s'ouvre pour pouvoir sélectionner le comportement à ajouter à l'objet.



FIGURE 2.6. – La liste des comportements.

Sélectionnez « Objet se déplaçant sur une plateforme » et validez.

i

Tous les comportements sont grisés car ils ne sont pas utilisés par le projet et sont donc désactivés par défaut (pour réduire le poids du jeu en ne prenant que ce qui est nécessaire). Ajouter pour la première fois au projet un comportement va automatiquement activer le module (appelé « extension ») associé.

De nouvelles propriétés apparaissent en-dessous de la catégorie « *PlatformerObject* » dans la liste des propriétés. Elles permettent de personnaliser le comportement de l'objet.



FIGURE 2.7. – De nouvelles propriétés apparaissent.

Nous allons les laisser comme elles sont actuellement.

2.3.2. Ajout du comportement à l'objet « Plateforme »

Maintenant, ajoutons le comportement « Plateforme » à l'objet « Plateforme ». Pour cela, allez dans les « Autres propriétés » de l'objet « Plateforme ».

i

Il n'est pas nécessaire de fermer et rouvrir le panneau « Propriétés ». En effet, le contenu du panneau change dynamiquement en fonction de l'objet sélectionné dans l'éditeur d'objets. Il suffit alors de juste sélectionner « Plateforme » pour faire apparaître ses propriétés.

L'ajout se déroule de la même manière que pour « Personnage » mais il faut utiliser le comportement « Plateforme ». Ce comportement comporte bien moins de propriétés que l'autre.

2. Les objets et les comportements

La propriété « Type » permet de spécifier le type de plates-formes : une plate-forme standard (par défaut, le joueur peut marcher dessus mais ne peut pas passer à travers du tout), un *jumpthru* « Accessible depuis le bas » (le joueur peut sauter à travers de bas en haut) ou une échelle (sur laquelle le joueur peut monter ou descendre). Dans notre cas, l'option par défaut est la bonne.

Vous pouvez alors fermer le panneau de « Propriétés ».

2.4. Changement de la couleur de fond de la scène

Ajoutons maintenant un beau ciel bleu à notre scène principale. Pour cela, faites un clic droit sur la scène dans le gestionnaire de projets et cliquez sur « Modifier les propriétés ». Une nouvelle fenêtre apparaît et un bouton permet de changer la couleur de l'arrière-plan. Vous pouvez mettre la couleur que vous souhaitez.

2.5. Premier test du jeu

Nous allons maintenant tester pour la première fois le jeu. Pour cela, cliquez sur « Aperçu » dans le ruban supérieur. Si vous avez choisi de créer un jeu natif, l'aperçu se déroule dans la fenêtre de GDevelop, sinon, l'aperçu se lance dans votre navigateur par défaut.

Utilisez les touches , , ,  pour déplacer le personnage, et  pour faire sauter le personnage.



FIGURE 2.8. – Premier test du jeu.

Nous avons déjà un jeu de base fonctionnel. Il nous reste à faire fonctionner les différentes animations du personnage et à ajouter les pièces, un score et des ennemis.

Les objets combinés aux comportements permettent déjà de donner vie au jeu de plates-formes. Néanmoins, se limiter aux comportements pour rendre un jeu interactif va considérablement limiter les possibilités de création. C'est pour cela que nous allons utiliser le système d'événements dans la partie suivante.

3. Les événements

Les comportements ne permettent pas à eux seuls de réaliser un jeu. En effet, ils agissent sur des aspects bien précis du jeu mais ne permettent pas une liberté totale sur le fonctionnement de ce dernier. C'est pourquoi nous allons aussi utiliser le système d'événements pour animer notre jeu. Pour cela, il faut aller dans l'onglet « Événements » (situé juste en dessous des onglets permettant de naviguer entre les scènes ouvertes).

3.1. Comment fonctionnent les événements ?

Pour comprendre le fonctionnement des événements, laissons de côté le jeu un instant.

3.1.1. Conditions et actions, quésaco ?

Une scène peut contenir plusieurs événements. Ces derniers sont composés de **conditions** et d'**actions**. Les conditions vérifient différents éléments du jeu (par exemple, le joueur est-il en train de sauter ?) et les actions agissent sur le jeu (par exemple, créer un ennemi à la position...). Au sein d'un événement, les actions sont exécutées si et seulement si toutes les conditions de l'événement sont vraies (les conditions d'un événement sont liées par un « et » binaire). Voici comment un événement est représenté dans GDevelop.



FIGURE 3.1. – Un exemple d'événement dans GDevelop.



Que se passe-t-il ici ?

D'après ce qui a été dit précédemment, si une instance de l'objet de nom « MonObjet » a sa position X (sur la scène) supérieure à 200 pixels, alors les actions de l'événement sont exécutées. Ici, il y a trois actions. Une première qui va supprimer une/des instance(s) de l'objet « MonObjet », et une seconde qui va créer/instancier une instance de l'objet « UnAutreObjet » à une position précise sur la scène. La troisième va afficher les objets « EncoreUnAutreObjet » (qui auraient été cachés au préalable).



Mais quelles instances de « MonObjet » vont être supprimées ? Toutes ou certaines en particulier ?

En fait, dans GDevelop, les conditions qui testent des objets *sélectionnent* des instances de l'objet. Ici, la condition « La position X de MonObjet est > 200 » est *vraie* et va *sélectionner* les instances de « MonObjet » qui respectent bien la condition. S'il y a plusieurs conditions dans un événement, les instances sélectionnées seront celles qui respectent toutes les conditions (qui parlent de l'objet spécifié).

Donc, l'action « Supprimer [l'instance de] l'objet MonObjet » ne va agir que sur les objets sélectionnés, donc les instances de « MonObjet » dont la position X est bien supérieure à 200 pixels.

Cette règle ne s'applique pas à la 2^e action de l'exemple, car elle est un peu spéciale : elle n'agit pas sur une instance d'un objet mais en crée une.

Enfin, la 3^e action agit sur des instances de l'objet « EncoreUnAutreObjet », or cet objet n'est jamais évoqué dans les conditions. Les actions qui agissent sur des objets qui ne sont pas traités par les conditions *vont s'exécuter par défaut sur toutes les instances de l'objet spécifié*. C'est donc ce cas de figure qui s'applique à l'exécution de la 3^e action.

Évidemment, il faut rappeler que les actions *ne s'exécuteront que si la condition est vraie*. Donc, toutes les instances de « EncoreUnAutreObjet » seront affichées si au moins un objet « MonObjet » a sa position X supérieure à 200 pixels. Il en va de même pour la 2^e action.

C'est cette subtilité, somme toute assez logique, qui va conférer une très grande puissance au système d'événements.

3.1.2. Les sous-événements

Dans GDevelop, il est également possible de créer des sous-événements qui sont en fait des événements à l'intérieur d'autres événements. Il sont placés en dessous de leur événement parent à la façon d'un arbre.



<http://zestedesavoir.com/media/galleries/2614/>

FIGURE 3.2. – Un événement contenant deux sous-événements.

Le concept est assez simple : les sous-événements ne sont pris en compte (conditions analysées et actions exécutées si elles sont vraies) que si les conditions de l'événement parent sont vraies. De plus, les instances d'objets sélectionnées par les conditions de l'événement parent sont propagées aux sous-événements.

Dans cet exemple, les instances de « MonObjet » qui ont une position X supérieure à 200 pixels seront :

3. Les événements

- coloriées en rouge (255;0;0) si elles ont une position Y inférieure à 400 pixels ;
- coloriées en vert (0;255;0) si elles ont une position Y supérieure ou égale à 400 pixels.

Les instances qui ont une position X inférieure ou égale à 200 pixels ne sont donc même pas traitées par les sous-événements à cause de cette propagation des instances sélectionnées. En fait, ces événements peuvent être réécrits sous forme de deux événements standards.



FIGURE 3.3. – Une autre façon de faire la même chose.

Évidemment, même si cela revient au même, cette manière de faire est moins facile à maintenir car une condition a dû être dupliquée.

i

Des sous-événements peuvent eux-même avoir des sous-événements et cela à l'infini.

Voilà ce qui boucle la théorie du fonctionnement des événements.

3.2. Gestion des animations du personnage


Maintenant que nous avons les bases du fonctionnement des événements, nous allons les utiliser afin de gérer l'animation du personnage. Nous voulons que son animation 0 s'affiche quand il est immobile et sur une plate-forme, son animation 1 quand il marche sur une plate-forme et son animation 2 quand il est en l'air (s'il saute ou tombe).

3.2.1. Saut et chute du personnage

Nous allons affecter l'animation 2 au personnage s'il tombe ou s'il saute. Pour cela, créez un nouvel événement en utilisant le bouton « Ajouter un événement » du ruban supérieur. Un événement vide est créé. Passez ensuite votre souris sur la partie gauche de l'événement pour faire apparaître le bouton « Ajouter une condition » et cliquez dessus. Une fenêtre apparaît, vous permettant de sélectionner la condition à ajouter dans sa partie gauche. Sélectionnez la condition « Est en train de tomber » qui se trouve dans la catégorie « Comportement plateforme ». Dès qu'elle est sélectionnée, des paramètres à remplir apparaissent sur la partie droite de la fenêtre.




http://zestedesavov

- « Objet » : c'est l'objet qui va être testé par la condition. Cliquez sur  pour ouvrir une fenêtre et sélectionner « Personnage ».

3. Les événements

- « Comportement » : le comportement qui sera utilisé. Ce n'est pas le nom français du comportement mais le nom donné lors de sa création. Dans quasiment tous les cas, il suffit



`http://zestedesavoir.com/media/galleries/2614/89eba2`

de cliquer sur le bouton *après avoir saisi l'objet* pour que le paramètre soit rempli automatiquement.

Voilà ce que devrait donner la 1^{re} condition.



`http://zestedesavoir.com/media/galleries/2614/`

FIGURE 3.4. – Les paramètres à saisir pour la première condition.

i

Vous n'êtes pas obligé d'utiliser toutes les fenêtres d'assistance pour remplir les paramètres. Vous pouvez les saisir directement.

Validez, et la condition a été ajoutée à l'événement. Vous pouvez à tout moment la changer en double-cliquant dessus.

Ensuite, en utilisant le bouton « Ajouter une action » qui apparaît au survol de la partie droite de l'événement, ajoutez l'action « Changer l'animation » qui se trouve dans la catégorie « Sprite (image animée) » > « Animations et images » en mettant comme paramètres ce qui suit.

- « Objet » : « Personnage ».
- « Signe de modification » : permet de définir l'opération mathématique utilisée avec le numéro d'animation actuel de l'objet « Personnage ». Ici, on mettra « = » car on veut affecter une nouvelle valeur (qui sera 2). D'autres opérations sont disponibles : on pourra par exemple utiliser « + » pour ajouter la valeur au numéro d'animation actuel de l'objet. Évidemment, cela n'a pas une grande utilité dans le cadre de cette action particulière, mais de nombreuses autres actions utilisent aussi ce paramètre « Signe de modification ».
- « Valeur » : c'est la valeur que nous souhaitons affecter, ici, en numéro d'animation (ou additionner si l'on avait choisi « + » en signe de modification). Mettez « 2 ».

Voici à quoi devrait ressembler ce 1^{er} événement.



`http://zestedesavoir.com/media/galleries/2614/`

3. Les événements

FIGURE 3.5. – Le premier événement.

Nous voulons également affecter l'animation 2 à « Personnage » s'il saute. Nous ne pouvons pas ajouter cela en 2^e condition de l'événement, car cela équivaldrait à faire un « et » entre les deux conditions. Nous pouvons alors créer un autre événement avec cette condition et y copier l'action du 1^{er} événement, ou utiliser une condition spéciale « ou » pour effectuer un « ou » binaire entre les deux conditions. Cette deuxième possibilité est la meilleure car elle évite de dupliquer une action : si jamais on change l'animation de la chute/saut, il n'y aura qu'une seule action à modifier.

Nous allons donc ajouter la condition « ou » qui est une condition un peu spéciale car elle peut contenir d'autres conditions (celles sur lesquelles faire le « ou »). De la même manière que pour la 1^{re} condition, ouvrez la fenêtre d'ajout de condition et ajoutez la condition « ou » qui se trouve dans la catégorie « Avancé ». Elle n'a pas de paramètre donc vous pouvez valider directement. Ensuite, il suffit de glisser-déposer la condition « Personnage est en train de tomber » à l'intérieur la zone où il est inscrit « Pas de condition » de la condition « ou ».

En survolant la condition déjà présente dans le « ou », cliquez sur « Ajouter une condition » afin d'ajouter la condition « Est en train de sauter » (catégorie « Comportement plateforme »). Remplissez les paramètres de la même façon que dans la condition « Est en train de tomber » et validez. Le résultat doit être celui-ci.



FIGURE 3.6. – Les deux conditions dans la condition « ou ».

3.2.2. Animation de déplacement et arrêt

Si nous regardons les conditions disponibles pour tester les mouvements du personnage, nous voyons que nous avons une condition pour tester si un personnage est sur une plate-forme et une autre pour tester s'il est mobile ou non.



FIGURE 3.7. – Les conditions pour tester notre personnage se déplaçant sur des plates-formes.

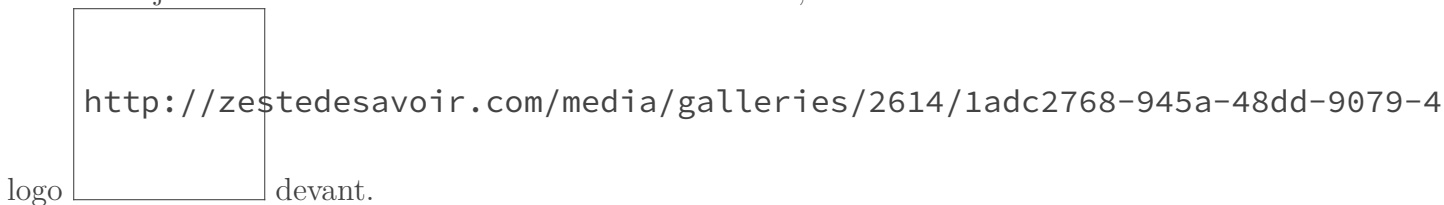
Nous allons utiliser ces deux conditions pour mettre les animations 0 et 1 du personnage. Ainsi :

- si le personnage est sur le sol **et** bouge : mettre l'animation de « Personnage » à 1 ;

3. Les événements

- si le personnage est sur le sol **et ne bouge pas** : mettre l'animation de « Personnage » à 0.

Avant de vous laisser réaliser les événements appropriés, sachez qu'il est possible d'inverser une condition pour réaliser le « ne... pas ». Une condition inversée renvoie « vrai » s'il est *fausse* et inversement. De plus, si elle traite des objets, elle sélectionne les instances d'objet qui ne valident pas la condition. C'est ce dont nous avons besoin pour faire « Personnage *ne bouge pas* ». Cela se met en place et utilisant la case à cocher « Inverser la condition » en bas de la fenêtre d'ajout de condition. Dans l'éditeur d'événements, les conditions inversées affichent le



Voilà, je vous laisse maintenant réaliser les événements. Au passage, pour ajouter un événement, vous pouvez continuer à utiliser le bouton du ruban ou vous pouvez utiliser le panneau mobile qui apparaît sous l'événement que vous survolez avec la souris.



FIGURE 3.8. – Panneau mobile/contextuel pour ajouter des événements.

Dès que vous pensez avoir réussi, affichez le cadre masqué juste en-dessous.

👁 Contenu masqué n°1

3.3. Changement de direction du personnage

En testant le jeu à cet instant, vous constaterez que les animations se déclenchent correctement mais que le personnage reste orienté à droite quelle que soit la direction vers laquelle il se dirige. Pour cela, nous allons utiliser l'action « Inverser l'affichage de l'objet horizontalement » qui permet de retourner un objet de type « Sprite » en réalisant une symétrie. Cette action se trouve dans la catégorie « Sprite (image animée) » > « Effet » et prend en paramètre le *sprite* à modifier et un « *yes/no* » pour savoir si l'on doit retourner le *sprite* ou non.

Pour savoir quand le retourner le *sprite* ou non, il faut évidemment être au courant de la direction vers laquelle se dirige le personnage. Il n'y a pas de conditions du comportement « Plateforme » pour savoir cela. À la place, nous allons tester la touche du clavier appuyée par l'utilisateur pour savoir si l'objet « Personnage » doit être tourné ou non. La condition se trouve dans la catégorie « Clavier » et s'appelle « Touche pressée ». Son seul paramètre est la

3. Les événements

touche du clavier à tester (un assistant permet en appuyant sur la touche souhaitée d'en donner le nom).



Ne pas confondre cette condition avec « Touche pressée (expression texte) », qui permet d'utiliser des expressions texte (variables...) pour désigner la touche à tester.

Nous devons donc réaliser deux événements :

- si la flèche droite est appuyée (« Right »), on doit désactiver l'inversion de « Personnage » ;
- si la flèche gauche est appuyée (« Left »), on doit activer l'inversion de « Personnage ».

À vous de jouer !

Dès que vous avez trouvé la solution, je vous invite à tester le jeu et à regarder la correction ci-dessous.

© Contenu masqué n°2

3.4. Déplacement de la caméra

Si vous désirez réaliser un niveau plutôt long, vous allez vite remarquer qu'il serait bien que la vue reste centrée sur le personnage. Encore une fois, les événements sont la solution à nos problèmes : une action permet de centrer la caméra sur un objet.

Au sein d'un nouvel événement, sans condition (ce qui signifie « exécuter tout le temps »), ajoutons l'action « Centrer la caméra sur un objet » en mettant « Personnage » en tant qu'objet, et en laissant les autres paramètres grisés vides.



FIGURE 3.9. – L'événement permettant de centrer la caméra sur le joueur.



Une autre action permet de centrer la caméra sur un objet *dans des limites*. Cela permet de centrer la caméra sur l'objet en limitant sa position dans un rectangle dont les coordonnées sont précisées dans l'action.

3. Les événements

Grâce aux événements, nous avons ajouté de nouvelles actions à notre jeu. C'est un système très puissant, qui permet de réaliser tout ce que l'on souhaite. Pour être sûr d'avoir la même chose, voici les événements que vous devriez avoir dans la scène « Nouvelle scène ».



FIGURE 3.10. – Les événements de « Nouvelle scène ».

Vous pouvez télécharger une archive contenant le projet dans son état actuel (ainsi que les ressources) [ici](#) ↗ .

Contenu masqué

Contenu masqué n°1

Solution

Vous avez trouvé ?

Je vais vous présenter deux solutions.

1^{re} solution

En utilisant deux événements, nous pouvons obtenir ceci.



FIGURE 3.11. – Première solution.

Remarquez bien que la condition « Personnage est en train de bouger » est inversée dans le 1^{er} événement.

Note : l'ordre des deux événements n'est pas important ici.

Néanmoins, vous constaterez que cette solution n'est pas parfaite et (si vous l'avez faite) vous avez dû copier deux fois la même condition : ce n'est pas idéal.

2^e solution

C'est pour cela que la 2^e solution est la meilleure des deux. Dans cette solution, nous utilisons des sous-événements. Pour ajouter un sous-événement à un événement, il suffit d'utiliser le bouton « Un sous-événement » à côté de « Ajouter un événement » dans la barre mobile qui apparaît au survol d'un événement.

3. Les événements



FIGURE 3.12. – Deuxième solution, avec les sous-événements.

C'est cette solution qui faut privilégier, car elle évite la duplication de conditions (même si celles-ci ne sont pas très coûteuses, d'autres le sont, comme les tests de collision). [Retourner au texte.](#)

Contenu masqué n°2

Rien de bien compliqué à expliquer. Dans chaque événement, nous testons une des deux touches et activons ou non l'inversion *horizontale* de « Personnage ».



FIGURE 3.13. – Les événements pour inverser l'affichage de « Personnage ».

Si vous testez votre jeu maintenant, le personnage devrait bien tourner en fonction de son déplacement. [Retourner au texte.](#)

4. Les variables et les expressions

Le jeu est tout à fait jouable. Mais vous constaterez qu'il manque un peu d'intérêt. C'est pour cela que, dans cette partie, nous allons ajouter des pièces que le joueur pourra collecter. Par ailleurs, un texte permettra d'afficher le score du joueur en fonction du nombre de pièces collectées par ce dernier.

4.1. Ajout des pièces

4.1.1. Ajout de l'objet

Dans un premier temps, créez un objet « Piece » qui sera un *sprite* avec pour unique image `coinGold.png`. Vous pouvez disposer plusieurs instances de l'objet sur la scène.



FIGURE 4.1. – Les pièces disposées dans le niveau.

4.1.2. Suppression des pièces quand le joueur les collecte

Maintenant que les pièces sont placées sur la scène, nous allons utiliser les événements afin de supprimer les pièces quand le joueur entre en collision avec ces dernières.

Pour cela, nous allons ajouter un nouvel événement. Cet événement permettra de supprimer les instances de « Piece » quand le joueur entre en collision avec. Je vous laisse chercher un peu les conditions et les actions à utiliser pour compléter l'événement.

Dès que vous pensez avoir trouvé, ou si vous éprouvez des difficultés à remplir l'événement, vous pouvez déplier le cadre ci-dessous.

© Contenu masqué n°3

4.1.3. Un peu de son

Nous allons maintenant ajouter un peu de son à notre jeu. En particulier, un « son de pièce » sera joué quand le joueur collectera une pièce. Pour cela, ajoutez l'action « Jouer un son » (catégorie « Audio ») à l'événement qui gère la suppression des pièces (car nous voulons jouer le son lors de la collecte d'une/de pièce(s)). L'action possède un paramètre obligatoire, qui est le chemin du son à jouer, et trois autres paramètres facultatifs. Dans notre cas, il s'agit juste de sélectionner le son à jouer, qui est le fichier « coin.wav » (fourni dans l'archive des ressources)

avec le bouton 



FIGURE 4.2. – L'action pour jouer le son.

i

GDevelop supporte les sons/musiques aux formats wav et ogg (le mp3 n'est pas supporté, car ce n'est pas un format libre).

D'ailleurs, dans GDevelop, vous constaterez qu'il existe des actions pour jouer un son, mais aussi des actions pour jouer une musique. La différence vient de la façon dont la piste audio est chargée et lue : en effet, les sons sont chargés entièrement avant d'être lus, alors que les musiques sont chargées par morceaux et pendant la lecture (juste les morceaux dont le logiciel aura besoin pour produire le son). Ainsi, les actions « son » sont plus adaptées aux bruitages et aux sons courts, alors que les actions « musiques » sont plus pratiques pour les pistes assez longues, car les charger en mémoire pourrait consommer trop de **RAM**.

La système de pièces est fonctionnel, il ne reste plus qu'à y ajouter le système de score.

4.2. Comment fonctionnent les variables ?

Pour pouvoir compter le score, nous allons devoir utiliser une **variable**. Mais ne vous inquiétez pas, les variables sont assez simples à utiliser dans GDevelop.

4.2.1. Une variable ?

i

Une variable peut être vue comme une boîte qui porte une étiquette (son nom) et qui contient une valeur. L'intérêt des variables, c'est que nous pouvons accéder facilement à leur valeur en utilisant leur nom (et aussi en modifier la valeur à partir du nom).

Cette valeur est uniquement conservée en mémoire durant l'exécution du jeu. Ainsi, en relançant le jeu, le contenu des variables est perdu, ce n'est donc pas un moyen de stockage définitif (pour faire un système de sauvegarde de partie, par exemple). Néanmoins, les variables sont très utiles pendant l'exécution du jeu pour permettre de stocker des informations comme le nom du joueur, son score dans le niveau, la vie d'un ennemi, etc.

4.2.2. Types de variables

Dans GDevelop, il existe deux types de variables : les variables ayant pour valeur **des nombres** et les variables ayant pour valeur **du texte**. Il faut savoir qu'il est possible, au cours de l'exécution du jeu, d'affecter une valeur en texte à une variable « nombre », auquel cas, elle deviendra une variable « texte » (l'inverse marche aussi).

i

Il existe aussi un autre type de variables : les variables structures, qui peuvent contenir d'autres variables à l'intérieur. Néanmoins, leur utilisation est plus complexe et nous n'en parlerons pas dans ce tutoriel.

4.2.3. Portées de variable

Les variables sont également définies par leur portée. La portée est simplement l'endroit où la variable existe (et sa valeur est conservée). Ainsi, il y a trois portées :

- **Les variables globales** : ces variables existent et leur valeur est conservée dans tout le jeu, donc n'importe quelle scène.
Exemple d'utilisation : pseudonyme du joueur ;
- **Les variables de scène** : ces variables existent et leur valeur est conservée dans la scène actuelle. Ainsi, un changement de scène implique la perte du contenu de ces variables (même si on relance la même scène).
Exemple d'utilisation : score du joueur dans le niveau en cours ;
- **Les variables d'objet** : ces variables existent pour chaque instance d'un objet. Les valeurs sont indépendantes pour chaque instance d'objet.
Exemple d'utilisation : la vie d'un ennemi (chaque instance de l'ennemi aura sa propre valeur de vie).

4.2.4. Comment utiliser les variables dans GDevelop ?

Les conditions pour tester la valeur (nombre) ou le texte d'une variable de scène ou globale se trouvent dans la catégorie « Variables » (sous-catégorie « Variables Globales » pour les variables globales). Les actions pour modifier leur valeur/texte se situent dans la même catégorie.

Quant aux conditions et actions pour tester/modifier les variables d'objet, elles se trouvent dans la catégorie « Tous les objets > Variables ».

4.3. Comptage de score avec une variable

Nous allons maintenant mettre en place une variable de scène afin de stocker le score du joueur pendant qu'il joue le niveau. Nous allons, dans un premier temps, définir la variable, puis ajouter une action pour incrémenter la variable à l'événement qui gère la collecte de pièces.

4.3.1. Définition de la variable

Nous allons dans un premier temps **définir** la variable, c'est-à-dire la créer et lui donner une valeur initiale. Suivant la portée souhaitée, la liste permettant de définir les variables se trouve à différents endroits :

- **pour accéder à la liste des variables de la scène**, il suffit de faire un clic droit sur la scène dans le gestionnaire de projets et de cliquer sur « Modifier les variables initiales » ;
- **pour accéder à la liste des variables globales**, il faut faire un clic droit sur le nom du projet dans le gestionnaire de projets, puis cliquer sur « Modifier les variables globales » ;
- **pour accéder à la liste des variables d'objet d'un objet précis**, il faut sélectionner un objet dans l'éditeur d'objets, puis aller dans ses « autres propriétés » (clic droit) et cliquer sur « Cliquez pour éditer... », à côté de « Variables » dans la catégorie « Variables d'objet ». Ces variables seront appliquées à toutes les instances de cet objet lors de leur création sur la scène.




Il est également possible de **définir des variables d'objet pour une seule instance précise d'un objet**, auquel cas il faut sélectionner une **instance** de l'objet sur la scène, aller dans ses propriétés et cliquer sur « Cliquez pour éditer... », à côté de « Variables » dans la catégorie « Propriétés de l'instance ». Ces variables d'objet spécifiques à l'instance viennent s'ajouter à celles définies pour l'objet (voire les remplacer si elles portent le même nom).

Dans notre cas, nous allons définir une variable de scène pour stocker le score, elle s'appellera « score ». Ouvrez donc la liste des variables de la scène « Nouvelle scène ». La fenêtre suivante apparaît.



FIGURE 4.3. – La liste des variables de la scène.



Le bouton  permet de définir une nouvelle variable en saisissant son nom. Dès que la variable est ajoutée à la liste, vous pouvez double-cliquer dessus pour définir sa valeur initiale : saisir un nombre permet de définir une variable de type numérique et saisir un texte (sans les guillemets) permet de définir une variable de type texte.

Je vous invite donc à créer la variable « score » et à lui affecter la valeur initiale 0.



FIGURE 4.4. – La variable « score » définie.

i

En réalité, toute cette étape de définition de variable n'est pas obligatoire. En effet, si la variable utilisée dans les événements n'est pas définie, GDevelop l'initialise à 0 ou avec un texte vide (suivant si la première action/condition qui l'utilise concerne les variables de type numérique ou textuel). Néanmoins, cette pratique est recommandée, car elle permet de saisir la vraie valeur initiale de la variable.

Vous pouvez fermer la fenêtre.

4.3.2. Augmentation du score lors de la collecte de pièces

Grâce aux événements, nous allons maintenant faire en sorte que le jeu ajoute 100 points au score quand une pièce est collectée. Cette incrémentation doit avoir lieu lors de la collecte d'une pièce, c'est donc toujours dans l'événement de gestion de la collecte de pièces que nous allons travailler.

Ajoutons donc à cet événement l'action qui permet de modifier la valeur (nombre) d'une variable de scène (« Variables > Valeur d'une variable »). Il y a trois paramètres à remplir pour cette action.

4. Les variables et les expressions

- « Variable » : c'est la variable dont nous souhaitons changer la valeur. Le bouton



permet de sélectionner la variable à modifier parmi celles qui ont été définies dans la scène (c'est pourquoi il peut être utile de déclarer ses variables) ;

- « Signe de modification » : c'est l'opération que nous voulons effectuer sur la variable. Ainsi, = permet d'affecter une valeur, + d'ajouter à la valeur actuelle de la variable, et ainsi de suite pour les trois autres opérations ;
- « Valeur » : c'est la valeur que nous voulons affecter, ajouter, soustraire... à la valeur de



la variable (suivant le choix fait dans « Signe de modification »). Le bouton permet de saisir des valeurs plus complexes (permettant par exemple de récupérer sous forme de nombre la valeur d'une variable), ce que l'on appelle des **expressions**.

Vu que nous voulons **ajouter** 100 à la valeur de la variable « score », l'action doit être remplie comme ci-dessous.



FIGURE 4.5. – Les paramètres de l'action de modification de variable.

Vous pouvez valider et ajouter l'action à l'événement qui devrait maintenant ressembler à cela.



FIGURE 4.6. – L'événement avec l'incrémement du score.

4.4. Les expressions

4.4.1. Qu'est-ce qu'une expression ?


Avant d'afficher le score sur la scène, nous avons besoin de parler des **expressions**. Les expressions sont en fait des constructions syntaxiques permettant de récupérer différentes valeurs liées au

4. Les variables et les expressions

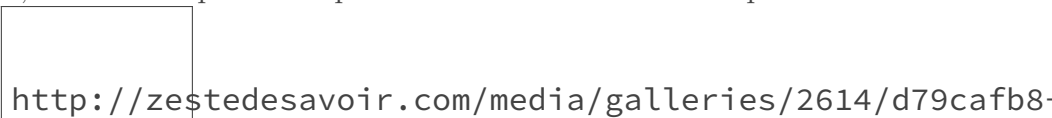
jeu, d'effectuer des opérations mathématiques ou des manipulations de texte, et de les passer aux paramètres des actions, conditions ou autres expressions elles-mêmes. Par exemple, il existe une expression pour récupérer le contenu d'une variable, une autre pour récupérer la taille de l'écran de l'utilisateur, etc. Tout comme les variables, il existe deux types d'expressions :


- **les expressions numériques** : elles manipulent des nombres et toutes les valeurs numériques. Il existe toutes sortes d'expressions numériques pour récupérer le contenu des variables de type nombre, la vitesse d'un objet... Ces expressions peuvent être saisies



dans les paramètres suivis par le bouton  (c'est le cas pour la « Valeur » dans l'action de modification de variable, par exemple) ;

- **les expressions textuelles** : elles manipulent du texte. Là aussi, il existe de nombreuses expressions textuelles qui permettent de récupérer le contenu sous forme de texte d'une variable de type texte, etc. Ces expressions peuvent être saisies dans les paramètres



suivis par le bouton  (c'est le cas pour le « Texte » dans les actions de modification de **texte** de variables).





Les boutons  et  permettent respectivement d'ouvrir l'éditeur d'expressions et l'éditeur de texte (expression texte).



FIGURE 4.7. – L'éditeur d'expressions numériques avec la liste des expressions disponibles.



FIGURE 4.8. – L'éditeur de texte avec une liste des expressions textuelles disponibles.

4.4.2. Quelle est la syntaxe des expressions ?

4.4.2.1. Syntaxe de base

La syntaxe dépend du type d'expressions.

Ainsi, les opérateurs `+`, `-`, `*` et `/` permettent d'effectuer respectivement une addition, une soustraction, une multiplication et une division dans les expressions numériques.

Exemple : `2+6` donne bien 8.

Le `+` permet de concaténer deux textes dans les expressions textuelles (mettre les deux textes bout à bout). Toujours dans les expressions textuelles, les textes statiques, c'est-à-dire les textes qui doivent apparaître tels quels dans le résultat de l'expression, doivent être entourés de `"`.

Exemple : `"Ceci est " + "un test !"` donne comme résultat le texte « Ceci est un test! ».



Notez que le nom d'une variable n'est pas une expression, c'est pourquoi son nom n'est pas entouré de guillemets quand on l'écrit dans une action.

4.4.2.2. Manipuler les données du jeu

Les objets et les variables fournissent des expressions pour récupérer leurs différentes valeurs. Ainsi, `Variable(score)` permet de récupérer le contenu de la variable de scène « score » sous forme de **nombre** (expression numérique) et `VariableString(score)` permet de récupérer son contenu sous forme de **texte** (la conversion est faite automatiquement si la variable contient un nombre). Pour les variables globales, il existe les expressions `GlobalVariable(maVariable)` et `GlobalVariableString(maVariable)`.

Pour toutes les expressions qui ont besoin d'un objet pour avoir du sens (par exemple, récupérer le contenu d'une variable d'objet, récupérer la largeur d'un objet...), l'expression est sous la forme `monObjet.expression(paramètres éventuels)`. Ainsi, `Ennemi.Variable(vie)` et `Ennemi.VariableString(vie)` permettent respectivement de récupérer le contenu de la variable « vie » d'une instance de l'objet « Ennemi » sous forme de nombre et de texte.

Ne vous inquiétez pas, les deux éditeurs d'expressions fournissent la liste des expressions disponibles et aident à leur saisie.

4.4.2.3. Utilisation des expressions

A la place d'écrire la valeur `100` en dur dans l'événement de comptage du score, nous allons définir une **variable de scène** « valeurPiece » contenant la valeur de score à ajouter à « score » en cas de collecte de pièce. Ainsi, nous allons mettre sa valeur à `100` et modifier l'action de modification de variable « score ».



FIGURE 4.9. – Action d’incrémentation améliorée.

Maintenant que vous avez acquis les bases du fonctionnement des expressions, nous pouvons continuer la création du compteur.

4.5. Affichage du score

4.5.1. Création du compteur

Nous allons utiliser un objet texte pour y afficher le score grâce à un événement. Créez donc un objet de type « Texte », nous l’appellerons « Compteur ». Faites un clic droit sur l’objet dans l’éditeur d’objets et cliquez sur « Editer » pour ouvrir son éditeur.



FIGURE 4.10. – L’éditeur d’objets texte.

La barre d’outils de l’éditeur contient un premier champ pour indiquer le chemin vers la police à utiliser (laisser vide pour utiliser la police intégrée à GDevelop), suivi d’un bouton pour aller choisir la police dans les fichiers. Les autres boutons permettent respectivement de changer la taille, la couleur, le style (gras, italique et souligné) du texte.

Changez donc la couleur du texte pour le mettre en une couleur lisible sur la scène, puis validez vos modifications.

i

Ce n’est pas la peine de saisir un texte dans l’objet, car nous modifierons son contenu via les événements pour afficher le score. En effet, le texte qui peut être mis dans l’éditeur est un texte statique, car les expressions ne peuvent être utilisées que dans les événements (elles sont interprétées à l’exécution du jeu).

4.5.2. Création d'un calque

Si nous disposons directement une instance de « Compteur » sur la scène, il bougera avec le reste des décors quand la caméra se centrera sur le joueur. C'est pour cela que nous allons créer un nouveau calque pour y placer l'objet.

Un **calque** peut être vu comme une couche de la scène ayant sa propre caméra. En créant un nouveau calque pour l'objet, il restera bien sur l'écran pendant que la caméra du calque par défaut (créé automatiquement avec la scène, celui où l'on a placé tous les autres objets) se centrera sur le joueur.

i

Par défaut, les actions de déplacement de caméra agissent sur la caméra du « calque de base » (calque par défaut). C'est donc le cas pour l'action de centrage de la caméra sur « Personnage ». Il est possible de les faire agir sur un autre calque en cochant leur paramètre « Calque » et en y sélectionnant un autre calque.

Pour gérer les calques, il faut ouvrir l'éditeur de calques accessible grâce au bouton « Éditeur de calques » dans le ruban « Scène ».




FIGURE 4.11. – L'éditeur de calques.

Vous pouvez y voir le « calque de base ». La flèche bleue à sa gauche indique que c'est le calque sur lequel les instances glissées-déposées depuis l'éditeur d'objets seront placées. Il suffit d'ailleurs de cliquer sur un autre calque dans la liste pour changer le calque sur lequel elles seront ajoutées.

Créez un nouveau calque (le nom n'a pas tant d'importance) avec le bouton



faites en sorte que ce nouveau calque soit en tête de la liste avec les flèches

. En effet, l'ordre a une importance, car les objets des calques s'affichent dans

4. Les variables et les expressions

l'ordre de leur calque. Enfin, sélectionnez le calque en cliquant dessus dans la liste afin que la flèche se trouve sur sa ligne. Vous pouvez désormais glisser-déposer « Compteur » sur la scène.

i

Je vous conseille de sélectionner de nouveau le « calque de base » après la manipulation pour ne pas placer par mégarde d'autres instances d'objets sur ce calque.

La scène devrait ressembler à l'image ci-dessous.



FIGURE 4.12. – La scène avec le compteur sur un deuxième calque.

En essayant le jeu, vous constaterez que le texte reste bien sur le coin de l'écran même quand le joueur se déplace.

4.5.3. Mise à jour du texte de « Compteur »

Nous allons maintenant créer un événement pour mettre à jour le contenu de l'objet. Cet événement n'aura pas de condition, car il faudra mettre à jour le texte continuellement, et aura pour action une action permettant de changer le texte de l'objet « Compteur ». L'action pour modifier le texte est « Objet Texte > Modifier le texte ». Elle prend en paramètre l'objet dont le texte doit être changé, un « Signe de modification » et une expression textuelle.

Mettez donc l'objet « Compteur » en paramètre et = en « Signe de modification ». Ensuite,



`http://zestedesavoir.com/`

ouvrez l'éditeur d'expressions textuelles en cliquant sur le bouton à côté du paramètre « Texte ».

Vous pouvez utiliser les panneaux en bas de l'éditeur pour ajouter des expressions. Pour ajouter l'expression qui récupère le contenu d'une variable de scène, il faut aller dans le panneau « Autres fonctions » et dans la catégorie « Variables ». Un double-clic sur « Variable de la scène » ouvre la liste des variables de la scène. Sélectionnez la variable « score » et validez. L'expression s'ajoute automatiquement dans la zone de texte. Vous pouvez y ajouter un petit texte comme "Score : " devant (n'oubliez pas le + entre les deux).

L'expression finale devrait ressembler à cela : `"Score : " + VariableString(score)`.

Vous pouvez valider en cliquant sur « Ok ». Enfin, validez l'action. L'événement final est comme ci-dessous.



FIGURE 4.13. – L'événement pour mettre à jour le texte.

i

Il n'y a pas de condition, car le texte doit être mis à jour continuellement (au cas où des pièces seraient collectées). En effet, **le résultat d'une expression est calculée à son affectation**. Donc, exécuter l'action une seule fois aura simplement pour effet d'afficher la valeur de « score » à un moment donné sans que cela soit mis à jour lors d'une modification de la variable.

Le compteur est désormais fonctionnel. Je vous invite à tester le jeu pour voir le résultat.

4.6. Amélioration du compteur

Le compteur de point n'est toujours pas parfait. En effet, il suffit de disposer les pièces d'une façon particulière pour faire apparaître un bug dans le comptage des points.

Vous pouvez essayer de reproduire ce problème en plaçant deux (ou plus) pièces à **exactement** la même hauteur (position Y) et de façon à ce qu'elles soient atteignables par le personnage quand il saute.



FIGURE 4.14. – Cas particulier de placement des pièces.

i

Vous pouvez utiliser le panneau de propriétés pour définir la même valeur dans la propriété « Y ».

Testez maintenant le jeu, et prenez les deux pièces en même temps. Vous constaterez que la variable « score » ne s'est incrémentée que de 100 points alors que deux pièces ont été collectées.

?

Pourquoi ?

Revenons sur l'événement qui s'occupe d'incrémenter le score.

4. Les variables et les expressions



FIGURE 4.15. – L'événement d'incrémentation du score.

Ce qui se passe dans ce cas est que la condition est bien validée au moment de la collision et « sélectionne » les deux pièces pour les actions. Ensuite, l'action « Supprimer l'objet **Piece** » va donc supprimer les deux instances de « Piece ». Ensuite, le son est joué. Enfin, l'action est exécutée. Mais cette action n'est exécutée qu'une seule fois, donc le score n'est incrémenté qu'une seule fois.

Pour corriger ce problème, nous allons utiliser une expression qui compte le nombre d'instances d'un objet prises en compte dans un événement : « Nombre d'objets » (qui est une expression numérique dans la catégorie « Autres fonctions > Objets »). L'expression réalisée prend la forme `Count(monObjet)` avec « monObjet » l'objet dont on veut compter le nombre d'instances.

Il suffit alors de multiplier le score à incrémenter par le nombre d'instances prises en compte de « Piece ». Remplacez donc `Variable(valeurPiece)` par `Variable(valeurPiece) * Count(Piece)`. N'oubliez pas de déplacer l'action de modification de variable avant la suppression des instances de « Piece » (sinon, `Count(Piece)` vaudra 0 car les instances auront déjà été supprimées).



FIGURE 4.16. – L'événement corrigé.

Le compteur devrait désormais fonctionner correctement dans toutes les situations.

Les pièces et le compteur de score fonctionnent désormais complètement.

Voici tous les événements de la scène, au cas où vous auriez un souci avec l'un d'eux.



FIGURE 4.17. – Les événements du jeu à la fin de la partie IV.

Le projet du jeu est disponible [ici](#) ↗ .

Contenu masqué

Contenu masqué n°3

4.7. Correction

L'événement doit agir quand le « Personnage » entre en collision avec une/des instance(s) de « Piece ». Nous allons donc utiliser la condition de collision qui se trouve dans la catégorie « Tous les objets > Collision ». Il suffit ensuite d'y mettre les deux objets dont nous souhaitons tester la collision.

Ensuite, il faut ajouter l'action « Supprimer l'objet **Piece** », qui permet de supprimer **des instances** de « Piece » de la scène (catégorie « Tous les objets > Objets »). D'après le principe de sélection des instances grâce aux conditions, seules les instances de « Piece » en collision seront supprimées.



FIGURE 4.18. – L'événement permettant de supprimer les instances de « Piece ».

[Retourner au texte.](#)

5. Annexes

Cette partie présente quelques notions supplémentaires afin d'améliorer et de finir le jeu. Les sections de cette partie sont **indépendantes** et peuvent être réalisées dans un ordre quelconque.

Note : d'autres annexes viendront compléter le tutoriel ultérieurement.

5.1. Exporter le jeu

Cette section présente comment procéder pour générer le jeu final à partir du projet.

5.1.1. La plateforme

Dans GDevelop, la « plateforme » correspond au format du jeu. Ainsi, suivant si vous avez choisi la plateforme « native » ou « HTML5 » au début du tutoriel, le jeu généré n'aura pas le même format :

- si vous aviez choisi « HTML5 », le jeu sera généré sous forme d'une page HTML5 avec les ressources (images et scripts javascript) ;
- si vous aviez choisi « natif », le jeu sera compilé dans un exécutable (suivant le système d'exploitation) avec les ressources à côté (images et événements du jeu compilé).

5.1.2. Changer la plateforme du jeu

Ne vous inquiétez pas, il est possible de changer à tout moment la plateforme utilisée par un projet. Il est d'ailleurs même possible d'activer les deux.



Toutes les conditions, actions et types d'objets ne sont pas disponibles dans les deux plateformes. Ainsi, la plateforme « HTML5 » possède bien moins de fonctionnalités que la plateforme « native ». Notez que **tout ce qui a été fait dans ce tutoriel est possible sur les deux plateformes.**

Pour gérer les plateformes possibles d'un projet, il suffit de double-cliquer sur « Extensions » dans le gestionnaire de projets. La fenêtre des extensions s'ouvre alors.



FIGURE 5.1. – La fenêtre des extensions.

Dans mon cas, la plateforme « HTML5 » est activée et la plateforme « native » est désactivée (grisée). En dessous, vous pouvez voir la liste des extensions de la plateforme sélectionnée et voir celles qui sont activées (elles l'ont été automatiquement lorsque nous avons ajouté de nouveaux types d'objets sur la scène).

Pour activer/désactiver une plateforme, il suffit de faire un clic droit sur elle (dans la liste du haut) et de cliquer sur « Utiliser cette plateforme » ou « Ne plus utiliser cette plateforme ».

Je vous laisse activer la ou les plateformes que vous souhaitez.

5.1.3. Génération du jeu final

Le jeu final peut être compilé via les boutons « Exporter pour le web » pour les jeux utilisant la plateforme HTML5 et « Compiler le projet en exécutable » pour les jeux natifs. Le logiciel vous proposera de définir où compiler/exporter le jeu.

Pour les jeux HTML5, GDevelop génère le code des événements en Javascript et exporte le tout avec un `index.html` et les ressources dans le dossier voulu. Vous pouvez tester le jeu en lançant `index.html`. Pour le mettre en ligne, il vous faudra un site web sur lequel uploader le dossier exporté en entier.



Google Chrome n'apprécie pas de lancer des scripts Javascript en local, vous ne pourrez donc pas tester le jeu final avant de l'uploader sur Internet.

Pour les jeux natifs, GDevelop va compiler le code des événements en binaire (par l'intermédiaire du C++) et va exporter cela avec un lanceur `VotreNomDeJeu.exe` ou `PlayLinux` (script pour Linux, à rendre exécutable) et les ressources. Libre à vous de zipper le dossier tout entier pour le redistribuer.



Si vous souhaitez compiler votre jeu natif pour plusieurs systèmes d'exploitation, il faudra faire les compilation avec GDevelop sur chacun d'eux.

Vous avez désormais un jeu de plates-formes simple mais fonctionnel. Vous pouvez vous inspirer de ce que vous avez appris pour rajouter des fonctionnalités à votre jeu : des ennemis, d'autres plates-formes (dont des *jumpthru*, par exemple). N'hésitez pas à consulter [la documentation/wiki du logiciel](#) ou à poster sur le [forum officiel](#) ou sur le forum de Zeste de Savoir.

Je vous souhaite bonne création avec GDevelop !

5. Annexes

i

Merci à Dominus Carnufex pour avoir relu mon tutoriel et corrigé les fautes et les erreurs typographiques.

Liste des abréviations

RAM Random Access Memory : la mémoire temporaire qu'utilise l'ordinateur pour traiter des données.. [21](#)