Projet de Développement Mobile : PlatVenture

1 Gestionnaire de version

Votre travail devra être sauvegardé dans le **gitlab** de l'université : **gitlab.univ-lorraine.fr**
Pour cela, créez un projet vide dans gitlab et ajoutez votre enseignant de TP comme membre relecteur (*reporter*).
Créez votre projet `libGDX` et activez le suivi de version dans `VCS / Enable Version Control Integration` en choisissant le système `git`. Ensuite, sélectionnez `VCS / Git / Remotes...` et ajouter l'URL de votre dépôt gitlab. Vos identifiants vous seront demandés, puis vous pourrez ajouter les fichiers sources (`java`) et effectuer le premier `commit / push` sur votre dépôt gitlab.

2 Description du projet

Dans ce projet, nous allons réaliser un **mini jeu de plateforme** :

- L'ensemble des données graphiques et sonores du jeu est disponible sur le site Arche du cours (**Données_Projet.zip**)
- Il doit fonctionner en mode `Desktop` et en mode `Android`
- L'application doit comporter 2 écrans différents :
  - Le premier est l'écran de présentation (image `Intro`), affiché pendant 3s (avec le son de victoire)
  - Le second est l'écran du jeu
- La vitesse d'affichage du jeu doit être réglée pour 60 images/s
- La largeur de la caméra orthographique est fixée à 16 unités du monde virtuel
- Chaque niveau est défini par un fichier texte contenant :
  - Les dimensions (largeur et hauteur) du tableau 2D représentant le niveau (2 premiers entiers du fichier)
  - Un temps imparti pour sortir du niveau et passer au suivant (3ème entier du fichier)
  - Les cases du tableau selon le codage suivant :

Élément	code(s)
Vide	`V`
Plateforme	`J` à `L`
Joyau	`1` à `2`
Sortie	`Z`

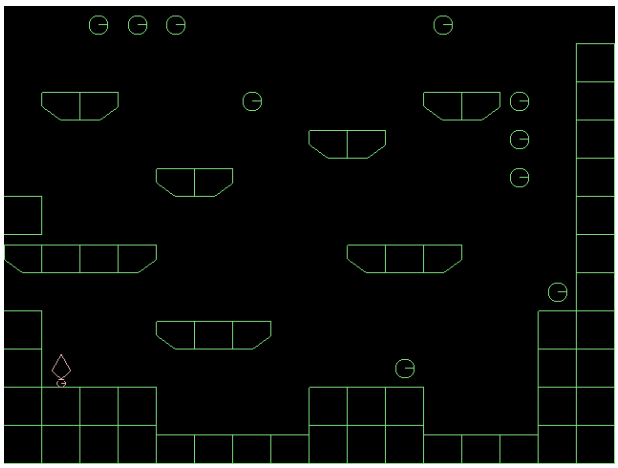
Élément	code(s)
Brique	`A` à `I`
Eau	`W`
Personnage (position initiale)	`P`

  - Une image de fond représentant le décor du niveau (nom du fichier image)
- La taille d'une case du tableau de niveau correspond à une unité du monde virtuel
- Le contrôle du personnage se fait :
  - Mode `Desktop` : en utilisant les flèches du clavier (flèche vers le haut pour sauter)
  - Mode `Android` : en appuyant sur l'écran tactile dans la moitié gauche pour aller à gauche et dans la moitié droite pour aller à droite (deuxième contact tactile pour sauter)
- Le jeu démarre au niveau 1 selon la configuration de départ du niveau
- Les textes affichés avec la police fournie dans les données, doivent avoir les propriétés suivantes :
  - La taille doit suivre un ratio de 60 pixels pour une largeur d'écran de 1024
  - La couleur est jaune avec une légère transparence (75% opaque)
  - Les bords sont noirs et leur épaisseur suit un ratio de 3 pixels pour une largeur d'écran de 1024
- Les textes sont affichés en haut de l'écran :
  - Le temps restant pour le niveau courant doit être centré en largeur
  - Le score du joueur doit être affiché à droite
- Les joyaux doivent être animés (séquences `Gem1.png` et `Gem2.png`)
- Le personnage est représenté par une image fixe selon son action en cours :

Inactif	Saut	Chute	Course
`Idle__000.png`	`Jump__006.png`	`Jump__008.png`	`Run__003.png`

- D'autres images sont fournies, permettant d'animer les actions du personnage (bonus)

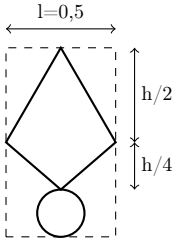
Les figures suivantes montrent les versions debug et finale du jeu au démarrage du premier niveau.



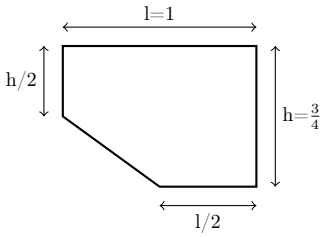
3 Dynamique du jeu

La dynamique du jeu est définie comme suit :

- La gravité (verticale vers le bas) est fixée à 10 unités / s^2
- Le joueur peut se déplacer à gauche et à droite et sauter
  - Chaque mouvement est traduit par une force appliquée au centre du personnage
  - L'intensité est fixée à 1 en x et à 40 en y
- Lors d'un saut, le joueur ne peut pas changer sa trajectoire
- Les propriétés physiques du joueur sont les suivantes :
  - Largeur : 0,5 (hauteur déduite des proportions de la texture)
  - Forme : diamant + cercle (rayon $\frac{1}{8}$ de la hauteur) →
  - Densité : 0,5
  - Restitution : 0,1
  - Friction : 0,5



- Les briques (et plateformes) ont les propriétés physiques suivantes :
  - Forme carrée unitaire (briques), rectangulaire (plateforme centrale)
  - Forme de rectangle avec biseau pour les extrêmités des plateformes →
  - Densité : 1
  - Restitution : 0,1
  - Friction : 0,25



- Les plateformes et l'eau étant rectangulaires, elles sont positionnés au bas de leur case du tableau de niveau
- Les joyaux et l'eau sont définis par des corps physiques *traversables* (`sensor`)
- Les comportements suivants du joueur doivent être mis en place :
  - Contact avec un joyau : le nombre de points (1 ou 2) correspondant au joyau est ajouté au score et le son correspondant est joué
  - Contact avec une brique d'eau : le joueur a perdu (voir la séquence de perte ci-dessous)
  - Fin de contact avec la flèche de sortie : le joueur a gagné seulement s'il n'est plus dans la zone de jeu au moment où il quitte la flèche (voir la séquence de victoire ci-dessous)
  - Sortie de la zone de jeu sans passer par la flèche : le joueur a perdu (voir la séquence de perte ci-dessous)
  - Contact avec une brique en ayant une vitesse de norme > 3 : un son de collision est émis
- Lorsque le temps imparti arrive à 0, le joueur a perdu (voir la séquence de perte ci-dessous)
- Séquence de perte :
  - Affichage d'un texte au centre de l'écran indiquant la perte
  - Son de perte
  - Délai de 2s et redémarrage du niveau (courant ou 1er) avec score à 0
- Séquence de victoire :
  - Affichage d'un texte au centre de l'écran indiquant la victoire
  - Son de victoire
  - Délai de 2s et chargement du niveau suivant (le 1er si on était au dernier niveau)

Une **vidéo de démonstration** du jeu est disponible sur le site Arche du cours.

4 Recommandations et indications de développement

Pour ce projet, les indications sont volontairement succinctes en ce qui concerne les étapes de développement afin de vous mettre en situation réelle vis-à-vis d'un travail respectant un cahier des charges.

Organisez-vous au mieux pour réaliser le jeu demandé et éventuellement proposer des extensions.

Voici juste quelques conseils utiles :

- ☞ Vous pouvez consulter la page `https://libgdx.com/dev` pour démarrer votre projet.
- ☞ Lors de la création de votre projet `libGDX`, sélectionnez les versions `Desktop` et `Android`, et incluez les options `Freetype` et `Box2d`.
- ☞ De même, une fois votre projet créé, il est recommandé d'utiliser l'API 23 au maximum pour que le jeu puisse être testé sur les tablettes de l'université.
- ☞ Pour développer ce jeu, il est nécessaire de commencer par sa **modélisation UML** :
  - — Quels sont les éléments qui nécessitent la définition d'une classe ?
  - — Pour faciliter la gestion d'une partie, il est nécessaire d'en distinguer les différentes phases (états).

  ⚠ **Le diagramme de classes vous sera demandé lors de la présentation finale.**
- ☞ La majeure partie du développement peut (et devrait) se faire sans l'affichage des textures, en travaillant directement avec l'affichage des objets physiques `Body` via un `Box2DDebugRenderer`.
- ☞ L'affichage et le placement correct des textures sur les objets constitue l'habillage visuel du jeu, et peut être réalisé indépendamment de la dynamique du jeu.
- ☞ Utilisez les logs ou affichages de texte pour vous aider à débugger
- ☞ Les *tâches* permettent de planifier des actions répétitives ou de gérer des comptes à rebours.
- ☞ N'hésitez pas à modifier certains paramètres du jeu lors de vos tests. Par exemple, pour tester la fin de partie, vous pouvez régler le temps de jeu à seulement quelques secondes.
- ☞ Il est important de noter que la méthode `render()` de votre écran principal de jeu sera appelée continuellement pendant tout le déroulement de l'application. Cela implique qu'elle est également appelée pendant les phases intermédiaires (attente, fin de partie,...) et cela doit être pris en compte dans votre gestion du jeu.
- ☞ Les collisions sont automatiquement gérées par le moteur physique. Cependant, il ne faut pas oublier de mettre celui-ci à jour à chaque itération du jeu.
- ☞ Le recours au `UserData` des `Body` est recommandé pour la reconnaissance des objets lors des contacts.