CodePilot - Spécifications techniques et informatiques

@Alexandre Malfreyt, @Lucas Monnier, @Baptiste Rey

Description des technologies utilisés

La technologie choisie pour le développement du jeu CodePilot est le moteur de jeu Godot, combiné avec le langage de programmation GDScript. Voici une description détaillée de cette technologie et les raisons pour lesquelles elle a été choisie :

Moteur de jeu Godot

Nous avons choisi Godot comme moteur de jeu pour CodePilot. Il s'agit d'un moteur open-source offrant une grande variété de fonctionnalités pour le développement de jeux 2D et 3D. Nous avons été attirés par sa simplicité d'utilisation, sa flexibilité et sa puissance.

Voici quelques caractéristiques qui nous ont convaincus :

- **Multiplateforme** : Godot prend en charge le développement de jeux pour une multitude de plateformes, y compris Windows, Linux, macOS, iOS, Android et HTML5. Cette polyvalence était essentielle pour nous, car elle nous permet de toucher un large public avec notre jeu.
- **Simplicité d'utilisation** : Nous avons trouvé que Godot était très facile à prendre en main, avec une interface utilisateur intuitive et une courbe d'apprentissage rapide. Cela le rend idéal pour notre équipe et surtout pour un petit projet scolaire qui ne nous laisse pas le temps d'apprendre à utiliser un gros moteur graphique bien plus complexe comme Unreal Engine.
- **GDScript** : GDScript, le langage de script spécifique à Godot, nous a également séduits. Il est conçu pour être facile à apprendre et à utiliser, grâce à sa syntaxe claire et concise. Sa similarité avec Python était un autre point positif pour nous, car plusieurs membres de notre équipe sont familiers avec ce langage.
- **Open-source** : Godot est entièrement open-source, ce qui signifie que son code source est accessible et modifiable par tous.

Langage de programmation GDScript

GDScript est le langage que nous avons choisi pour le développement de CodePilot. Voici pourquoi :

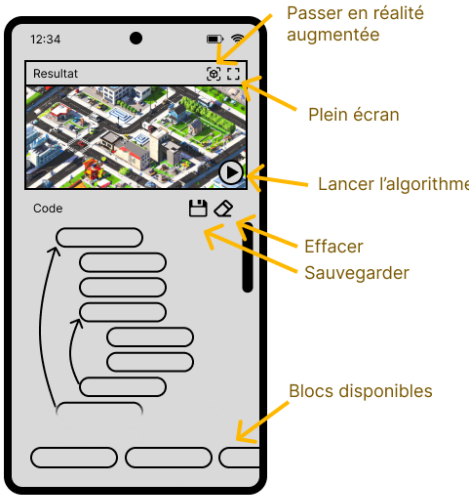
- **Facilité d'apprentissage** : Basé sur Python, GDScript est facile à apprendre pour la plupart d'entre nous. Sa syntaxe simple et ses conventions claires en font un choix pertinent compte tenu du temps disponible pour le développement du jeu.
- **Intégration transparente avec Godot** : GDScript est conçu pour fonctionner avec Godot, ce qui lui donne un accès complet à toutes les fonctionnalités du moteur de jeu. Cela nous permettra de créer un jeu avec des mécaniques relativement complexes tout en conservant une structure de code claire et organisée.
- **Performances optimisées** : Bien que GDScript soit un langage interprété, il est optimisé pour le moteur de jeu Godot, ce qui lui permet d'offrir des performances compétitives même pour les jeux complexes.
- **Courbe d'apprentissage douce** : Comme notre équipe a un niveau de compétence et d'experiencec différents en ce qui concerne le développement de jeux vidéos, il était important de choisir une technologie avec une courbe d'apprentissage douce. Godot et GDScript permettent donc à tous les membres de contribuer efficacement au projet.

Charte graphique et maquettes d’écran

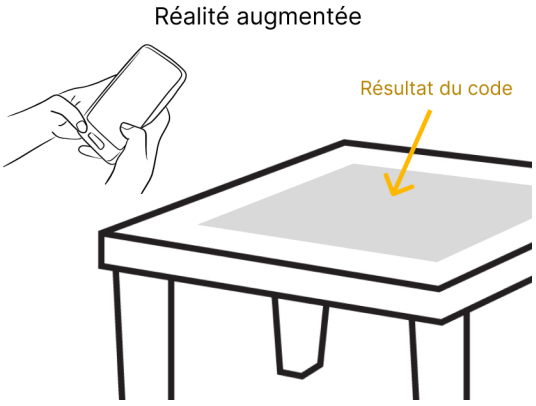
Maquettes d’écran

Voir le fichier original :
https://www.figma.com/file/WWxQ7Wwiarq0K4SKri3f5b/SA%C3%89401---CodePilot?type=design&node-id=1%3A46&mode=design&t=YcLfzlfm4ne2Z43g-1

1. Mobile

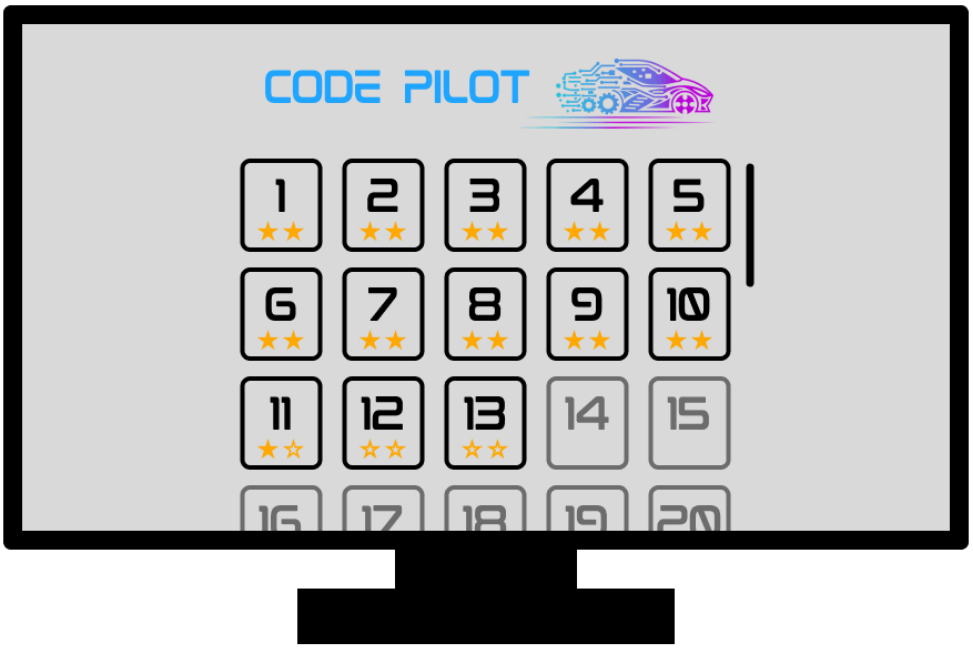


Interface du jeu (Mobile)

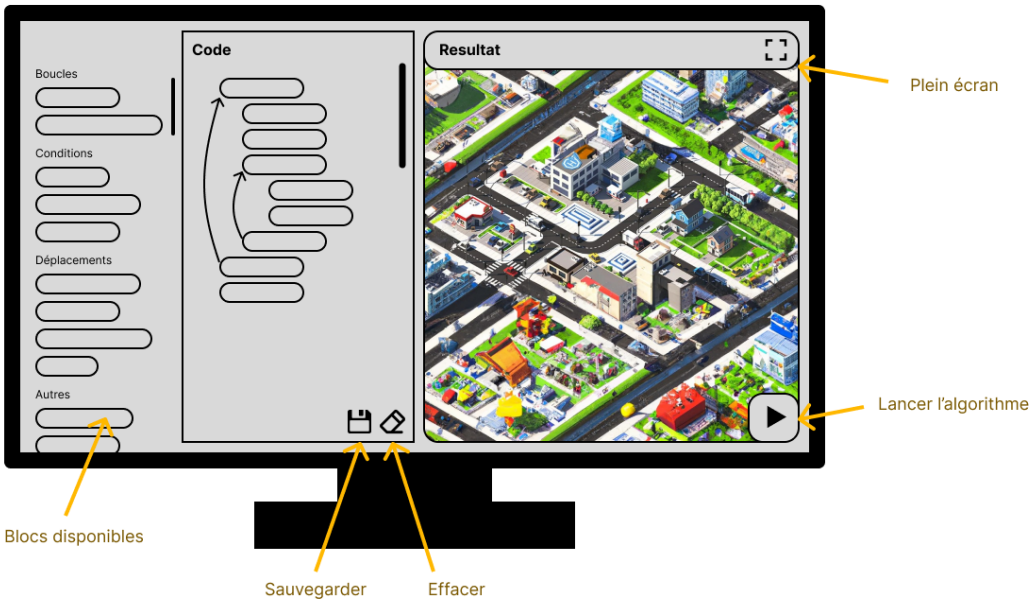


Réalité augmentée (facultatif)

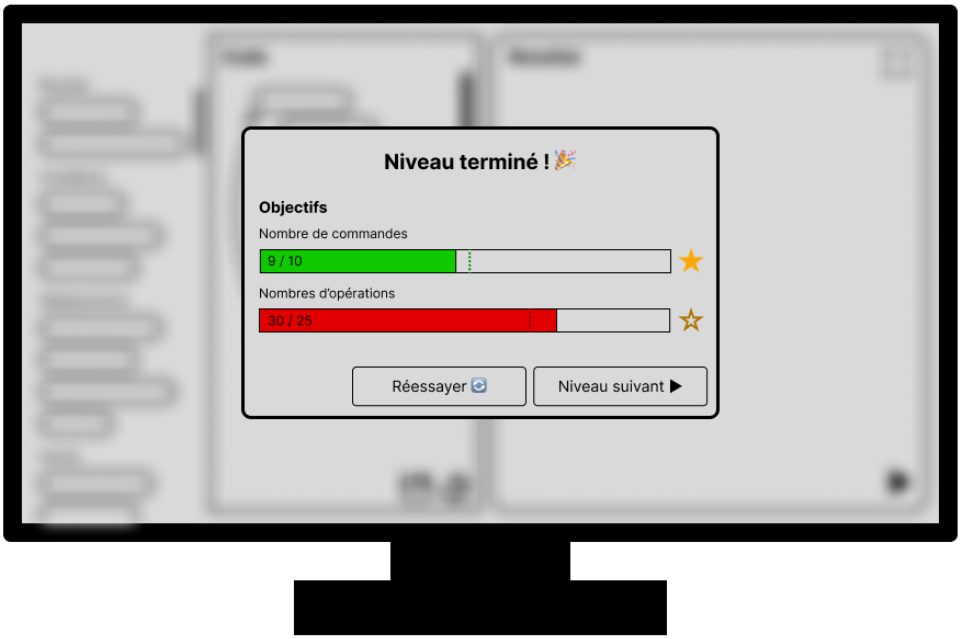
2. PC



PC - Menu



Interface du jeu (PC)



Niveau terminé (PC)

Description de la charte graphique

Couleurs

La charte graphique de CodePilot utilise des couleurs vives et engageantes pour captiver l'attention des utilisateurs tout en reflétant l'aspect ludique et éducatif du jeu. Voici les couleurs principales utilisées :

- Charbon : **#000000**
- Rose fluo : **#ff0080**
- Pourpre : **#c800ff**
- Violet foncé : **#3d1f63**
- Bleu éclairé : **#437899**
- Bleu ciel : **#17b2ff**

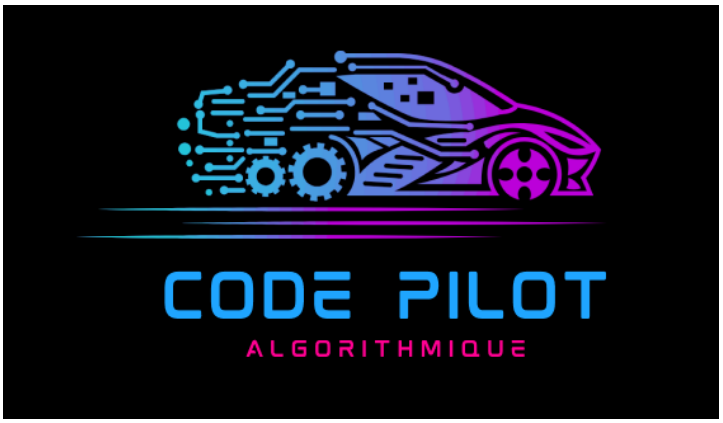
Ces couleurs sont choisies pour leur contraste et leurs capacité à attirer l'œil tout en restant harmonieuses.

Interface utilisateur

L'interface utilisateur de CodePilot est conçue pour être intuitive et accessible, afin de faciliter l'expérience de jeu pour les utilisateurs de tous niveaux. Les éléments d'interface utilisent les couleurs de la charte graphique pour créer une ambiance cohérente et attrayante.

Logo

Le logo de CodePilot représente une voiture se déplaçant sur une route, avec des éléments de code s'entrelaçant, symbolisant le parcours d'apprentissage en algorithmique que propose le jeu. Les couleurs vives et le design dynamique du logo contribuent à renforcer l'aspect ludique et éducatif du projet.



Logo “CodePilot”

Le logo de CodePilot est visible ci-dessus, avec son design distinctif représentant la voiture et les éléments de code. Il est également animé pour ajouter un élément de dynamisme et d'attrait visuel supplémentaire.

Animation du logo

L'animation du logo de CodePilot apporte une dimension supplémentaire à l'identité visuelle du projet, en mettant en mouvement la voiture et les éléments de code pour créer un effet dynamique et captivant.

https://prod-files-secure.s3.us-west-2.amazonaws.com/a1f4ee1e-a651-47ff-855c-afc2e798c091/e7dd8932-54ae-4f51-93c7-0190168156f3/Auto_tech.mp4

Animation du logo “CodePilot”

Technologies associées pour produire/gérer les contenus graphiques ou interface

- Pour la conception graphique du logo, nous avons utilisé **Adobe Photoshop**.
- Pour l'animation du logo, nous avons fait appel au logiciel **Adobe After Effects**.
- Pour la création des différents assets vectoriels du jeu en 2D (interface, textures, ...), nous avons utilisé **Figma.**
- Pour la création de modèles 3D, nous avons utilisé **Autodesk Fusion 360** et **ThinkerCad.**
- Pour l'affichage de l'interface graphique 2D et des modèles 3D dans le jeu, nous utiliserons la librairie par défaut de Godot.q

---

Descriptif technique

Diagramme de classes (UML)

Dans Godot, nous comptons principalement sur l'utilisation de classes de base préprogrammées telles que les objets 3D, les niveaux, etc., que nous allons modifier. La hiérarchie et la structure de ces classes sont déjà établies. De plus, le langage de programmation de Godot (GDScript) n'est pas fortement orienté objet. Par conséquent, l'utilisation d'un diagramme de classe ne semble pas appropriée pour ce projet.

Génération procédurale des niveaux

Les joueurs auront la possibilité de générer des niveaux de manière procédurale pour augmenter la durée de vie du jeu. C'est-à-dire des niveaux générés aléatoirement avec des règles prédéfinies.

Chaque niveau aura une "graine" d'aléatoire (aussi appelée "seed") sous la forme d'une suite de chiffre. Le fait d'entrer deux fois la même seed créera le même niveaux, même sur deux copies du jeu différentes. Cela permettra aux joueurs de se partager des niveaux qu'ils ont aimés.

Modèle de stockage des données

Notre jeu stockera seulement les données basiques associées à chaque niveau et donc n'utilisera pas de base de données relationnelle complexe en SQL.

Données de sauvegarde

Les données seront à la place stockées en local dans des fichiers au format `.codepilot` (simples fichiers éditables dans un logiciel d'édition de texte avec un format personnalisé pour nos besoins), qui contiendront les données de sauvegarde du jeu (c'est-à-dire les algorithmes écrits par les joueurs) dans un format similaire à du pseudo code.

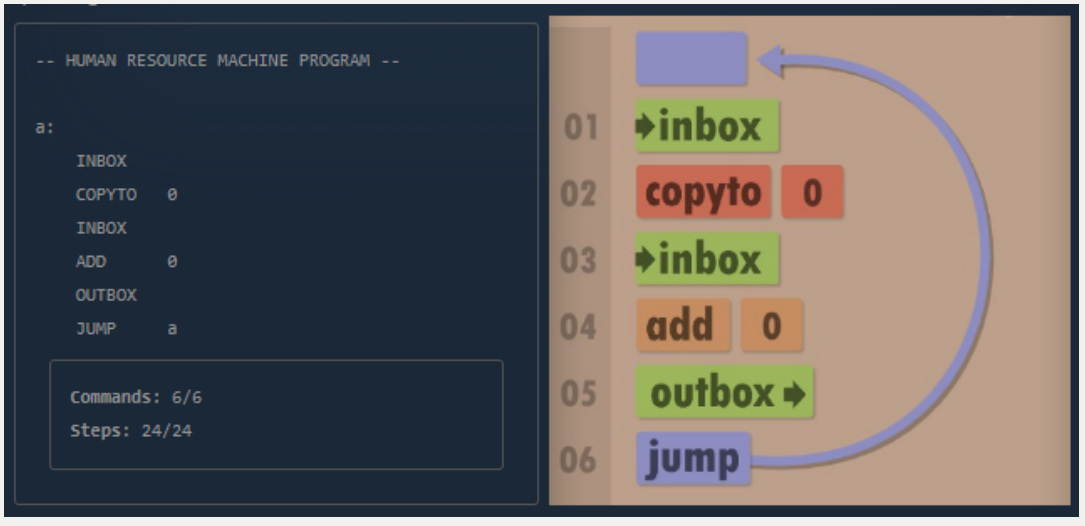
Lorsque le joueur enregistrera un algorithme créé dans l'interface du jeu, un fichier de sauvegarde sera généré avec son algorithme dans un syntaxe lisible par un humain. Puis lorsque le même niveau sera réouvert plus tard, ce fichier sera interprété pour être ouvert dans l'interface du jeu.

Exemple de fichier de sauvegarde (syntaxe non définitive) : `saves/level1.codepilot`

```
start
save num = 5
save x = 10
save y = 15
repeat num
    goto x y
    copy
    goto 0 0
    paste
end
```

💡 Inspiré du jeu Human Resource Machine qui propose une fonctionnalité similaire

(à gauche : le fichier de sauvegarde, à droite : l'interface du jeu)



Données liées au niveaux

Les données des niveaux en eux même seront stockées stockées dans un format `.json`. Cela concerne à la fois les niveaux conçus par nous mêmes, et les niveaux générés procéduralement.

Données des niveaux non procéduraux :

Exemple de niveau (modèle de données non définitif) : `levels/level1.json`

```
{
    "name": "Niveau 1",
    "difficulty": "1",
    "id": "001",
    "required_score": "2000",
    "best_score": "3456",
    ...
    "data" : [
        ...
    ]
}
```

Diagramme de déploiement (UML)

Le jeu fonctionnera uniquement en local (sur un seul appareil) avec un emplacement sur le disque dur pour les fichiers de sauvegarde qui seront accédés en lecture et en écriture par le code source du jeu.