BERNARD Léo
ATZA Maxime

PLANNING POKER

Conception agile

PRESENTATION :

Dans ce projet, nous vous présentons un jeu de planning poker réalisé avec le moteur de jeu Godot.

Godot est un logiciel libre et multiplateforme qui permet de créer des jeux en 2D ou 3D de manière simple et intuitive. Godot offre de nombreuses fonctionnalités, telles que la programmation visuelle, l'éditeur de scènes, un moteur physique, et bien d'autres.

Nous avons utilisé Godot pour concevoir une interface graphique attrayante et interactive pour notre jeu de planning poker.

Notre jeu est conçu pour être un multijoueur local, c'est-à-dire que les joueurs partagent le même appareil et jouent à tour de rôle.

Le jeu permet de créer des joueurs, de choisir des cartes par joueur, de suivre une partie jusqu'à ce que tous les joueurs soient d'accord sur l'estimation finale et de sauvegarder les données dans un fichier Json.

Le fichier Json nous permet de sauvegarder les données de manière structurée.

Notre but est de faire un outil de planning poker pratique, fun et adapté aux équipes agiles.

CHOIX PATRON DE CONCEPTION :

Pour notre projet, nous avons choisi des patrons de conception adaptés à nos besoins et à Godot.

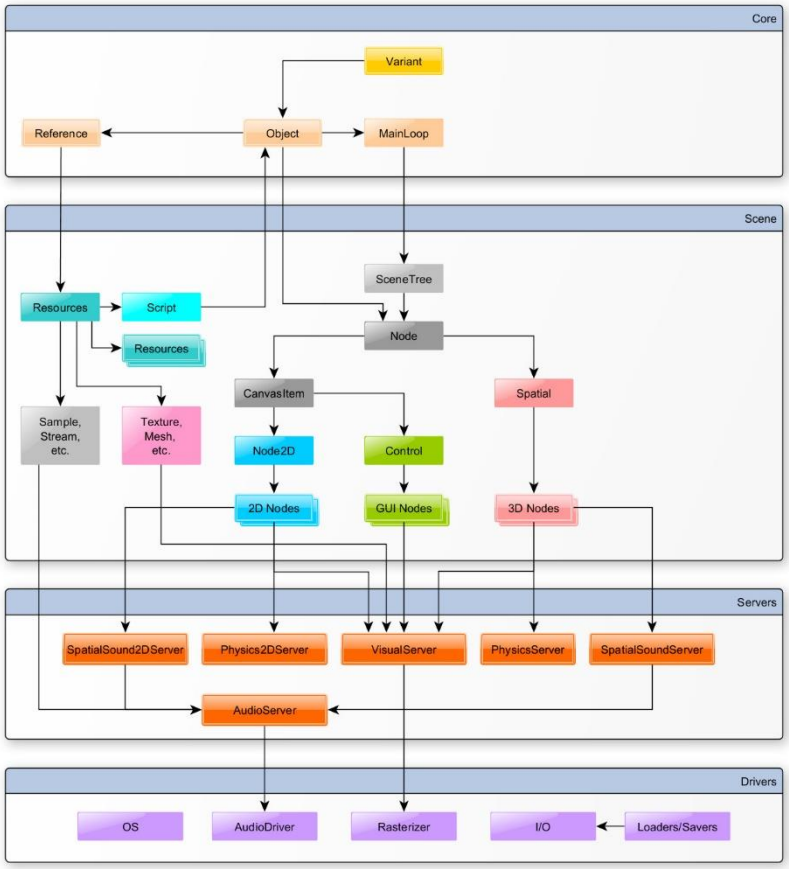
Au niveau architectural, nous avons suivi le modèle natif de Godot, qui repose sur une architecture en couches. Ce modèle nous permet de séparer les différentes parties du système en différentes classes. Ce patron nous permet également d'hériter des classes de Godot.

Au niveau logiciel, nous avons principalement utilisé le patron Décorateur en héritant des classes Node, Node2D, Area2D et Control de Godot. Ce patron nous permet d'ajouter des fonctionnalités à un objet sans modifier sa structure. Il nous permet aussi de redéfinir les fonctionnalités des classes parentes.

Au niveau visuel, nous avons opté pour une architecture visuelle en navigation par pages et glisser-déposer. Ce type d'architecture offre une expérience utilisateur intuitive et interactive, en permettant aux utilisateurs de naviguer facilement entre les différentes vues du jeu et de manipuler les éléments avec le curseur de manière fluide et ludique.

EXPLICATION PATRON DE CONCEPTION :

Voici un diagramme illustrant la structure et les relations entre les classes natives de Godot :



Godot est un moteur de jeu qui se base sur une architecture en couches. Cette architecture permet de séparer les différentes parties du système en fonction de leur niveau d'abstraction et de leur rôle. Il existe quatre couches principales dans Godot :

- La couche Core contient les classes de base qui définissent les types de données et les objets utilisés par le moteur. Parmi ces classes, on trouve la classe Object, qui est la classe mère de tous les autres objets, la classe Reference, qui gère le comptage de références pour les objets, et la classe Variant, qui représente une valeur de n'importe quel type.

- La couche Scene est composée des classes qui permettent de créer et de manipuler les éléments du jeu, comme les entités, les interfaces ou les animations. La classe mère de cette couche est la classe Node qui définit tous les éléments de la scène. Il existe différents types de Nodes selon leur fonctionnalité, ceux qui nous intéressent sont : Les Nodes graphiques (Node2D, Control), les Nodes physiques (Area2D) et les Nodes audio (AudioStreamPlayer).

- La couche Servers contient les classes qui gèrent les aspects bas-niveau du moteur, comme le rendu graphique, le son ou la physique. Ces classes sont appelées serveurs, car elles fournissent des services aux autres classes du moteur.

- La couche Drivers contient les classes qui gèrent l'interaction avec le système d'exploitation et les périphériques matériels, comme la souris, le clavier, les enceinte... Ces classes sont appelées pilotes, car elles font le lien entre le moteur et les ressources externes.

Notre projet hérite de cette conception en agissant principalement dans la couche concernant la scène.

Nous avons choisi d'utiliser le patron Décorateur pour créer nos propres classes en héritant des classes Node, Node2D, Area 2D ou Control. Ce patron nous permet d'ajouter des fonctionnalités à un objet sans modifier sa structure. Il nous permet aussi de redéfinir les fonctionnalités des classes parentes.

Par exemple, nous avons créé la classe Carte, qui hérite de la classe Node2D, pour représenter une carte du jeu. Nous avons ajouté des propriétés comme la valeur, le placement, la taille… Et des méthodes comme valuechange, drag, drop... Nous avons également redéfini la méthode _process pour gérer l'animation de la carte.

Concernant le visuel, nous avons opté pour une architecture visuelle en navigation par pages et glisser-déposer. Ce type d'architecture offre une expérience utilisateur intuitive et interactive, en permettant aux utilisateurs de naviguer facilement entre les menus, la partie et ses étapes. Cette navigation se fait avec l'utilisation de boutons. Le but a été de simplifier au maximum l'interface pour une utilisation simple et agréable.

Nous avons utilisé la classe Control pour créer l'interface du menu. Nous avons utilisé la classe Node2D pour créer les éléments graphiques, comme les cartes, la main du joueur et le tableau de résultats. Nous avons utilisé la classe Area2D pour la zone de dépôt des cartes.

INTEGRATION CONTINUE :

Pour gérer notre projet, nous avons utilisé GitHub et Trello, deux outils qui facilitent la collaboration et l'organisation des tâches.

Nous avons utilisé GitHub pour déposer les avancées du projet à chaque nouveauté et pour bénéficier de l'intégration continue et posséder des versions de notre projet.

Trello est un outil qui permet de créer des tableaux avec des listes et des cartes représentant les tâches à réaliser. Nous l'avons utilisé pour organiser tout ce qui était à faire et ce qui a était fait.

Trello nous a permis de gérer les tâches de manière logique et organisé.