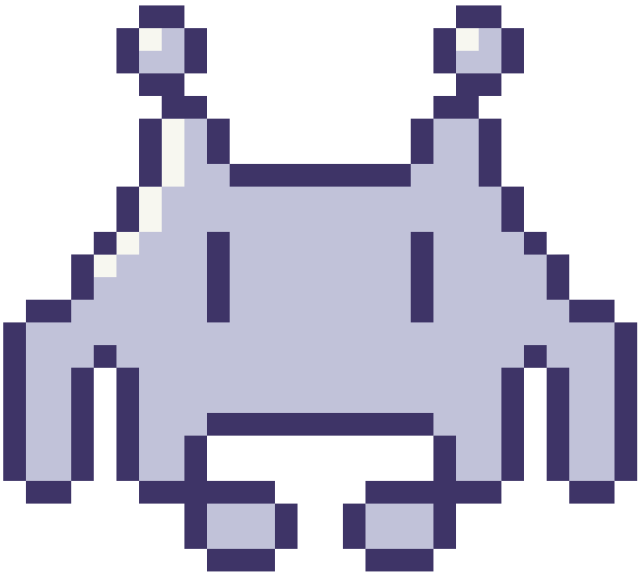
{EPITECH.}

ARCADE

a retro platform



Contributeurs :

Pierre Bourgery
Léo Maurel
Matthias Soual
Paul Arbez
Raphael Castres
Théo Bary

Fonctionnement :

Communication Graphique/jeu :

Système de paquets :
Le jeu et le graphique communiquent grâce à un système de paquets. La norme des paquets est la suivante:
Paquet ouvrant => NB paquets, pour NB informations => Paquet fermant.
Les paquets existants sont DATA, DRAW, DRAW_STRING, CHANGE_COLOR, SET_GAME, SET_GRAPHIC stockées dans un enum, dans cet ordre ci.

Les paquets sont définis comme suit: std::tuple<EventType, EventData> avec EventType, l'enum des paquets, et EventData, un typedef sur std::variant<std::nullopt_t, std::string, std::size_t, short, double, bool, unsigned int>

Voici la liste de tous les paquets disponibles:

- **DATA** pour les informations des paquets.

- **DRAW** pour dessiner un carré de 4 x 4. DRAW encapsule 3 DATA, 2 std::size_t puis un short. Les size_t sont pour la position en x, la position en y et le short, possédant 16 bits correspond aux sous parties du carré à dessiner, la première sous partie étant le plus grand bit.

- **DRAW_STRING** pour écrire du texte, il encapsule 4 DATA, 2 std::size_t, une std::string et un bool. Les size_t pour la position en x, la position en y, la string à écrire, et le booléen qui permet de savoir si la string doit être en surbrillance.

- **CHANGE_COLOR** permet de changer la couleur pour tous les DRAW à venir jusqu'au prochain CHANGE_COLOR, il encapsule 1 DATA, qui est un unsigned int. L'unsigned int est défini comme suit, les 8 bits avec la plus grande valeur correspondant a la composante alpha, les 8 suivants pour le rouge, ensuite le vert et le bleu.

- **SET_GAME** et **SET_GRAPHIC** permettent de changer les bibliothèques de jeu/graphique, ils encapsulent 1 DATA, qui est une std::string qui correspond au chemin vers la bibliothèque à charger.

Gestion des inputs:
Les inputs sont stockés dans une map contenant un événement (std::string, ex: “UPWARD”) et son état (booléen, true ou false). Cette map est créée dans le “core”.

Elle est stockée dans un shared_ptr comme suit :
std::shared_ptr<std::map<std::string, bool>>.

Le shared_ptr doit être envoyé dans chaque bibliothèque pour lier les inputs entre le graphique et le jeu. Tant qu’une touche appartenant à cette map est enfoncée, sa valeur est égale à true, une fois relâchée, elle est égale à false.

Voici les inputs qui sont enregistrées :

```
{"UP", false},
{"DOWN", false},
{"LEFT", false},
{"RIGHT", false},
{"SPACE", false},
{"ENTER", false},
{"ESC", false},
{"TAB", false},
{"BACKSPACE", false},
{"A", false},
{"Z", false},
{"E", false},
{"R", false},
{"T", false},
{"Y", false},
{"U", false},
{"I", false},
{"O", false},
{"P", false},
{"Q", false},
{"S", false},
{"D", false},
{"F", false},
{"G", false},
{"H", false},
{"J", false},
{"K", false},
{"L", false},
{"M", false},
{"W", false},
{"X", false},
{"C", false},
{"V", false},
{"B", false},
{"N", false},
{"F1", false},
{"F2", false},
{"F3", false},
{"F4", false}
```

Implémenter une bibliothèque graphique/jeu :

Implémentation des bibliothèques de jeu :
Les bibliothèques de jeu sont chargées par le "core" via la fonction "createGame". Elle prend en paramètre le shared_ptr de la map des touches, ainsi qu'une référence à un int et une std::string, correspondant au nom du joueur et à son dernier score, qui sont des informations créées dans le "core".
Les bibliothèques de jeu doivent avoir une méthode "tick", qui prend en paramètre un double correspondant au temps entre chaque appel de cette méthode.
Cette méthode renvoie une file de paquets (std::queue<std::tuple<EventType, eventData>>). Cette file correspond aux différentes informations qui devront être traitées par la bibliothèques graphique ainsi que le "core".

Implémentation des bibliothèques graphiques :
Les bibliothèques graphiques sont chargées par le "core" de la même manière que pour les bibliothèques de jeu.
Les bibliothèques graphiques contiennent trois méthodes appelées par le "core".
La méthode "updateKeybinds", qui ne prend aucun paramètre, modifie les booléens dans la map d'inputs, en fonction de si la touche est appuyée ou non. Cette méthode ne renvoie rien.
La méthode "readEvent" prend en argument la file renvoyée par le jeu. C'est cette méthode qui va lire les paquets et enregistrer les informations pour ensuite les afficher. Cette méthode modifie l'état du graphique sans renvoyer de valeur.
La méthode "draw" ne prend pas de paramètres et dessine l'état du jeu qui a été mis à jour par "readEvent" et renvoie une file d'évènement.

Gestion des bibliothèques dans le core :
Pour changer de bibliothèque graphique, certaines touches ont été assignées par défaut :

- "F2" pour changer de bibliothèque de jeu,
- "F3" pour la bibliothèque graphique.
- "ÉCHAP" pour mettre fin à l'exécution,
- "F1" pour recharger le jeu servant de menu.

Ces touches ne devraient pas être utilisées par les jeux, si elles le sont, cela mènera à des comportements inattendus.
Pour le jeu servant de menu, des paquets ont été créés pour lui permettre de communiquer avec le "core" un changement de bibliothèque graphique ou de jeu. Pour effectuer ce changement, il faut envoyer un paquet “SET_GAME” ou “SET_GRAPHIC” avec le chemin de la bibliothèque à charger.

Gestion des bibliothèques par le Core: