

ING1

Projet d'informatique - semestre 2



ECE WORLD



Antoine Hintzy - 2022-2023

# ECE World

Ce semestre, nous vous proposons de créer votre propre parc d'attraction, en utilisant la bibliothèque graphique **ALLEGRO**.

Par équipe, vous créerez votre parc proposant au moins une attraction par étudiant.e, parmi les nombreuses attractions proposées dans ce cahier des charges.

Vous avez le choix du thème de votre parc, et pouvez adapter l'apparence de chaque attraction comme bon vous semble.

Chaque attraction proposée est plus ou moins difficile à mettre en place, vous serez évalués sur la qualité de votre implémentation, la fluidité de votre jeu, son apparence, et le niveau des attractions que vous déciderez d'implémenter (le niveau de difficulté vous est donné). Chaque étudiant doit coder de A à Z au moins une attraction issue de ce cahier des charges, puis, peut ensuite implémenter jusqu'à deux attractions supplémentaires (du CDC ou non). Le jury estimera leur difficulté lors la soutenance.

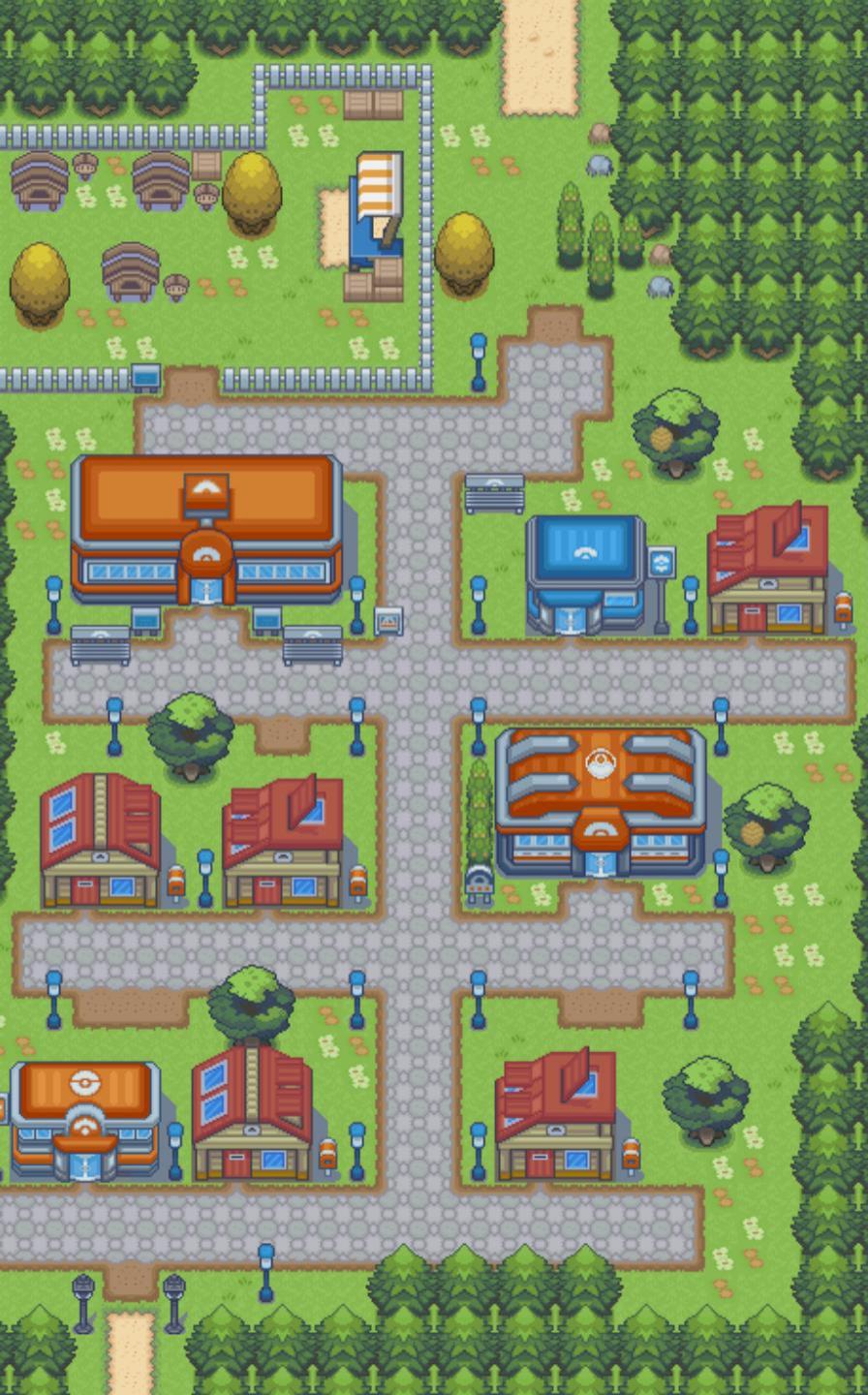
# Joueurs

Le jeu se jouera à deux joueurs (*ou plus si vous le souhaitez*), représentés chacun par un personnage.

Les joueurs choisissent à tour de rôle quelle est la prochaine attraction.

*Il faudrait idéalement que chaque joueur nomme son personnage, soit avec `scanf` dans le terminal, soit à l'aide du clavier directement dans le jeu, soit à la souris (écriture manuscrite, signature).*



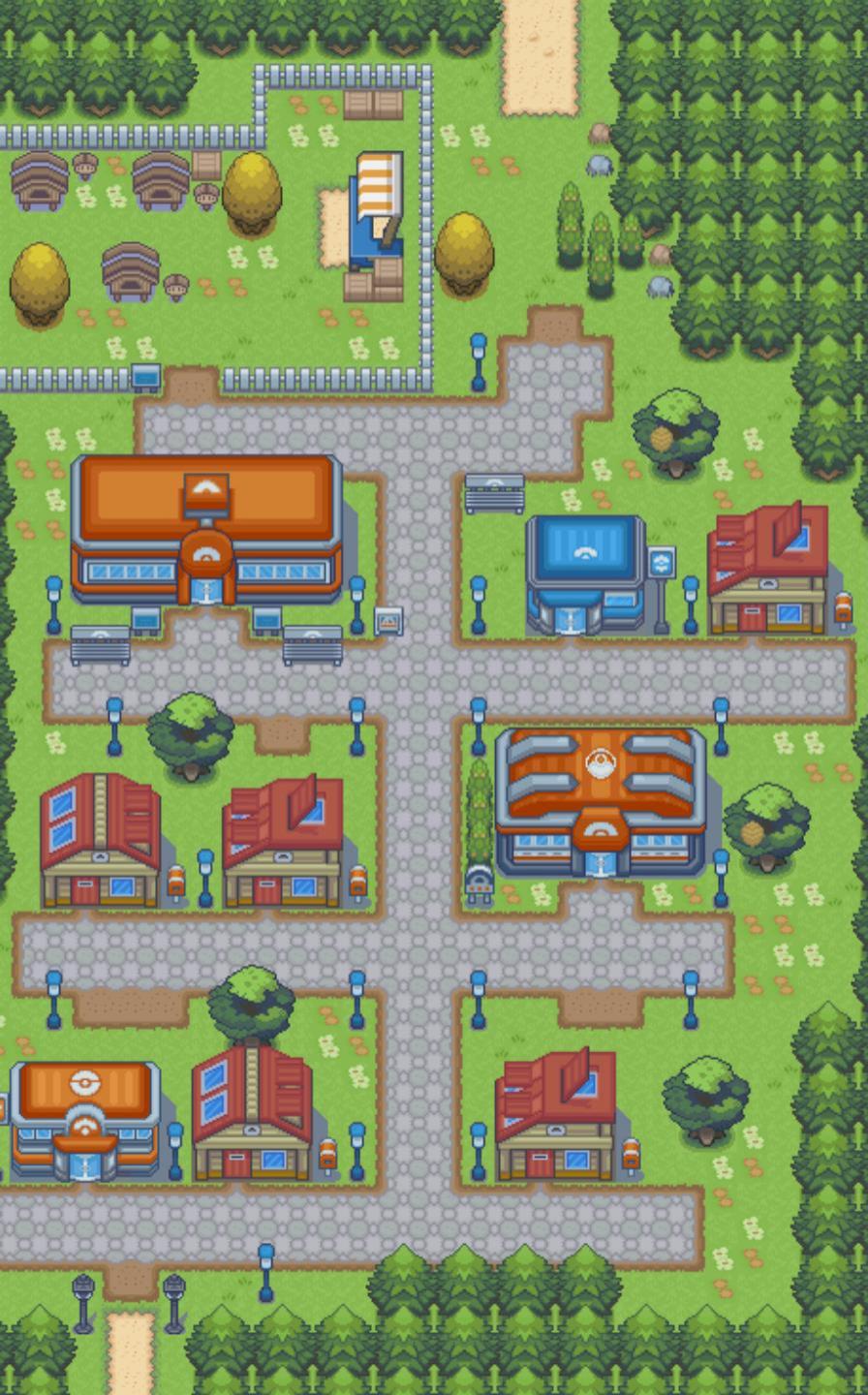


# Carte

Pour accéder à une attraction, le joueur qui joue déplace au clavier son personnage sur une carte.

Chaque attraction est représentée sur la carte par un "bâtiment" (qui ressemble si possible à l'attraction).

Lorsque le personnage rentre dans un bâtiment, le mini-jeu correspondant se charge. Une fois le jeu terminé (victoire/défaite), le personnage sort du bâtiment et c'est au tour du joueur suivant de choisir l'attraction.



# Carte

La carte fait office de menu (par exemple, la porte de sortie du parc permet de quitter le jeu, un panneau d'affichage permet d'accéder aux statistiques, etc.).

Sur la carte, les joueurs devront être capables :

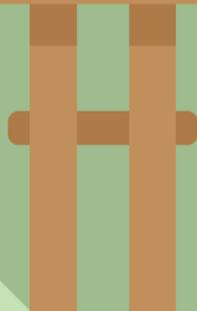
- de se déplacer sur la carte
- de commencer une attraction et donc de lancer un mini-jeu
- d'afficher les statistiques
- de quitter

# Statistiques / classement

Un tableau récapitule les statistiques (meilleures performances globales) pour chaque attraction, ainsi que les **performances** de chacun des joueurs. La performance de chaque attraction peut se mesurer en temps, en nombre de canards pêchés, etc.

## Sauvegarde/chargement

Les meilleures performances de tout les temps sont sauvegardées dans un fichier et affichées pour permettre une comparaison partie après partie.





# Tickets

Chaque joueur commence le jeu avec 5 tickets. Chaque attraction coutre un ticket, et peut rapporter un ou plusieurs tickets au(x) gagnant(s).

A photograph of a person's hands holding a large, shiny gold trophy. The trophy has a wide base, two handles on the sides, and a decorative rim at the top. The background is a solid blue.

# Victoire

Lorsqu'un joueur n'a plus de ticket, il perd et c'est l'autre joueur qui remporte la victoire, les statistiques sont alors affichées.

# ATTRACTI~~O~~N~~S~~



Vous devez implémenter au moins une attraction par membre d'équipe



 Facile

# Pêche aux canards

*Les deux joueurs jouent à tour de rôle.*

Objectif	Attraper le plus de canards dans un temps imparti
Victoire	Le gagnant remporte un ticket
Performances	Nombre de canards pêchés
Spécificités	Le curseur de la souris joue le rôle de canne à pêche. Les canards doivent être déposés dans un panier



 Facile

# Taupe-la !

*Les deux joueurs jouent à tour de rôle.*

Objectif	Des taupes apparaissent aléatoirement. Chaque joueur doit toucher un maximum de taupes avant qu'elles ne disparaissent
Victoire	Le joueur ayant touché le plus de taupes remporte un ticket
Performances	Nombre de taupes touchées dans le temps imparti
Spécificités	Les taupes ont à chaque fois la même probabilité d'apparaître. Elles restent par exemple entre 1 et 2 secondes

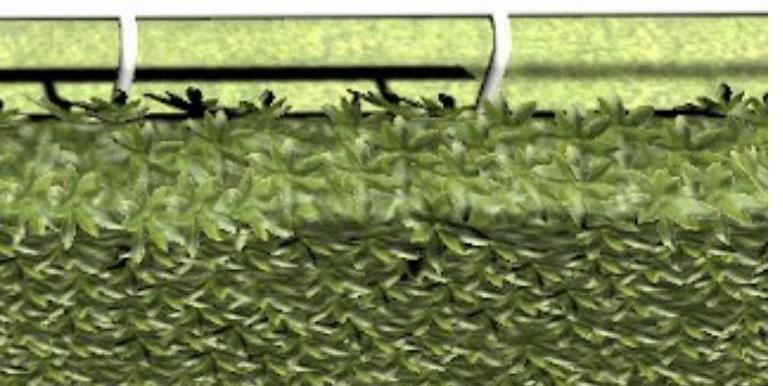
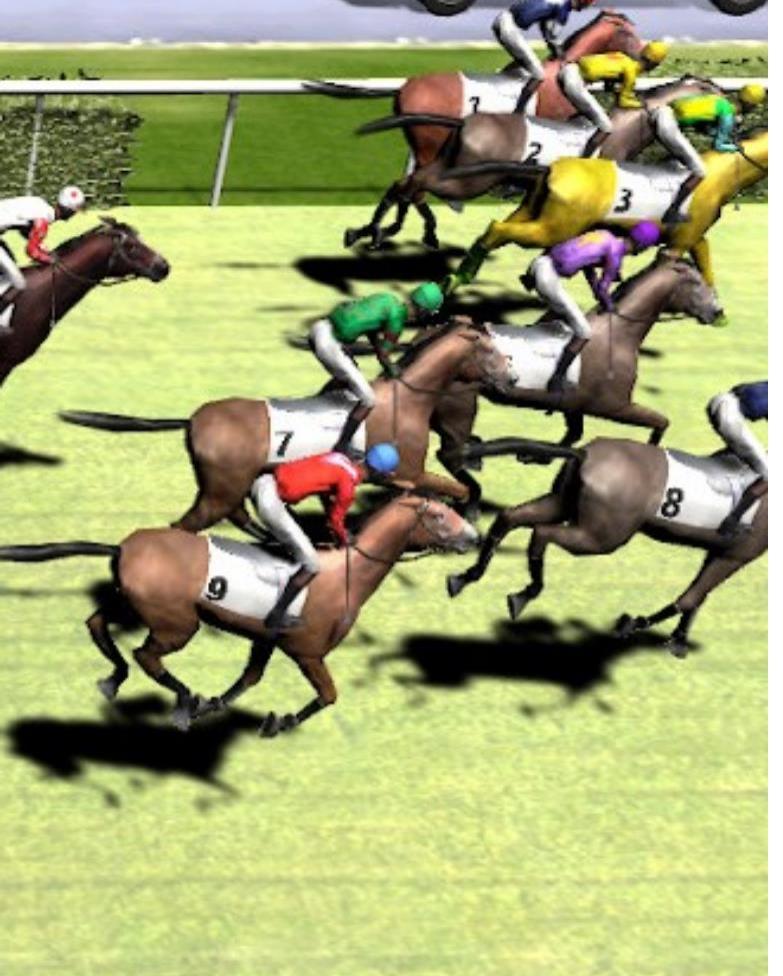


Facile

# Tir aux ballons

*Les deux joueurs jouent à tour de rôle.*

Objectif	Etre le plus rapide pour tirer sur tous les ballons
Victoire	Le joueur le plus rapide remporte un ticket
Performances	Temps
Spécificités	Les ballons peuvent changer de direction à tout moment



Facile

# Paris hippiques

Moyen

si les chevaux sont animés.

*Les deux joueurs font leur pari avant le début de la course.*

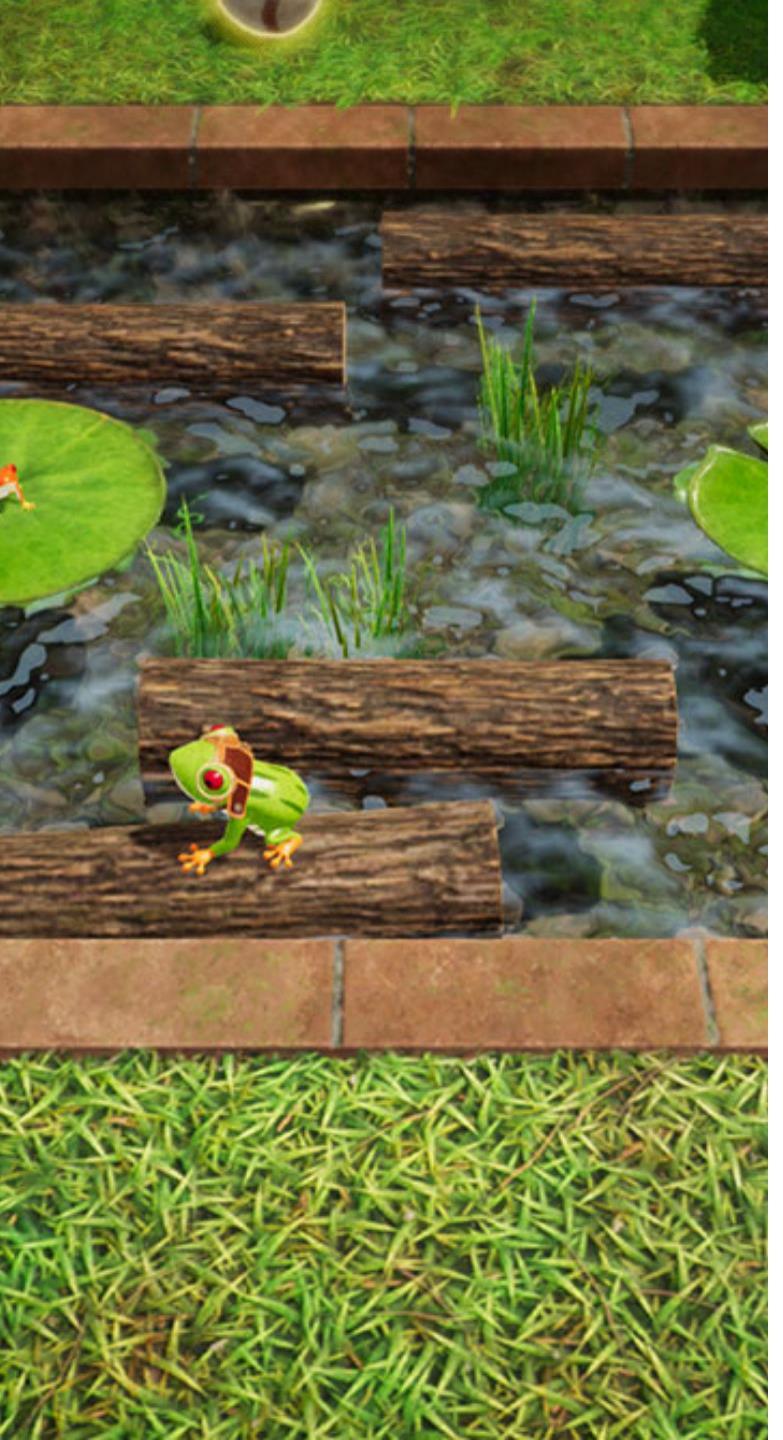
<b>Objectif</b>	Parier sur le cheval le plus rapide
<b>Victoire</b>	Le(s) joueur(s) ayant parié sur le bon cheval remportent un ticket
<b>Performances</b>	Nombre de victoires



 *Moyen* **Jackpot**

*Seul un joueur joue.*

<b>Objectif</b>	Le joueur baisse la manivelle en espérant obtenir trois symboles identiques
<b>Victoire</b>	Trois symboles identiques accordent ou retirent un ticket (en fonction du symbole répété)
<b>Performances</b>	Nombre de victoires globales



Moyen

# Traversée de rivière

*Les deux joueurs jouent à tour de rôle, ou en même temps.*

<b>Objectif</b>	Traverser la rivière sans tomber dans l'eau ni être emporté hors de l'écran par le courant
<b>Victoire</b>	Les joueurs ayant traversé la rivière remportent un ticket
<b>Performances</b>	Temps (0 si échec)
<b>Spécificités</b>	Le joueur bouge avec le rondin sur lequel il est



 Moyen

# Palais des glaces

*Les deux joueurs jouent à tour de rôle ou en même temps.*

Objectif	Chaque joueur doit, à tour de rôle, trouver le chemin vers la sortie du labyrinthe
Victoire	Le joueur le plus rapide remporte un ticket
Performances	Temps
Spécificités	Le labyrinthe est généré aléatoirement (voir algorithmes en ligne). Chaque joueur joue sur le même labyrinthe



 Difficile

# Héro de la guitare

*Les deux joueurs jouent à tour de rôle.*

Objectif	Tenir le plus longtemps sans se tromper/rater une note (au clavier)
Victoire	Le joueur qui tient le plus longtemps remporte un ticket
Performances	Temps
Spécificités	Les ronds peuvent apparaître de façon aléatoire ou programmée sur une musique

 Difficile

# Serpent

*Les deux joueurs jouent à tour de rôle ou en même temps.*



Objectif	Devenir le plus long en ramassant des éléments et tenir le plus longtemps sans se rentrer dedans
Victoire	Le joueur ayant ramassé le plus d'éléments remporte un ticket
Performances	Temps et nombre d'éléments récoltés
Spécificités	Le serpent sera géré par <b>liste chaînée</b>

# Travail à faire / livrables

Vous serez évalués lors de votre soutenance sur la qualité de votre **conception** (anticipation), votre **organisation**, la qualité de votre **implémentation**, la **jouabilité** de votre jeu, sa **fluidité**, ainsi que sur votre **présentation orale**.

Voici les différents livrables à rendre sur Boostcamp (*les deadlines y sont listées*):

- Preuves de conception (.pdf)

Aucune rédaction n'est demandée, ce document regroupe toutes vos recherches de conception (DTI, ACD, Algorithmes, schémas...) effectuées avant votre implémentation.

- Code source (.zip), téléchargé depuis GitHub.com
- Slides de soutenance, réalisées avec **Marp** (.pdf)
- *Bonus : vidéo et infographie (jusqu'à 1 point)*

# Git

Votre code devra être versionné avec Git, **du début à la fin**. Les commits doivent être réguliers et bien nommés.

Le dépôt de votre équipe sera créé et géré par **GitHub Classrooms** (voir les instructions sur Boostcamp). Ne le créez pas vous-même.

 Chaque membre de l'équipe doit envoyer lui-même son code sur GitHub, **régulièrement**. Une personne n'ayant aucun commit se verra attribuer la note de 0/20 au projet.

 Un cours Git est disponible sur Boostcamp.

# Soutenance

La soutenance se découpera en trois phases :

- Présentation (slides)
- Démonstration
- Questions / réponses

Votre présentation devra obligatoirement être faite en utilisant **Marp (documentation)**.

Les contenu des slides vous sera donné sur Boostcamp, dans un template Marp que vous adapterez à votre projet.

# Marp

+ VS Code

Marp s'intègre de façon optimale à Visual Studio Code.

1. Installez VS Code
2. Installez l'extension Marp for VS Code
3. Le dépôt de votre projet contient un dossier presentation, ouvrez-le dans VS Code en tant que nouveau projet pour éditer votre présentation Marp.

 Puisque le dossier presentation se trouve dans le dossier/dépôt Git, ses fichiers sont versionnés. Lorsque vous éditerez le contenu du fichier presentation dans VS Code les modifications seront visibles dans CLion ! Vous pourrez alors les envoyer à GitHub.

# Livrables bonus

Les livrables bonus sont à déposer sur Boostcamp. Ils octroient un bonus pouvant atteindre 1 point, en fonction de leur qualité. Ils pourront être présentés lors d'événements école, en Journées Portes Ouvertes par exemple. Ils présentent comment se déroule un projet d'informatique à l'ECE (conception, implémentation, résultat).

## Vidéo

Evitez les plans de type interview et ajoutez si possible des sous-titres.

## Affiche de film (A3, portrait)

## Infographie (A3, portrait)

L'infographie retrace de façon synthétique votre avancée lors de ce projet. Vous présenterez votre conception, votre implémentation, et le résultat attendu.

# Méthodes de travail

## Conception

L'étape de conception est **très importante** : chaque année, des équipes de projet se lancent dans le code sans avoir réfléchi à sa conception et se retrouvent bloqués à 60% du projet car ils ont fait de très mauvais choix algorithmiques (la façon dont sont stockées les données par exemple).

**Avant** de commencer à coder, vous **devez** tous avoir une vision claire et globale de la façon dont va être organisé/structuré/implémenté votre jeu.

# Mises en garde

## Mémoire

Si vous voulez un jeu fluide, sans latence, vous devez anticiper sa conception et veiller en permanence à vous poser la question suivante : est-ce que ce que je suis en train de faire est lourd pour l'ordinateur ?

Par exemple, il ne faudra pas charger l'image d'un canard autant de fois qu'il y a de canards, une seule fois suffit. Attention aussi à ne pas charger les images dans une boucle, car cela allouerait l'image en mémoire autant de fois que la boucle se répète. Veillez aussi à libérer toutes vos images, polices, etc. Réfléchissez au meilleur endroit où charger l'image pour qu'elle ne le soit qu'une seule fois et seulement si cela est nécessaire.

Ne mélangez pas la mise à jour des données et l'affichage : mettez à jour les données, puis appelez une fonction qui s'occupera de redessiner la totalité de la fenêtre.

# Mises en garde

N'attendez pas le dernier moment pour fusionner vos jeux, ça ne marchera pas.

Au début, passez du temps à vérifier comment vous pouvez faire un jeu composé de mini-jeux, en sachant que l'on doit revenir à la carte à la fin d'une attraction, en récupérant les performances.

Une fois que ça fonctionne, vous pourrez commencer à implémenter les minis-jeux.

Attention à ne pas faire la même chose plusieurs fois inutilement : on installe le clavier une seule fois dans tout le projet, pas dans chaque mini-jeu...

Est-ce que vous comptez utiliser la même file d'événements dans tous les minis-jeux ou en créer une nouvelle à chaque fois ? Réfléchissez bien !

# CMakeLists.txt pour *Windows* et *MacOS*

```
cmake_minimum_required(VERSION 3.10)
project(ece_world C)

set(CMAKE_C_STANDARD 99)

add_executable(ece_world main.c)

IF(WIN32)
    target_link_libraries(ece_world -lmingw32 -lallegro-5.0.10-monolith-md-debug)
ELSE()
    INCLUDE_DIRECTORIES(/usr/local/Cellar/allegro/5.2.8.0/include)
    LINK_DIRECTORIES(/usr/local/Cellar/allegro/5.2.8.0/lib)
    file(GLOB LIBRARIES "/usr/local/Cellar/allegro/5.2.8.0/lib/*.dylib")
    message("LIBRARIES = ${LIBRARIES}")
    TARGET_LINK_LIBRARIES(ece_world ${LIBRARIES})
ENDIF(WIN32)
```

💡 Remplacez `/usr/local/Cellar/allegro/5.2.8.0` par le chemin d'installation d'Allegro sur votre Mac.